Master 1 informatique – Université Claude Bernard Lyon 1 – lundi 6 janvier 2020

Numéro de copie d'examen :

M1IF 03 – Conception d'applications Web – Examen

Durée : 1 heure 30 – Documents autorisés (4 pages max) – Ordinateurs, calculatrices, tablettes, téléphones portables… interdits

Questions de cours (barème : 16 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. Pourquoi un en-tête `Accept` peut-il contenir plusieurs types MIME ?

   Pour la négociation de contenus : le serveur peut renvoyer la réponse, par ordre de priorité, dans l'un des types demandés.

2. Pourquoi n'utilise-t-on pas la méthode GET pour les requêtes d'authentification ?

   Les données passent en clair dans l'URL.

3. Donnez deux exemples architecturaux dans lesquels un proxy permet de gagner en scalabilité.

   - redirection des requêtes vers des serveurs spécialisés (traitements dynamiques, pages statiques…)
   - load balancing

4. Pourquoi, lors de l'initialisation d'une connexion HTTPS, faut-il échanger 2 clés différentes entre le client et le serveur ?

   - clé publique (asymétrique) du serveur : pour chiffrer la clé symétrique spécifique au client
   - clé symétrique : pour chiffrer et déchiffrer (rapidement) les données échangées

5. Donnez deux moyens permettant de transmettre une variable d'une servlet vers une JSP.

   - en la plaçant dans le contexte applicatif
   - en la plaçant dans un attribut de requête

6. Pour chacun des deux moyens ci-dessus, expliquez comment récupérer cette variable dans une JSP.

   - <jsp:useBean scope="application" type="..." id="..."/>
   - <%= request.getAttribute(...) %> (toute autre syntaxe ou bean de scope request acceptés).

7. Citez une limitation dans la configuration des servlets et des filtres dans un descripteur de déploiement web.xml.

   Impossible de définir un mapping pour une forme d'URL complexe (avec une regexp par exemple).

8. En REST, pourquoi l'authentification par header HTTP est-elle préférable à l'authentification par cookie ?

   Le client peut choisir d'envoyer ou non un token dans un header HTTP, alors qu'un cookie est renvoyé automatiquement au

serveur.

9. Citez deux types de données qui transitent classiquement dans un JSON Web Token (JWT).

   - identifiant de l'utilisateur
   - variables de session

10. En quoi le principe « Hypermedia As The Engine Of Application State » (HATEOAS) de REST permet-il à un serveur d'exposer une Web API plutôt qu'une application Web ?

    Le serveur décrit le fonctionnement de son API avec une documentation et des contrôles hypermédias, et c'est le client qui choisit les actions à réaliser en fonction de la finalité d'une application donnée.

11. Dans le cadre d'une application Web en AJAX, donnez deux avantages à mettre en place une Web API RESTful côté serveur, plutôt que de réaliser uniquement des servlets/JSP répondant uniquement aux requêtes spécifiques à l'application.

    Scalabilité, testabilité, maintenabilité, réutilisabilité, évolutivité, possibilité de recourir à des outils externes (frameworks)…

12. Dans le cadre d'une application Web en AJAX, donnez deux inconvénients à mettre en place une Web API RESTful côté serveur, plutôt que de réaliser uniquement des servlets/JSP répondant uniquement aux requêtes spécifiques à l'application.

    Temps de développement initial, lourdeur pour une petite application, performances (pas forcément optimisé pour les requêtes spécifiques de l'application)…

13. Quel est le rôle minimal d'un moteur de templating dans une Single-Page Application (SPA) ?

    « Remplir » la vue courante avec les données du modèle.

14. Quel rôle peut aussi jouer un moteur de templating pour optimiser les performances d'une SPA ?

    Pré-compiler les templates, pré-calculer les vues correspondant aux ressources non affichées (chargées en async)…

15. En quoi le fait d'adopter une approche DevOps aide-t-il à optimiser la performance d'une application Web ?

    En maîtrisant toute la « stack » (infrastructure + code applicatif), on peut : identifier les problèmes de performance (goulots d'étranglement) -> infra, et les corriger (reconcevoir les ressources de manière adaptée à la charge) -> code.

16. Citez deux tâches qu'un framework côté serveur permet d'automatiser.

- mise en place d'un pattern MVC
- routage des requêtes vers le bon contrôleur
- configuration des moteurs de template
- …

Numéro de copie d'examen :

Programmation (barème : 8 points)

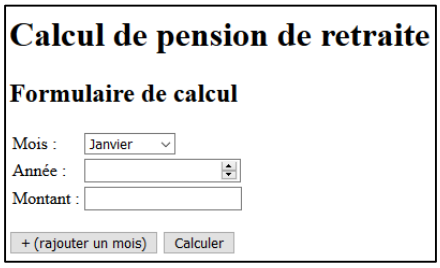
Dans cette partie, il n'est demandé que du code. Inutile d'écrire autre chose, ce ne sera pas corrigé.

Le Gouvernement vous a choisi.e pour concevoir et réaliser le simulateur de calcul des pensions de retraites permettant de savoir combien toucheraient les retraités après la nouveau réforme. L'API vous est imposée et est décrite avec Swagger comme illustré ci-dessous.

On ne considère ici que les requêtes formulées en JSON et les réponses renvoyées de la même façon. L'API et la SPA cliente sont sur le même serveur.

17. Écrivez la méthode de service d'un `HttpFilter` qui :
    - vérifie le type MIME puis parse en JSON le corps de la requête pour obtenir un objet de type `List<Remuneration>`,
    - vérifie que le nombre d'éléments de ce tableau correspond à l'API,
    - en fonction de ces vérifications, renvoie l'erreur appropriée ou ajoute cet objet aux attributs de la requête puis applique le pattern « chaîne de responsabilité ». (4 pts)

La partie cliente est à réaliser sous forme de SPA, comme indiqué en cours. Un formulaire simplifié avec son interface sont donnés ci-dessous.




```
<section id='formulaire' class='active'>
  <h2>Formulaire de calcul</h2>
  <div id="remunerations">
    <p id="remuneration0">
      Mois : <select id="mois0">
        <option value="1">Janvier</option>
        ...
        <option value="12">Décembre</option>
      </select>
      Année : <input type="number" id="annee0" min="1920" max="2020">
      Montant : <input type="text" id="montant0">
    </p>
  </div>
  <p>
    <button onclick="ajouter()">+ (rajouter un mois)</button>
    <button onclick="calculer()">Calculer</button>
  </p>
</section>
<section id='resultat' class='inactive'>
  <h2>Résultat du calcul</h2>
  <p>D'après nos calculs, vous devriez toucher <span id="pension">X</span> euros mensuels.</p>
</section>
```

- La fonction `ajouter()` rajoute un élément `p` à la `div` « `remunerations` », en incrémentant les id
- La fonction `calculer()` envoie la requête à l'API, place le résultat dans le `span` « `pension` » et navigue sur la vue « `résultat` ».
- La valeur de l'option sélectionnée dans l'élément `select` « `mois0` » peut être obtenue par : `$("#mois0 :selected").val()`

18. Écrivez la fonction `calculer()` en utilisant jQuery ; si le serveur renvoie une erreur, vous l'afficherez dans une fenêtre `alert()`. (4 pts)

Les réponses écrites sur la copie d'examen ou en-dessous de cette ligne ne seront pas corrigées.