Département Télécommunications, Services et Usages

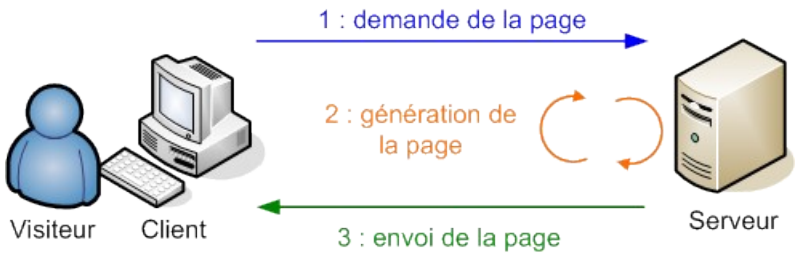
TD WEB SI - 3 & 4

1. Objectifs des TD/TP WEB SI :

- Partie 1 : configuration, lancement et tests d'utilisation du serveur Web Apache
- Partie 2 : développement d'un framework de génération de pages Web dynamiques côté serveur en PHP
- Partie 3 : intégration dans le framework de pages Web dynamiques côté client en Javascript

Ces trois objectifs se dérouleront sur deux séances de TD machine et une séance de TP. Les fichiers de configuration, le code du framework ainsi qu'un compte-rendu en pdf de 3 pages feront l'objet d'un rendu évalué.

2. Principes de fonctionnement d'un serveur Web



Un serveur Web permet la publication de ressources. Il s'exécute sous forme d'un démon écoutant les requêtes de client sur un port fixe en utilisant les protocoles HTTP (port 80) et HTTPS (port 443).

Obtenir une ressource consiste à :

- L'identifier en utilisant une URI de la forme :
  protocole://nom-machine:port/nom-ressource
  (http://ares-lemouel-1.citi.insa-lyon.fr:8080/framework-web par exemple)
- La générer : une ressource peut être un élément statique, comme un fichier, la génération consiste alors juste à retourner ce fichier. La ressource peut également être dynamique et le résultat de l'exécution d'un programme (la ressource catalogue-livres sur un site d'un éditeur est par exemple le résultat d'un programme faisant un accès sur un autre serveur de base de données). Cette génération peut s'effectuer par tout type de programmes : compilés, interprétés dans différents langages.
- Retourner la ressource sous un certain format : une ressource peut être destinée à différentes utilisations : la visualisation dans un navigateur, l'inclusion dans une application, l'exécution sur le client, etc. Il convient donc de typer cette ressource de

manière à ce que le client sache comment l'interpréter. Une ressource peut contenir des données (par exemple image/jpeg), du code (par exemple application/javascript) ou une composition des deux.

La partie 1 du TD va consister à examiner comment configurer apache pour faire cette correspondance entre une URI et les ressources mises à disposition dans le serveur. La partie 2 du TD va consister à développer un framework de génération de page Web dynamiques en utilisant le langage interprété PHP. La partie 3 du TD va consister à bien structurer ce framework pour générer des ressources contenant des données HTML et du code Javascript.

3. Préambule

- L'archive framework-3TC-WEB.tar.gz disponible sur Moodle contient le squelette des fichiers de configuration Apache, du framework Web POKEMON.

- La documentation PHP est disponible sur http://www.php.net/ , http://fr.php.net/

- Une documentation Javascript est disponible sur https://developer.mozilla.org/fr/JavaScript

4. Partie 1 : Apache

1. Configuration

- Dans l'archive, le répertoire etc/apache2 contient une recopie des fichiers de configuration global d'une machine disponibles dans /etc/apache2.
- Modifiez dans le répertoire etc les fichiers de configuration pour bien positionner les valeurs des variables 'User', 'LockFile', 'PidFile', 'ErrorLog', 'DocumentRoot', etc (!) avec les bons chemins (remplacer les /usr /etc /var par vos répertoires locaux de configuration).
- Modifier le port d'écoute du serveur dans la variable Listen. Les ports de 1 à 1024 sont réservés au root (ports de protocoles standards). La variable Listen est par défault positionnée sur le port 80. Choisissez un port supérieur à 1024 (par exemple 8080).
- Exécuter en mode utilisateur le serveur Apache avec la commande:

```
/usr/sbin/apache2  -D DEFAULT_VHOST -D INFO -D LANGUAGE -D PHP5
                   -d /usr/lib/apache2
                   -f $TP_DIR/etc/apache2/httpd.conf
                   -k start
```

- Vérifiez son exécution avec la commande : ps aux | grep apache
- Tester le bon fonctionnement du serveur en connectant un navigateur Web sur http://localhost:8080/ou en exécutant wget http://localhost:8080/ avec la page index.html

- Tester le bon fonctionnement du module Apache PHP en vous connectant sur http://localhost:8080/index.php

2. .htaccess

Les fichiers .htaccess sont des fichiers de configuration d'Apache permettant de définir des règles dans un répertoire et dans tous ses sous-répertoires. Ils peuvent être utilisés pour interdire l'accès au répertoire ou à certaines ressources, pour protéger un répertoire par un login et mot de passe, ou pour changer la correspondance entre l'URI et la ressource visée.

- Dans votre répertoire var/www/localhost/htdocs, tester l'interdiction d'accès par une machine précise d'un autre binôme, tester l'interdiction d'accès à toutes autres machines que localhost.
- Dans votre répertoire var/www/localhost/htdocs, ajouter une image jpeg et tester l'interdiction d'accès à ce type de fichier.
- Dans votre répertoire var/www/localhost/htdocs, tester l'accès à l'aide d'un login et mot de passe.

Les fichiers .htaccess permettent notamment la correspondance entre une certaine URI et une ressource disponible dans le serveur. Ce mapping s'effectue en utilisant dans .htaccess les règles de réécritures (RewriteRule).

- Dans votre répertoire var/www/localhost/htdocs, écrivez une règle de réécriture qui permet de pointer vers la ressource correspondant à votre langue http://localhost:8080/index.html sera redirigé vers http://localhost:8080/index-fr.html ou http://localhost:8080/index-en.html
- Dans votre répertoire var/www/localhost/htdocs, écrivez une règle de réécriture qui permet de raccoucir vos URI http://localhost:8080/list/2010 correspondra par exemple à une redirection vers la page http://localhost:8080/rep1/rep2/download?list=2010

5. Partie 2 : Le Framework POKEMON (PHP On a Killer Environment Mainly Oversimple for Noob … bref un framework simple pour jeune et gentil étudiant TC)

Le répertoire src/framework-3TC-WEB/ contient le squelette du framework. Vous aurez plusieurs 'trous' à remplir pour le faire fonctionner. L'idée du framework POKEMON est que toute URI requête sur le répertoire du framework est redirigée vers index.php. Un routage de la requête est ensuite effectué vers la bonne ressource. Une ressource est implantée selon le modèle MVC (Modèle Vue Contrôleur).

1. Mise à disposition du framework dans le serveur Web

- Dans votre répertoire var/www/localhost/htdocs, effectuez un lien vers le répertoire source du framework. Vérifiez dans votre etc/apache2/vhosts.d/default_vhost.include que l'option FollowSimLinks est bien activée pour htdocs.

2. Règles de réécriture

- Dans votre répertoire src/framework-3TC-WEB/, créez un fichier .htaccess qui redirige toutes les URI vers index.php.

3. Routage des URI

Le modèle qui va permettre de représenter une ressource est de la forme : http://localhost:8080/pokemon/moncontroller/monaction/monid

- Le fichier index.php va faire appel à deux fonctions que vous devez développer dans lib/routing.php

  $route = route_for_request_path($_GET["request_path"]); // Cette première fonction va récupérer l'URI, la parser, et retourner un tableau contenant [« controller » => « moncontroller », « action » => « monaction », « id » => « monid »]

  route_to($route); // Cette fonction effectuera l'invocation de la fonction monaction(monid) sur la classe MoncontrollerController se trouvant dans les répertoires et fichier app/moncontroller/moncontroller.php

4. Une application

Une application se décompose en un Contrôleur, qui réalise le code métier de l'application, et une Vue, qui réalise la mise sous un certain format des données/codes résultats à retourner. Cette séparation des préoccupations permet de développer proprement.

- Tout contrôleur va hériter d'un contrôleur de base, défini dans lib/controller.php, où vous aurez à développer la fonction render_view($view_name, $params). Cette fonction est appelée par chaque contrôleur lorsqu'il veut afficher ses résultats. Cette fonction doit inclure le fichier permettant le rendu se trouvant dans view/moncontroller/view_name
- Tout contrôleur va également posséder une fonction render_error($http_status, $message, $context_map) qui lui permet d'afficher son propre message d'erreur – par exemple lors de la mauvaise vérification de champ d'un formulaire. Cette fonction peut très bien rediriger vers une erreur globale commune à tout les controlleurs qui se trouve dans static/html/error.php
- L'ensemble des contrôleurs peut avoir besoin de parties communes pour le rendu – par exemple une entête de page, un pied de page ou des menu identiques. Ce rendu commun sera effectué par les fonctions link_to_css($name) et render_partial($name, $params), qui se trouve dans lib/controller.php mais en dehors de la classe contrôleur de base. Ces fonctions peuvent très bien inclure des feuilles de styles prédéfines static/css/ et des vues statiques prédéfinies dans apps/views/partials/

A noter: il vous est donné la gestion des sessions dans les fonctions de construction du contrôleur, session_get, session_set, clear_session.

5. A rendre : l'application TODO List

Ce TP sera à rendre et évalué. Les modalités (dates, format archive, etc) vous seront communiquées par mail.

Vous aurez à rendre le framework POKEMON complété, opérationnel avec une application TODO List.

Les spécifications de cette application sont les suivantes :

- liste des éléments (action par défaut) :
  http://localhost:8080/pokemon/todolist/index

- ajout d'un élément dans la liste :
  http://localhost:8080/pokemon/todolist/save/acheterlait/demain

- retrait d'un élément dans le liste :
  http://localhost:8080/pokemon/todolist/done/idtodolist

- remise à zéro de la liste :
  http://localhost:8080/pokemon/todolist/reset