**Kilian CHOLLET EFREI M1APP**

**Amaury KILIAN Question Projet JEE**

**Lucas MASSON**

1. **Comment définiriez-vous brièvement Java EE ?**

Java Enterprise Edition est un Framework de la plateforme Java conçu pour le développement Web notamment. Elle fournit plusieurs API comme JSP, EJB, Servlet ect.

1. **L'offre globale de Java EE est-elle identique à celle de PHP ? Sinon donnez 5 différences entre Java EE et PHP.**

Non l’offre de Java EE est différente, et plus complète que celle de PHP.

Ils n’ont pas exactement le même paradigme :

* JEE : Paradigme objet uniquement, structuré, impératif.
* PHP : multiparadigme => Impératif, objet, fonctionnel, procédural, réflexif. Langage interprété

JEE est basé sur un typage fort et statique alors que PHP a un typage dynamique et faible.

Pour exécuter du PHP il faut un interpréteur PHP alors que pour JEE il faut une JVM.

PHP est uniquement dédié au développement Web alors que JEE peut être utilisé pour tout type d’application.

PHP n’a pas besoin de serveur d’application qu’autre un serveur WEB à l’inverse de JEE.

JEE a son packaging propre pour le déploiement les archives WAR, alors PHP n’en possède pas.

1. **Si vous créez une classe et que vous la nommez "MaServlet.java" cela en fait-il automatiquement une servlet ? Pourquoi ?**

Non, il ne suffit pas d’appelé une classe « Servlet » pour qu’elle soit une servlet. Pour qu’une classe soit une servlet il faut que la classe implémente l’interface « Servlet ».

1. **Au démarrage d’une application, le conteneur Web initialise une seule instance d’une servlet. Cela implique-t-il que la servlet ne puisse exécuter qu’une tâche à la fois ?**

Non, car bien qu’il existe qu’une instance de servlet, cette instance est partagée par tous les threads. Elle peut donc s’exécuter de manière simultanée et donc multi-tâches.

1. **Le fait d’utiliser EL/JSTL dans les JSP rend-t-il votre application automatiquement compatible avec le patron MVC ?**

Non, l’utilisation d’EL/JSTL n’implique pas un patron MVC. En effet il est possible de faire des requêtes SQL en JTSL donc dans des views, ce qui brise le patron MVC.

1. **Nous avons vu que les JSP se situent dans la "Vue" du patron MVC. Or qui dit "Vue" dit "Affichage client", vous confirmez ? Or dans JSP, "S"= "Server". N'y a-t-il pas là une incohérence ? Expliquez**

La « Vue » et l’« Affichage client » ne sont pas la même chose, l’Affichage client correspond à ce que l’utilisateur voit alors que la Vue correspond au code qui sert à générer ce que voit l’utilisateur. Il est donc cohérent d’avoir le S de JSP qui signifie « Server », les JSP correspondent à la Vue car il s’agit de code permettant d’obtenir ce que l’utilisateur verra.

1. **En Java EE, la "Vue" doit-elle obligatoirement être gérée dans des JSP ? Donnez des exemples**

Non la « Vue » n’est pas forcément gérée dans des JSP, elle peut être faite dans un controller (génération de texte HTML / CSS) mais cette pratique casse le principe du modèle MVC.

1. **Quel est l’intérêt d’utiliser, dans les projets d'entreprise, un fichier de type ".properties" pour stocker les informations de connexion à la base?**

Cela permet de centraliser les données de configuration de l’application et de pouvoir les modifier sans toucher au code (= découplage).

1. **Quelles sont les deux syntaxes pour lire le contenu d’un fichier .properties? Y-a-t-il une différence entre ces deux syntaxes ?**

Pour utiliser un fichier properties on peut utiliser un FileInputStream ou alors passer par le classLoader. Pour le classLoader le fichier doit être dans le dossier racine du classpath.

1. **request.getParameter et request.getAttribute sont-elles deux instructions identiques? Expliquez**

Non ces instructions sont différentes, les paramètres sont des variables mises « automatiquement » par le système (exemple : en html un champ input dans un formulaire engrange un paramètre) alors que les attributs sont uniquement placés par le développeur via un setAttribute et donc il est seul « maitre » de cette liste de variable.

1. **Présentez brièvement l'outil Maven. Et selon vous quel est l'intérêt d'utiliser un tel outil ?**

Maven est un outil de gestion d’automatisation de production et de dépendances, qui permet entre autres d’automatiser des actions comme le pré-processing, la compilation, l’édition des liens … Cela permet de garantir le bon ordre de fabrication de l’application. Par ailleurs, il permet une gestion efficace des dépendances.

1. **Qu'entend-on par "objet implicite" ? Donnez 5 exemples. Comment les utilise-t-on ?**

Les objets implicites sont des objets qui sont déclarés et instanciés automatiquement avec la plateforme JEE, on note parmi ces objets : out, param, request, response, session et d’autres.

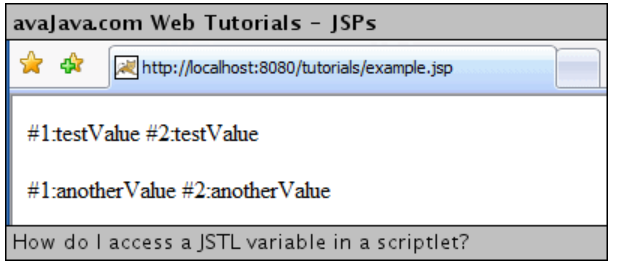
1. **Utilisez-vous les mêmes objets implicites lorsque vous codez en scriptlets ou en EL ? Expliquez avec un exemple.**

Oui, nous utilisons les mêmes objets implicites en scriptlets ou en EL car ces deux API sont deux façons différentes de modifier des fichiers HTML « dynamiquement » et se base sur ce que fournis le serveur d’application (requête, réponse, session ect...) qui sont unique pour un instant donné et repris depuis différentes formes.

1. **Les trois familles de syntaxe, scriptlets, balises action, EL et JSTL peuvent-elles coexister dans un même fichier JSP ? Argumentez avec un exemple**

Oui elles peuvent parfaitement co-exister car elles n’ont pas d’impact entre elles et donc cela évite les « conflits ».

Exemple :

1. **Les servlets que nous avons créées gèrent les échanges via le protocole HTTP. Est-ce donc impossible d'utiliser une servlet dans le cadre d'un autre protocole ? Par exemple FTP, SMTP... ?**

Il est tout à fait possible d’utiliser des servlets utilisant d’autres protocoles tels que FTP ou SMTP. On utilisera donc pas l’objet HTTPServlet mais des objets FTPServlet ou SMTPServlet.

1. **Existe-t-il un rapport entre l'objet scope session et la session de l'utilisateur ? Ou est-ce une coïncidence sémantique ?**

Oui, l’objet scope session et la session de l’utilisation se relié. Les données de la session de l’utilisateur sont stockées dans l’objet scope session.

1. **En quoi consiste l'injection de dépendance ou CDI en anglais (Context Dependency Injection) ?**

L’injection de dépendance consiste à créer dynamiquement un objet contenant les dépendances nécessaires à l’exécution. L’avantage est de ne plus résoudre les dépendances à la compilation mais uniquement à l’exécution.

1. **Nous avons vu que la configuration d'une application Java EE de type Web se fait dans le fichier web.xml. Indiquez une autre façon de configurer l'application.**

On peut également configurer l’application via des « Annotations » directement dans le code.

1. **JavaBean = Enterprise Java Bean. Êtes-vous d'accord ? Pourquoi ?**

Non un JavaBean et un Entreprise Java Bean sont différentes. Un JavaBean est une façon de modéliser une donnée par un convention (attributs privés + getter et setter et constructeur vide …) alors qu’un Entreprise Java Bean est un composant de Java EE qui reprend le principe de JavaBean en rajoutant des fonctionnalités comme la gestion de session, la sécurité, la transaction ect... Il en existe trois types : Session Bean, Entity Bean et Message Driven Beans.

1. **Existe-t-il un risque lié à une instruction du type "SELECT \* from CLIENTS where ID =" + idClient idClient étant une variable ? Si oui, précisez la nature de ce risque et indiquez comment il a été géré dans JDBC et dans JPA**

Oui c’est risqué, car nous utilisons une variable directement dans notre requête SQL et dela nous expose donc à une injection SQL malicieuse : cela consiste à exploiter la faille pour récupérer des informations stockées en base de données (ou pire, modifier des données).

1. **Dans la requête JPQL "SELECT a from Amis a”, que représente Amis ? Expliquez**

Amis représente une Classe métier (entité) de notre application. Les requêtes JPQL permettent de s’abstraire de la base de données.

1. **Vous avez réalisé une migration de JDBC vers JPA. Alors pourriez-vous indiquer avec précision quels sont les changements dans votre application ? Quels fichiers ont été rajoutés/supprimés/modifiés (classes, drivers, fichiers JSP, fichiers xml, autres fichiers, ...) ? La servlet joue-t-elle encore le rôle de contrôleur ou est-ce l'EntityManager qui joue ce rôle ?**

Ayant migré de JDBC vers JPA, nous n’avons plus de code SQL présent dans le code ou les paramètres de l’application. Nous interagissons seulement avec les entités (java), et c’est JPA qui se charge de transactions avec la base de données. Cependant, c’est toujours la servlet qui joue le rôle de contrôleur.

1. **Qu'entend-on par déployer une application ? Comment se fait le déploiement en Java EE ?**

Le déploiement d’une application consiste à installer une application web sur un serveur d’application et de lancer le serveur d’application pour faire fonctionner l’application. Cela se fait via des fichiers de type WAR ou similaire qui sont des fichiers de ressource d’application web. Il contient des fichiers comme web.xml, un répertoire WEB-INF et autres, il contient l’ensemble des pages webs et des classes compiler. Le serveur web va se baser sur l’archive WAR pour lancer l’application web.

1. **Est-il possible d'accéder à l'application que vous développez sur votre PC en utilisant son adresse IP ? Expliquez**

Oui c’est possible, si on déploie en local, il est possible de se connecter à l’application par l’adresse IP de l’hôte. Localhost correspond alors à l’adresse IP interne du PC.

1. **Quelle est l'API de Java EE permettant de gérer les Web services SOAP ? Idem pour les Web services RESTful ? Pourquoi deux API pour des Web services ?**

L’API de Java EE JAX-WS permet de gérer les web services SOAP basé sur des fichiers XML.

Alors que pour des web services RESTful l’API utilisé est JAX-RS qui utilise sur des fichiers JSON.

Ils existent deux API différentes car en fonction de l’architecture de web service choisie un type de fichier de transfert est préconisé ; JSON pour REST et XML pour SOAP. Ils ont tous deux des particularités ce qui a créé le besoin d’avoir deux API différentes.