# Projet Covoiturage TP 4

## Description du thème

Ce TP est le quatrième d’une série mettant en œuvre le développement mobile en utilisant la bibliothèque jQuery Mobile.

|  |  |
| --- | --- |
| Propriétés | Description |
| **Intitulé long** | Des TP permettant la découverte d’un mini-framework mobile jQuery Mobile (JQM) |
| **Formation concernée** | BTS Services informatiques aux organisations |
| **Matière** | SLAM 2, PPE, SLAM 4 |
| **Présentation** | Les TP proposent de développer une application mobile cross-plateform à différentes itérations du cycle de développement |
| **Notions** | Savoirs  • D4.1 - Conception et réalisation d’une solution applicative  • D4.2 - Maintenance d’une solution applicative  Savoir-faire  • Programmer un composant logiciel  • Exploiter une bibliothèque de composants  • Adapter un composant logiciel  • Valider et documenter un composant logiciel  • Programmer au sein d’un framework |
| **Pré-requis** | Développement web, PHP, JavaScript |
| **Outils** | Un environnement de développement pour le web, Firebug pour suivre les appels jQuery et Ajax |
| **Mots-clés** | Application mobile, jQuery, jQuery Mobile, Ajax |
| **Durée** | 3h40 |
| **Auteur(es)** | Patrice Grand |
| **Version** | v 1.0 |
| **Date de publication** | Mars 2014 |

## Énoncé

La première itération est terminée (cf. les 3 premiers TP) ; les différents tests sont concluants concernant la stabilité de la bibliothèque jQuery Mobile et sa portabilité sur les plateformes cibles (IOS, Android, WindowsPhone).

Dans cette deuxième itération l’accent sera mis sur le temps de réponse de l’application.

Un constat

Tout en étant assez fluide, l’application pourrait voir ses performances améliorées ; 5 appels au serveur sont faits pour demander le rechargement de l’index et ceci provoque un temps d’attente certes très supportable mais qui pourrait être fortement réduit. Des mesures ont été effectuées ; elles sont présentées dans l’annexe 1.

D’autre part, la tendance actuelle des applications web-mobile s’oriente vers une architecture mono-document : un seul document HTML est chargé au démarrage de l’application, il contient toutes les pages et des appels Ajax sont faits pour charger les données dans la page active.

C’est cette option qui va guider la deuxième itération.

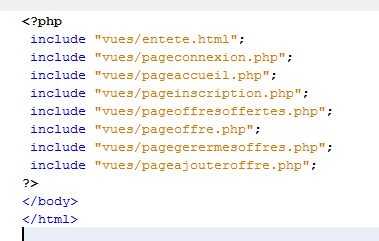
Votre chef de projet vous demande de rédiger une courte note analysant le trafic présenté en annexe 1 et anticipant les gains en terme de rapidité dans l’hypothèse retenue pour cette seconde itération.

## Question 1

*Rédiger la note demandée. Cette tâche est estimée à 30 minutes.*

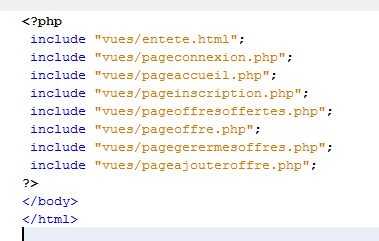
Les bases de la nouvelle version

Cette nouvelle version va présenter un index très allégé, sans code PHP (si ce n’est les *include* présents pour raison de lisibilité et d’organisation des pages).



La plupart des pages devront bien sûr être modifiées et toilettées de leur code PHP.

Le développement va être divisé en 3 parties autour des fonctionnalités des pages :



3

2

1

Partie 1

Cette partie ne nécessite que peu de transformation (Ajax a déjà été utilisé pour les pages connexion/accueil), si ce n’est une page accueil à ajouter qui ne contiendra que le menu et le logo :



## Question 2

*Votre chef de projet vous demande de commencer par écrire le code de cette page, dont l’id sera « pageaccueil ». Il estime que 10 minutes seront suffisantes.*

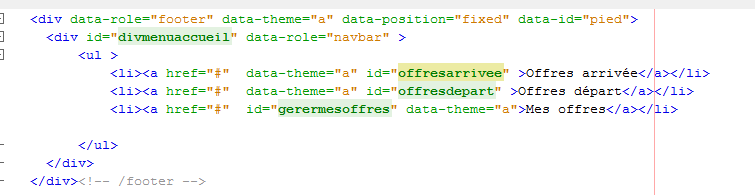
Pour terminer cette partie 1, votre chef de projet vous demande d’utiliser une grande partie du code de l’application existante ; l’essentiel des modifications portera sur les liens des pages et bien sûr le comportement associé dans le fichier *monJs*.

## Question 3

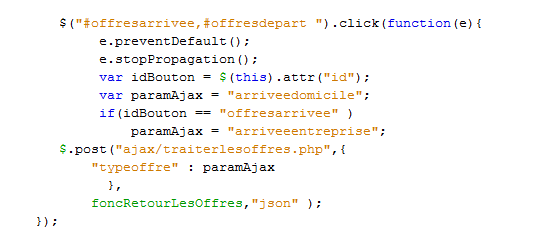
*Réalisez cette tâche que votre chef de projet estime à une heure.*

Partie 2

Cette partie est plus délicate, il faudra remplacer le chargement de l’index par un appel Ajax pour récupérer les données. Le menu (pied.html) doit être modifié, vous avez décidé de le remplacer par du code qui ne fait plus appel à l’index :

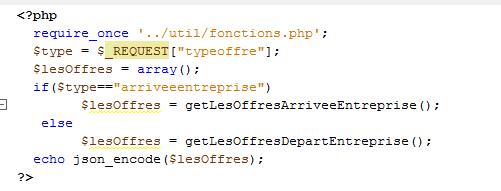


Vous continuez par gérer les événements click sur les deux premiers liens *href* avec le code jQuery suivant :

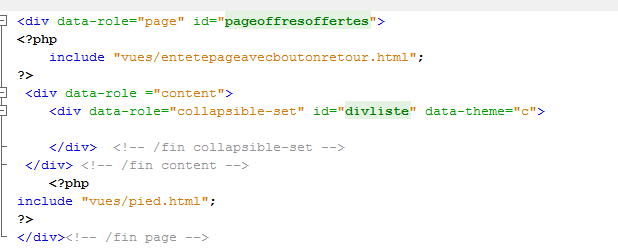


Le sélecteur $("#offresarrivee, #offresdepart ") permet d’abonner les deux éléments d’identifiant *offresarrivee* et *offresdepart* au même événement : vous décidez ainsi d’informer l’appel Ajax de la nature de la demande.

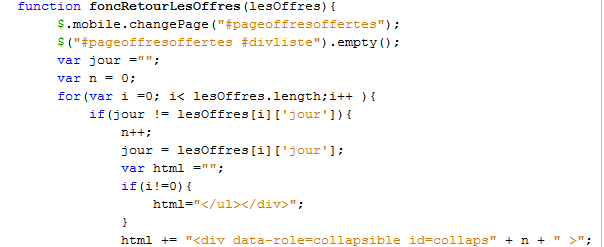
La fonction Ajax va pouvoir fournir la bonne ressource ; vous poursuivez en écrivant cette nouvelle fonction Ajax *traiterlesoffres.php* :



Vous préparez votre page permettant d’afficher les offres :



Il ne vous reste plus qu’à écrire le code jQuery dans la fonction *foncRetourLesOffres*. Elle commence par le code :



Remarque : votre chef de projet vous conseille d’utiliser des sélecteurs jQuery très précis car comme toutes les pages sont chargées, il a un risque que le code que l’on pensait concerner un sélecteur concerne un autre qui a le même id. C’est pourquoi, la méthode *empty* (qui vide la liste) s’applique à la *divliste* de la page *pageoffresoffertes.* Ce qui par ailleurs améliore les performances car jQuery filtre déjà la recherche par l’id de la page.

Il vous demande aussi d’être attentif à la création des objets jQuery ; ceci se fait en deux temps, d’une part~~ge~~ il y a interprétation du code HTML présent dans la page active et ensuite une seconde *passe* permet de transformer les éléments HTML en éléments jQuery (ceci se fait au chargement de la page). La seconde *passe* se déroule sans problème si tous les éléments HTML sont présents au moment du chargement ; par contre quand c’est le code jQuery qui insère une *listview* par exemple il y a lieu de générer un événement de création dans le code. Ceci se fait par :

*$("#<conteneur>").trigger('create');*

## Question 4

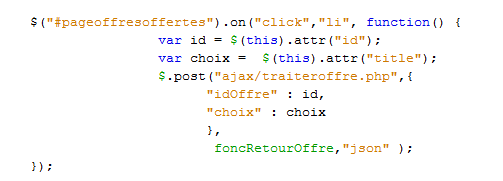
*Terminez l’écriture de la fonction foncRetourLesOffres que votre chef de projet estime à une heure.*

Pour cette partie 1, il ne vous reste plus qu’à afficher le contenu de l’offre ; la page de présentation *pageoffre.php* restera identique.

Il vous précise quelques points :

* Il ne faut pas utiliser l’événement *click* directement mais passer par la méthode *on* qui s’exécutera lorsque les éléments jQuery (ici les *li*) seront construits
* La méthode *on* s’utilise à partir d’un élément conteneur (la page *pageoffresoffertes*) et concerne un (ou plusieurs) élément(s), ici tous les éléments *li*.

Vous êtes arrivé à écrire la méthode :



## Question 5

*Ecrire la fonction foncRetourOffre qui va permettre de renseigner les différents éléments présents dans la page pageoffre.php. Le temps estimé est d’une heure.*

## Annexe 1 : analyse du trafic avec Firebug

1. Chargement de la page de login et traitement de la connexion :

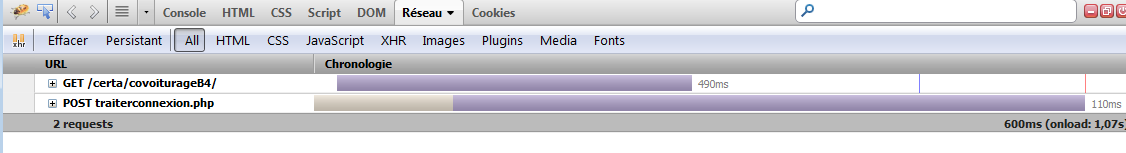


fig 1

Remarque : les barres ont une longueur très approximative, ne se fier qu’aux valeurs

1. Chargement de la page visualisant les offres :
2. Sous Firefox :

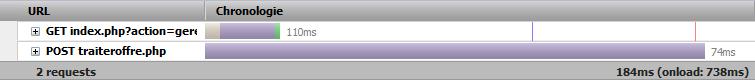


fig 2

1. Sous Chrome :

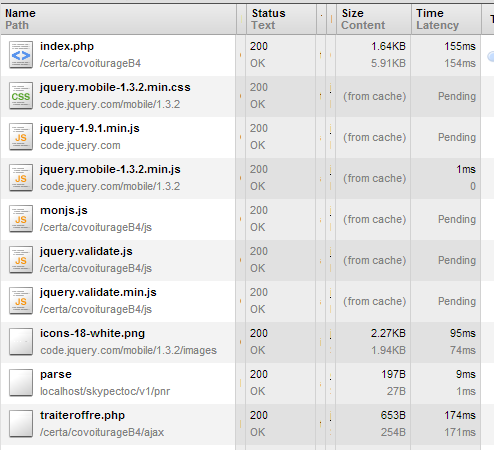


fig 3

1. Chargement de la page de gestion des offres

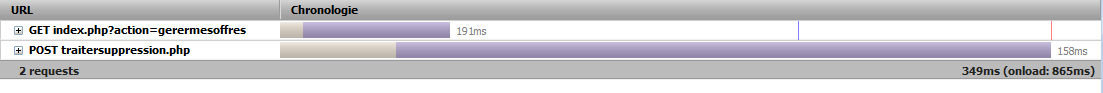


fig 4

## Corrigé

## Question 1

Analyse du trafic.

Cette étude se base sur un essai à un moment donné, avec un état du réseau particulier. Si des mesures étaient effectuées à un autre moment les résultats seraient différents. Plus que les valeurs réelles des activités réseaux, il est préférable de retenir les proportions entre types d’activités.

On peut remarquer (fig 1) que la séquence du premier chargement de l’index prend un certain temps, 490 ms ; on peut noter que le second chargement de l’index (pour afficher les offres) prend moins de temps, 110/150 ms (fig 2 et fig 3). Ceci s’explique par la mise en cache (fig 3) d’un certain nombre de fichiers. Le chargement de la troisième version de l’index (fig 4) conserve ce temps relativement réduit.

Concernant les échanges Ajax (ici les méthodes POST), on constate un gain important.

On peut donc envisager de réduire de manière significative les temps de chargement des pages en supprimant les 5 appels à l’index. Notons néanmoins que dans une architecture mono-document, le chargement initial de l’application (qui contiendra ainsi toutes les pages) sera plus long que dans la version actuelle.

## Question 2, 3, 4, 5

Voir le code.