Master 1 informatique – Université Claude Bernard Lyon 1 – lundi 7 janvier 2018

# M1IF 03 – Conception d'applications Web – Examen

**Durée : 2 heures – Documents autorisés (4 pages max) – Ordinateurs, calculatrices, tablettes, téléphones portables… interdits**

## *Questions de cours (barème : 12 points)*

**Maximum : 1 phrase ET 3 lignes (en caractères lisibles). Les réponses plus longues ne seront pas lues. Il est inutile de recopier les slides du cours.**

1. Quel est l'intérêt de passer des paramètres en `URL-encoded` dans le corps d'une requête ?
   Le décodage se fait de la même façon que s'ils étaient passés dans l'URL -> indépendant de la méthode HTTP.
   Accepté aussi : mettre les paramètres dans le corps de la requête permet de ne pas les faire apparaître dans l'URL et donc de les sécuriser, si le site est accédé en HTTPS.
2. Quel est l'intérêt de passer des paramètres en JSON dans le corps d'une requête ?
   On peut sérialiser des objets plus complexes que des champs de formulaires (tableaux, dictionnaires).
3. Pourquoi un navigateur affiche-t-il une fenêtre de confirmation quand on lui demande d'actualiser une page obtenue après l'envoi d'une requête en POST ?
   Car POST est une méthode avec effet de bord.
4. Que se passe-t-il si l'on requête avec son navigateur une page PHP sur un serveur nginx, mais qu'aucun module *ad hoc* (comme `PHP-FPM`) n'est installé sur ce serveur ?
   Le serveur renvoie le fichier tel quel, sans l'interpréter et avec un type MIME fantaisiste (par exemple application/php). Le client ne sait pas gérer ce type de fichiers et propose à l'utilisateur de le télécharger.
5. Quel est le principal inconvénient des méthodes de load balancing `round-robin` et `least-connected` ?
   Elles nécessitent un mécanisme de gestion des sessions séparé / centralisé.
   Accepté aussi : un mécanisme de centralisation du cache (0,5 pts)
6. Plutôt que de mettre une variable en public dans une servlet, quel mécanisme vaut-il mieux utiliser ?
   La mettre dans le contexte applicatif.
7. Quel est le mécanisme correspondant à la question précédente pour les JSP ?
   scope="application"
   Accepté aussi (si réponse précédente fausse) : utilisation de beans.
8. Décrivez basiquement comment fonctionne un `ViewResolver` en Spring Web MVC.
   Il récupère la réponse du contrôleur, identifie les données du modèle, trouve le fichier correspondant à la vue et envoie les 2 à la classe capable d'interpréter la vue et de générer la réponse.
9. Comment le fait de travailler en « orienté-ressource » permet-il de réduire la taille des URLs ou des paramètres des requêtes envoyées au serveur ?
   En utilisant les méthodes HTTP adéquates, on évite de spécifier les opérations à réaliser dans l'URL.
   Accepté aussi : le nom des paramètres n'a pas à apparaître dans l'URL
10. Pourquoi le nœud racine de la représentation DOM d'un document n'est-il pas l'élément racine du document ?
    Car il peut exister d'autres nœuds au même niveau que la racine du document, comme la déclaration de doctype ou des commentaires.
11. Pourquoi est-il logique d'avoir 2 outils de templating (potentiellement identiques) dans une application AJAX ?
    Car les données manipulées côté serveur et côté client ne sont pas les mêmes.
    Accepté aussi : pour la négociation de contenus.
12. Pourquoi du point de vue de la performance, faut-il éviter les directives `@import` dans les feuilles CSS ?
    Car le téléchargement des feuilles importées ne pourra être lancé qu'après les phases téléchargement, création de jetons et analyse lexicale de la feuille contenant la directive, retardant d'autant la construction du CSSDOM.

## *Étude de cas (barème : 10 points)*

Grâce à l'application de chat que vous avez brillamment réalisée en TP, le Gouvernement a décidé de vous confier la réalisation d'une partie de l'application Web qui permettra à chacun.e de participer au « grand débat national ». Durant celui-ci, « Des débats se tiendront […] en ligne, sur une plateforme numérique dédiée qui permettra de déposer des contributions. Le Gouvernement propose quatre thèmes de débats : […] Les organisateurs de débats locaux ont la liberté de choisir tout autre thème qui leur semble pertinent. » (source : gouvernement.fr).
Cette application devra permettre à ses utilisateurs :

- D'organiser un *débat* autour d'un *thème* donné (1)
- De participer à plusieurs à un débat sous la forme d'une discussion textuelle synchrone (2)
- De marquer une partie du texte de la discussion comme une *contribution potentielle* (3)
- De consulter les *contributions définitives* existantes sur le même thème (4)
- De supprimer une contribution potentielle si elle existe déjà (5)
- De reformuler / modifier collaborativement une contribution potentielle (6)
- De valider une contribution potentielle pour qu'elle fasse partie des contributions définitives sur ce thème (7)

Quelques indications supplémentaires :
- Pour les spécifications (1) et (2) ci-dessus, vous réutiliserez votre application de chat (débat = chat, thème = salon)
- Pour les spécifications (3) à (6), vous l'étendrez en implémentant la notion de contribution.
- Vous utiliserez donc les technologies vues en cours : Spring Web MVC et REST côté serveur, Single-Page Application côté client et AJAX pour leur communication (le serveur ne sert que des pages HTML statiques ou des données en JSON).
- Vous ne vous occuperez pas de la création des comptes ni de l'authentification des utilisateurs et conserverez le mécanisme utilisé en TP pour récupérer le pseudo d'un utilisateur.
- Compte tenu de la charge attendue, il faut que votre application soit performante.

## Conception (barème : 6 points)

13. Vous aurez bien entendu besoin d'un contrôleur Spring pour gérer les requêtes relatives aux contributions définitives, et d'un second pour les contributions potentielles. Listez, en plus de ces contrôleurs, les éléments **supplémentaires** dont vous aurez besoin pour réaliser cette partir de l'application. Indiquez leurs types et à quoi ils servent.
    Beans : ContributionPotentielle (1 contrib par thème), ContributionsDefinitives (liste des contribs existantes)
    Vues (-> XML, JSON…) : ContributionPotentielle, ContributionsDefinitives
14. Vous aurez également besoin de modifier certains éléments existants de l'application (on ne tient pas compte des changements de noms pour chat et salon). Listez ces éléments et indiquez brièvement les modifications envisagées.
    Page HTML statique (SPA) de l'application : ajout de vues « contribution potentielle » (la contribution en cours de rédaction) et « contributions existantes » (les contributions définitives sur le thème)
    Scripts métier : marquage de texte (3), création / requête de notification de création d'une contribution potentielle (3), visualisation des contributions définitives (4), modification / actualisation / suppression de la contribution potentielle (5, 6, 7)
15. Décrivez les mécanismes que vous allez mettre en œuvre pour optimiser votre application côté client. On suppose que l'utilisateur est connecté et qu'un thème valide est choisi. La tâche principale considérée pour le chemin critique de rendu sera la visualisation de la contribution potentielle courante.
    App shell : déjà existant sur la SPA donnée en TP
    Priorisation du script de téléchargement de la contribution potentielle
    Mise en defer du script de modification de la contribution potentielle
    Mise en async des autres scripts AJAX (chat…)
    Lazy loading des contributions définitives
    Éventuellement : réduction de la page HTML et lazy loading des autres sections

## Programmation (barème : 4 points)

16. Écrivez le contrôleur Spring permettant de traiter des requêtes sur la contribution potentielle en cours de discussion sur un thème.
    Attendu : classe de contrôleur Spring annoté, avec les méthodes correspondant aux méthodes POST, GET, PUT et DELETE sur une URL du type /theme/{t}/contribution-potentielle.
    Ces méthodes doivent utiliser un bean représentant la contribution potentielle, et renvoyer les bons codes de réponse HTTP.

17. Écrivez la fonction JavaScript qui va permettre, une fois du texte sélectionné, de le proposer en tant que contribution potentielle. Elle l'enverra au serveur et en fonction de la réponse, affichera une erreur ou changera de route locale (`#contribution`).
    Remarques / indications :
    - une fois qu'une contribution a été proposée, il ne doit pas être possible d'en proposer une autre tant que la contribution courante n'a pas été soit validée comme contribution définitive, soit supprimée.
    - La méthode permettant d'accéder au texte sélectionné est `window.getSelection().toString()`
    - La fonction JS à appeler pour changer de vue est `show(hash)`

- Fonction JS sans paramètre
- Déclaration de la variable texte
- Requête AJAX en POST au contrôleur précédent (3)
- Callback -> en fonction du code de réponse :
    i. Si succès appel de la méthode show
    ii. Sinon, affichage d'une erreur (1 bis)