Nom : **Prénom :**

Durée de l'épreuve : 1h30'

Documents autorisées : Aucun (TE sur feuille de papier uniquement). Les ordinateurs doivent être éteints. Travail individuel : pas d'accès à la messagerie ni à Internet.

A rendre : L'énoncé du TE avec votre nom. La (les) feuilles de réponses avec nom et prénom.

Partie 1 : Questions théoriques *20 pts*

Question 1 – Typologies d'architecture *4 pts*

- *Expliquez les différences entre une architecture de type « monolithe » et une architecture de type « distribuée » sous forme de schéma commenté et décrit*

Question 2 – JJE vs Spring *4 pts*

- *Expliquez les différences entre la plateforme JEE et le framework Spring*

Question 3 – MVC *4 pts*

- *Qu'est-ce que MVC ? Expliquez son fonctionnement sous forme de schéma.*

Question 4 – Architecture client-serveur *4 pts*

- *Qu'est-ce qu'une architecture client-serveur ?*
- *Quelles sont les 3 concepts clés de ce type d'architecture ?*

Question 5 – Spring bean *4 pts*

- *Quelle est le scope (visibilité) par défaut d'un « bean » Spring ?*
- *Citez 2 autres scope existant (autre que le scope par défaut) ?*

Partie 2 : Étude de cas *20 pts*

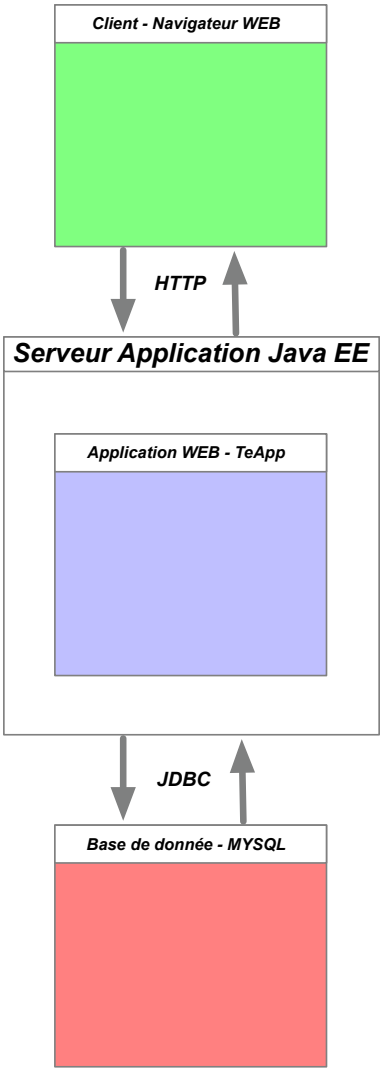
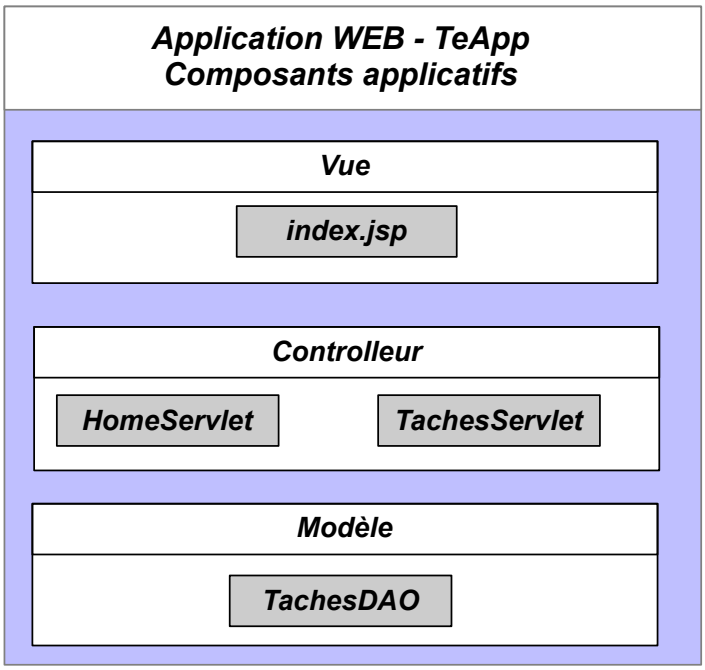
*Vous êtes engagés dans une entreprise qui a développé une application : **TeApp**. Vous êtes mandatés pour fournir une expertise de l'application et également pour implémenter de nouvelles fonctionnalités. Cette application souffre de certaines incohérences quant à sa conception, mais malgré tout elle remplit les fonctionnalités demandées.*

Initialement, l'application devait remplir les fonctionnalités suivantes :

- *Affichage d'un message de « bonjour » personnalisé à un utilisateur*
- *Lister les tâches à réaliser*

A noter que les 2 fonctionnalités ne sont pas liées. La liste des tâches n'est pas liée à un utilisateur.

*L'application répond sur le contexte (contextPath) suivant : **/TeApp** (http://localhost:8080/TeApp)*

Schéma logique de principe	***Schéma des composants applicatifs (classes et jsp)***
	

*Le code applicatif du composant **HomeServlet** est le suivant :*

```
@WebServlet(
     name="homeServlet",
     urlPatterns = ""
)
public class HomeServlet extends HttpServlet{

   @Override
   protected void doGet(HttpServletRequest req, HttpServletResponse resp) {

      if(req.getParameter("qui") != null) {
         req.setAttribute("utilisateur", req.getParameter("qui"));
      }

      try {
         req.getRequestDispatcher("index.jsp").forward(req, resp);
      } catch (ServletException | IOException e) {
         e.printStackTrace();
      }
   }
}
```

*Le code applicatif du composant **TachesServlet** est le suivant :*

```
@WebServlet(
     name="tachesServlet",
     urlPatterns = "/taches"
)
public class TachesServlet extends HttpServlet{

  TachesDAO tachesDAO = new TachesDAO();

   @Override
   protected void doGet(HttpServletRequest req, HttpServletResponse resp) {

      if(Boolean.valueOf(req.getParameter("showTaches"))) {
         req.setAttribute("taches", tachesDAO.getAllTaches());
      }

      try {
         req.getRequestDispatcher("index.jsp").forward(req, resp);
      } catch (ServletException | IOException e) {
         e.printStackTrace();
      }
   }
}
```

*Le code applicatif du composant **index.jsp** est le suivant :*

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ page import="java.util.List" %>

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>TeApp</title>
  </head>

  <body>
        <%
            String utilisateur = (String) request.getAttribute("utilisateur") ;
            if(utilisateur == null){
               utilisateur =  "inconnu") ;
            }
         %>
    <p>
      Bonjour <%= utilisateur %>!
    </p>

    <% if(request.getAttribute("taches") != null){%>

      <h1>Liste des tâches</h1>

    <% List<String> taches = (List)request.getAttribute("taches");

      for(String tache : taches){%>

      <p><%= tache %></p>

    <% }
    }%>
  </body>
</html>
```

*La classe **TachesDAO**, qui permet l'accès aux données comporte une seule méthode :*

```
public List<String> getAllTaches() {
    return Arrays.asList(new String[]{"tache 1","tache 2","tache 3","tache 4"});
}
```

*Par souci de simplification, une tâche **est modélisée au format String**, et la liste des tâches est **en mémoire**.*

Feuille de réponse

Nom : **Prénom :**

Partie 1 – Questions théoriques

Question 1 – Typologie d'architectures *4 pts*

- *Expliquez les différences entre une architecture de type « monolithe » et une architecture de type « distribuée » sous forme de schéma commenté et décrit*

Question 2 – JJE vs Spring *4 pts*

- *Expliquez les différences entre la plateforme JEE et le framework Spring*

Question 3 – MVC *4 pts*

- *Qu'est-ce que MVC ? Expliquez son fonctionnement sous forme de schéma*

Question 4 – Architecture *4 pts*

- *Qu'est-ce qu'une architecture client-serveur ?*

- *Quelles sont les 3 concepts clés de ce type d'architecture ?*

Question 5 – Spring bean *4 pts*

- *Quelle est le scope (visibilité) par défaut d'un « bean » Spring ?*

- *Citez 2 autres scopes existants (autre que le scope par défaut) et expliquez leurs niveaux de visibilités?*

Partie 2 - Étude de cas

Question 1 – Analyse du fonctionnement *5 pts*

Pour chaque url ci-dessous, écrivez le résultat de l'affichage de la page dans le navigateur. Le format (html) n'est pas déterminant, au minimum les valeurs affichées.

URL	*Résultat*
/TeApp	
/TeApp?utilisateur=Mickey	
/TeApp?qui=Mickey	
/TeApp/taches	
/TeApp/taches?showTaches=true	

Question 2 – Résolution de problèmes *10 pts*

L'architecte du projet vous mandate pour traiter le problème suivant :

- *Si on appelle cette url : **http://localhost:8080/TeApp/taches?showTaches=true&qui=moi** il n'y pas moyen d'afficher en même temps un message de bonjour à un utilisateur et la liste des tâches. L'utilisateur affiché est toujours « inconnu ».*

1. *Expliquez pourquoi cette fonctionnalité ne se pas comporte pas comme prévu (pourquoi cette url n'affiche pas **et** la liste des tâches **et** le message personnalisé pour l'utilisateur) ?*

2. *Schématiser les interactions entre les composants dans le cadre de cette requête. Utiliser des cercles pour les composants, un rectangle pour le client et des flèches pour les interactions. Expliquer le but des interaction et le protocole utilisé.*

3. *Proposez une démarche afin de régler ce problème et de pouvoir afficher le message personnalisé et la liste des tâches avec une seule url. Décrivez les actions que vous effectuez pour chaque composant impacté. Si de nouveaux composants doivent être mis en place, décrivez leurs fonctionnements.*

Composant impactés	***Description de la tâche***

Question 3 – Nouvelles fonctionnalités *5 pts*

En partant de la situation initiale *l'architecte trouve qu'il serait bien d'ajouter une fonctionnalité permettant de créer des tâches via l'application (jusqu'à maintenant elles sont créées directement en base de données !!!), il vous demande de lister les tâches à effectuer et les composants impactés :*

Composant impactés	***Description de la tâche***