Master 1 informatique – Université Claude Bernard Lyon 1 – mercredi 4 janvier 2017

M1IF 03 – Conception d'applications Web – Examen

Durée : 1 heure 30 – Documents autorisés – Ordinateurs, calculatrices et téléphones portables interdits

Questions de cours (barème : 14 points)

1. Que contient le header « Host » et pourquoi doit-on obligatoirement l'indiquer dans une requête HTTP 1.1 ?
   Il contient le nom de l'hôte (noms de serveurs). En HTTP 1.1, un serveur peut héberger plusieurs hôtes. Une fois la résolution DNS faite, le nom du serveur « disparaît » de la requête. Il faut donc spécifier le nom du serveur que l'on adresse dans ce header.
2. Quel est le point commun entre les headers Last-Modified et ETag ? Citez deux exemples d'applications, l'un où il est plus logique d'utiliser le premier, et l'autre où c'est le second qui est préférable.
   Ce sont des headers de gestion de cache. Last-Modified s'utilise pour une ressource simple (RESTful) à laquelle tous les clients ont accès, et pour laquelle ils ont besoin d'avoir la même version. ETag s'utilise pour des réponses où il faut gérer plus finement le cache, par exemple si tous les clients ne doivent pas avoir la même version, ou si les dates de modification envoyées par le client ne sont pas suffisamment informatives pour retrouver la version à servir.
3. Comment pourrait-on implémenter l'équivalent d'un ou de plusieurs filtres (javax.servlet.Filter) à l'aide de servlets en Java ?
   Implémenter la FilterChain sous forme d'une liste de pathe correspondant à des servlets différentes, utiliser la méthode de service pour le traitement du filtre et request.getRequestDispatcher(chainIterator.next ()).forward(request, response);.
4. Quel est l'intérêt d'utiliser plusieurs types de ViewResolver en Spring Web MVC ?
   Pour la négociation de contenus : chacun aura un mode de templating plus adapté au type de contenus générés.
5. Donnez deux headers HTTP à positionner lorsqu'un serveur renvoie une réponse au sujet s'une ressource RESTful, et expliquez pour chacun d'eux à quelle propriété de REST il correspond.
   Content-Type -> négociation de contenus
   Last-Modified -> gestion du cache
   Link -> HATEOAS…
6. En quoi peut-on dire que la fonction jQuery.jetJSON() n'effectue pas réellement une requête AJAX ? Donnez un exemple de « vraie » requête AJAX en jQuery.
   jQuery.JSON() demande du JSON et non du XML.
   Pour de l'AJAX : jQuery.ajax({dataType : "xml", … })
7. Citez deux exemples de données que l'on peut obtenir avec l'objet window.performance.timing.
   Cf. https://w3c.github.io/perf-timing-primer/#resource-timing

Étude de cas (barème : 8 points)

Suite au dernier pic de pollution, vous avez été engagé-e par la ville de Lyon pour réaliser une application Web de covoiturage local. Cette application devra permettre à ses utilisateurs de :

- Se créer un compte (l'identifiant sera l'adresse email) et s'authentifier
- Proposer des trajets réguliers (par exemple, Ambérieu-La Doua, du lundi au vendredi, à 7h30), en indiquant les dates de début et de fin, ainsi que la parité de leur plaque d'immatriculation
- Visualiser les trajets proposés, et pour chaque trajet, les prévisions météo, de qualité de l'air, et routières (restrictions de circulation)
- Mettre en relation gratuitement les utilisateurs (pas de paiement en ligne à gérer)

Cette application s'appuiera sur Spring Web MVC. Elle sera optimisée pour répondre aussi rapidement que possible aux requêtes des utilisateurs. Toutefois, les vues « principales » (pages HTML) seront générées côté serveur en JSP. Les ressources non critiques seront requêtées en AJAX et traitées côté client. Les prévisions sont fournies par différents services, interrogés par un bean Spring, qui met à disposition des méthodes `String getMeteo(LocalDateTime heure, String ville)`, `String getAir(LocalDateTime heure, String ville)` et `String getTraffic(LocalDateTime heure, String ville)`.

Les questions ci-dessous ne concernent que les deux cas d'utilisation liés aux trajets (proposition et visualisation). Ne traitez pas ceux liés à la création de compte, à l'authentification ni à la mise en relation des utilisateurs.

Conception (barème : 5 points)

8. Listez les différents éléments dont vous aurez besoin pour réaliser la proposition et la visualisation de trajets. Indiquez leurs types (composants Spring (précisez le type), JSP, classes Java, pages statiques, scripts…) et à quoi ils servent.
   CECI EST UN EXEMPLE DE REPONSE – TOUT AUTRE STRUCTURE SENSEE EST TOUT AUSSI VALIDE.
   Barème : 3 points max. pour cette question ; chaque élément recevable et argumenté = ¼ pt
   Contrôleur Spring : TrajetController : gère les requêtes POST/PUT (proposer) et GET (visualiser)
   Bean Trajet : classe Java qui fait le lien avec la BD pour le trajet
   Bean user : idem pour l'utilisateur
   JSP propositionTrajet.jsp : vue constituée d'un formulaire « personnalisé » (inclut le nom de l'utilisateur)
   JSP affichageTrajet.jsp : vue présentant la structure de base d'un trajet + éventuellement les infos du trajet (en fonction du mode d'optimisation choisi) + éventuellement les infos de prévision (idem)
   Classe statique de gestion des prévisions
   Si interrogation en AJAX des prévisions, il faut un contrôleur Spring, un bean qui interroge la classe statique et un outil de templating (JSP) qui renvoie la vue.
   Scripts JS de chargement des infos en AJAX pour la visualisation
   Script JS de validation des formulaires d'envoi
   jQuery si ces scripts en ont besoin
   Feuille CSS
   Fichiers de config Spring (facultatif)
9. Décrivez, à l'aide de diagrammes UML appropriés, leurs communications lorsqu'un utilisateur clique sur un lien lui permettant de visualiser un trajet. Les éléments du SI externes seront modélisés comme des boîtes noires.
   Remarque : pour cette question, vous devez optimiser le chemin critique de rendu (CRP) de l'application et montrer qu'elle réagit en conséquence.
   Barème : 3 pts max ; ½ pt par échange de données bidirectionnel (requête/réponse ou appel de méthode) entre les éléments cohérents et conformes au sujet.
   Bonus : +1 (max) pour optimisation du CRP.
   Faire un diagramme de séquence ou de communication. Dans tous les cas, un diagramme dynamique, représentant :
   Côté client : L'utilisateur clique sur un lien →requête synchrone
   Côté serveur : contrôleur Spring →contrôleur délégué →interrogation du modèle (trajet) →renvoie les infos (date, origine, destination…) du trajet →retour au contrôleur délégué →composition de la vue côté serveur (JSP)
   Côté client : Affichage de la vue → async/defer : script de requêtage asynchrone prévisions →requête(s) asynchrone(s) →callback : affichage des prévisions

Programmation (barème : 3 points)

10. Écrivez la JSP permettant de renvoyer au client la vue principale correspondant à la visualisation d'un trajet.
    Points notés (1/2 pt pour chaque) :
    Syntaxe JSP
    Syntaxe HTML
    Contenu de la balise head : link -> Feuille de style…
    Récupération des données dans les beans
    Templating des données dans la page
    Scripts JS (à la fin du body ou en defer/async)