Dévelo'Pont
Évaluation - Octobre 2017

17 octobre 2017

L'exercice 1 est obligatoire, à faire en premier et individuellement.

L'exercice 2... aussi !

Pour la suite les exercices se décomposent en trois aspects : Backend, Frontend, Trucs Avancés Super Balaises.

Le Backend va consister à créer une API. Si vous avez du mal ou préférez vous concentrer sur le Frontend, vous êtes autorisés à utiliser l'API de quelqu'un d'autre. De toute façon utiliser votre API ou celle d'un autre ne change rien au travail à accomplir pour utiliser cette API.

Et ce n'est pas une erreur si les derniers exercices de SQL ont été fait sur la même base de donnée. La connexion MariaDb est la suivante :

mysql ://DS2 :DS2@developont.fr :3306/DS2

Serveur=developont.fr, port=3306, utilisateur=DS2, mot de passe=DS2, base de donnée=DS2

PHPMyAdmin ici : https://developont.fr/phpmyadmin/

1 Validation, accessibilité.

Vous trouverez un fichier *page.html*, qui inclus un fichier *page.css*.

L'affichage de cette page est correct sous Firefox, comme indiqué dans le pied-de-page, mais le code est laid, et l'affichage est différent par exemple sous Chrome.

L'objectif de ce premier exercice va être de nettoyer le code HTML, de le rendre valide W3C, d'utiliser les balises modernes HTML5, et les styles CSS3 avec lesquels vous jouez depuis 5 mois.

Le code final doit répondre à ces critères :

— ne pas s'afficher significativement différemment du code originel sous Firefox ;
— s'afficher à l'identique au premier coup d'oeil sous Chrome au minimum ;
— ne pas utiliser de classes CSS inutiles ;
— ne pas utiliser de styles CSS spécifique à un navigateur ;
— ne pas utiliser la moindre balise DIV, mais uniquement des balises contextuelles donnant des indications sur la structure du contenu ;
— ne pas utiliser la moindre balise HTML servant uniquement à la mise en page (par exemple pas de <DIV class='vertical-space'> pour mettre une marge verticale entre deux éléments) ;
— avoir un code correctement indenté, et donc clairement lisible ;
— être plus petit que le fichier d'origine, en octets, et en nombre de lignes ;
— être un minimum responsive (sans entrer dans les détails), c'est à dire ne pas avoir d'affichage absurde - avec des textes qui se chevauchent par exemple - si on réduit la taille de la fenêtre vers 600 pixels de large, ou si on agrandit le texte pour mieux lire.

Ayez en tête les possibilité d'ajout de contenu, et surtout de modification du contenu en javascript !

2 Mettez en production !

Mettez votre travail dans un dépôt git initié pour l'occasion, commitez l'exercice précédent, et clonez le dépôt sur le serveur developont.fr dans votre répertoire www/ avec comme nom DVD.

Les fichiers seront donc accessible https ://developont.fr/~votrenom/DVD/

Faites ça en exploitant les fonctionnalités magiques de git, qui permet de sauvegarder son travail et de l'envoyer ailleurs : https://stackoverflow.com/questions/4142936/git-clone-from-local-to-remote

Inutile de passer par gitlab/github, SSH c'est magique, utilisez SSH.

3 CRUD, les bases du SQL

-> Vous pouvez passer directement à l'exercice 6, il n'est pas nécessaire de faire les exercices dans l'ordre, et vous pouvez revenir sur un exercice pour compléter une réponse.

Sur le serveur MariaDb de developont.fr, vous trouverez une base de donnée nommée DS2, qui est une collection de DVD, avec des consommateurs, des catégories, etc. pour la gestion d'une boutique de vente de DVD.

Cet exercice consiste à écrire quatre types de requêtes SQL correspondant à la base d'utilisation d'un modèle de donné, le modèle CRUD : Create, Read, Update, Delete.

Vous devez créer :

- un nouveau CUSTOMERS ;
- un nouveau PRODUCTS dont le titre ne commence *pas* par « a » ;

Vous devez lire :

- la liste des PRODUCTS correspondant à une CATEGORIES donnée ;
- la liste des CATEGORIES pour lesquelles un CUSTOMERS a emprunté un DVD dans son CUST_HIST. (Donc tel client a emprunté des DVDs appartenant aux catégories Enfants, Horreur, et Sports par exemple)

Vous devez mettre à jour :

- les données d'un CUSTOMERS ;
- lui assigner un historique d'achat dans CUST_HIST, avec une ORDERS correspondante.

Vous devez supprimer :

- un PRODUCTS qui n'a jamais été acheté, et dont le titre commence par a, vérifiez s'il est en stock, le cas échéant supprimez-le du stock aussi ;
- un CUSTOMERS au nom imprononçable, avec la liste de toutes ses ORDERS, et son CUST_HIST.

4 REST plus qu'à faire le PHP

Vous trouverez un fichier api.php qui est une base d'API en PHP (sérieux), qui permet de lister les DVD par catégorie. On ne demande pas ici de corriger le code, mais de l'étendre.

Vous devez créer des interfaces supplémentaires pour effectuer à partir d'une requête HTTP les requêtes construites à l'exercice précédent, ou d'une autre requête de votre choix, tant que ça fonctionne.

La première interface est pré-écrite et permet de traiter le cas d'une requête PUT vers : api.php ?add=CUSTOMERS qui prend en paramètres POST les colonnes de la table CUSTOMERS, et servira à ajouter un nouveau client.

On vous demande de faire au minimum une interface pour chaque lettre de CRUD, donc une insertion (changez add=CUSTOMERS par add=CATEGORIES pour ajouter de nouvelles catégories par exemple !), une lecture (en plus de celle qui existe déjà), une mise à jour, et une suppression.

Les données retournées seront *toujours* en JSON, sauf si isset($_GET['help']), comme indiqué dans le code déjà écrit. Dans ce cas un texte documentant l'appel, ses paramètres, son utilité, et ses valeurs de retours, sera affiché. Inspirez-vous de ce qui existe, gagnez du temps, et travaillez sur un projet existant sans chercher à le réécrire (important en entreprise, très important).

5 On change d'échelle, le travail à la chaîne c'est l'avenir du prolétaire informaticien.

L'objectif ici est d'étendre l'API pour qu'elle accepte de faire des traitements en masse.

La lecture en masse est naturelle, on affiche des listes avec des filtres, par exemple les DVD de science-fiction à moins de 10€. Ce qui est intéressant c'est de pouvoir ajouter, supprimer ou modifier des lots d'éléments, sans avoir à le faire un par un.

Modifiez une méthode d'ajout d'élément de votre API pour qu'elle accepte un paramètre CSV, qui sera un fichier au format CSV, avec sur chaque ligne un nouvel enregistrement, permettant ainsi d'ajouter plusieurs éléments d'un seul coup.

Vous pouvez choisir d'utiliser une fonction spéciale, plutôt que par exemple add=CUSTOMERS faire batchadd=CUSTOMERS pour ne pas tout mélanger dans votre code.

Un fichier CSV c'est simplement une liste d'élément, dont chaque propriété est mise côte à côte et séparée par en général un point-virgule ou une virgule :

```
Isaac;Asimov;107 Grande Rue;;Pont-en-Royans;;38680;FR;;issac.asimov@developont.fr;;;;;R.Daneel.Olivaw;LawZero;97;;M
Philip;Dick;375 Rue du Temple;;Pont-en-Royans;;38680;FR;;philip.k.dick@developont.fr;;;;;SFGuru;Ubik;89;;M
Ursula;Le Guin;432 Rue de l'Horloge;;Pont-en-Royans;;38680;FR;;ursula.k.le.guin@developont.fr;;;;;Terremère;OrsinianTales;87;;F
```

6 Rendre la page dynamique

D'un côté l'API, de l'autre côté l'interface. On va donc ici travailler sur l'interface, qui sera web. En partant de la page HTML initiale, l'exercice ici est de la rendre dynamique. Commencez par la renommer pour garder le résultat de l'exercice 1 visible sans altérations.

Dans un premier temps, modifier la liste affichée, selon un ou deux critères de sélection, par exemple afficher des clients, des commandes, ou des DVD selon la catégorie, mais dans tout les cas, quelque chose de disponible via l'API. La requête pour lister les DVD par catégorie existe déjà, donc tout le monde peut l'exploiter.

Dans un second temps, ajouter des formulaires pour ajouter des éléments, ou en modifier, ou en supprimer, selon ce qui est disponible dans votre API.

Ici trois chemins s'offrent à vous :

— tout faire en PHP, sélectionner une liste ou valider un formulaire recharge la page qui est générée en PHP ;
— tout faire en Javascript (avec ou sans jQuery), faire une requête ajax vers l'API, et modifier la page à la volée ;
— utiliser un framework Javascript, type Angular, Vue ou Riot, et essayer de construire une application web basée sur l'API.

Très important : vous n'êtes *pas* obligé d'utiliser votre propre API, et pouvez utiliser celle des autres, voire en utiliser plusieurs qui ne proposent pas les même fonctionnalités.

Cependant si vous choisissez la voie du PHP, il est possible d'appeler directement les fonctions PHP de l'exercice 4, sans passer par les requêtes REST. Essayez dans ce cas d'avoir malgré tout une API fonctionnelle, et de ne pas copier-coller les fonctions, et les requêtes SQL. À noter aussi qu'il est tout à fait possible de faire des appels à une autre API en PHP, en faisant des requêtes GET et POST avec la bibliothèque CURL : http://php.net/manual/fr/book.curl.php.

À noter que les bibliothèques Javascript les plus courantes sont disponibles dans le répertoire js.

7 Faites du beau !

Faites vous plaisir et rendez tout ça beau, dynamique, vivant, ergonomique, classe, avec des effets, des animations, des fondus, des super-héros qui traversent l'écran, et un clone d'Arkanoid en pixel-art dans le footer (ou un clone de pac-man, ou de super-mario, ou de defenders, ou de tetris, ou de pong, ou de rien du tout ! Mais faites-vous plaisir, c'est un ordre.).

8 Mise à jour dynamique avec les websocket

Pour les plus aventuriers, créez un serveur websocket (node.js ou python sont les solutions les plus simples), créez une connexion websocket en Javascript, et modifiez une page par exemple listant les derniers clients, dès qu'un nouveau client est ajouté dans la base de donnée.

Vous pouvez ensuite attendre qu'un autre apprenant ajoute un client, et le voir apparaître immédiatement sur votre page, ou simplement ouvrir une autre page de navigateur et le faire vous-même.

Pas besoin de gérer plusieurs cas, l'objectif n'est pas de faire un framework complet, mais simplement de réussir à faire fonctionner un cas particulier unique.

Bravo, vous avez fait le tour des buzzwords à la mode sur le web !

Conseils, à lire avant d'agir

Faites les exercices 1 et 2 la première matinée, ce sera notre dernière réelle évaluation de vos compétences HTML/CSS.

Pour tout le reste vous avez jusqu'à la fin de la semaine pour rendre au minimum soit les exercices 3 et 4, soit les exercices 6 et 7, à jongler avec la Code Week, les tables rondes du numérique et vos projets ! Vous pourrez continuer la semaine suivante sur les parties que vous n'aurez pas traitées, mais les retours se feront déjà sur les parties traitées.

Donc refaites l'exercice 2 tous les jours, dès que vous avez avancé sur un exercice, faites un commit, et envoyez sur developont.fr ! Le premier qui me dit qu'il a perdu son travail suite à une fausse manip fini dans la Bourne sans bouée, pas d'excuse, vous commitez, vous envoyez sur le serveur. *Tout le temps.*

N'envoyez pas de réponse par mail, on ira voir sur vos URLs : https://developont.fr/~votrenom/DVD/, par commodité, ne mettez pas de fichier index.html ou index.php, qu'on puisse voir directement en ligne les fichiers existants.

Commentez vos API, et indiquez sur le canal de discussion de la promo quand vous avez une API qui fonctionne, même incomplète, pour permettre aux autres de les utiliser et d'avancer sur la partie Frontend.

De même n'hésitez pas à indiquer que vous avez des Frontend fonctionnels pour permettre à d'autres de tester leurs API.

Et dernier point important : il vous est parfaitement autorisé de virer les styles par défaut de la page de l'exercice numéro 1 dès que vous l'aurez validé une première fois, mais l'exercice 1 doit respecter la charte graphique initiale !