EISTI 2008-2009 – Examen

Java EE

1h30 – Aucun document autorisé

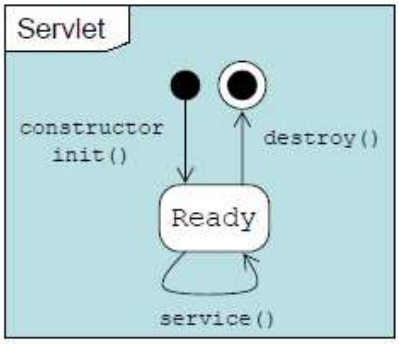
NOM : **PLACE :**

PRENOM : **GROUPE TD :**

Q1 : cycle de vie d'un servlet (2 points)

Donnez le diagramme d'état représentant le cycle de vie d'un servlet dans son conteneur.

1. Chargement de la classe
2. Instanciation du servlet
   - constructeur par défaut
3. Appel de init()
4. Appel(s) de service()
   - 1 thread par requête
5. Appel de destroy()



Q2 : redirection vs. forward (4 points)

Un servlet peut « appeler » une JSP :

- avec une redirection :

```
response.sendRedirect("myJSP.jsp");
```

- ou avec un forward :

```
RequestDispatcher
    jsp=resquest.getRequestDispatcher("myJSP.jsp");
jsp.forward(request,response);
```

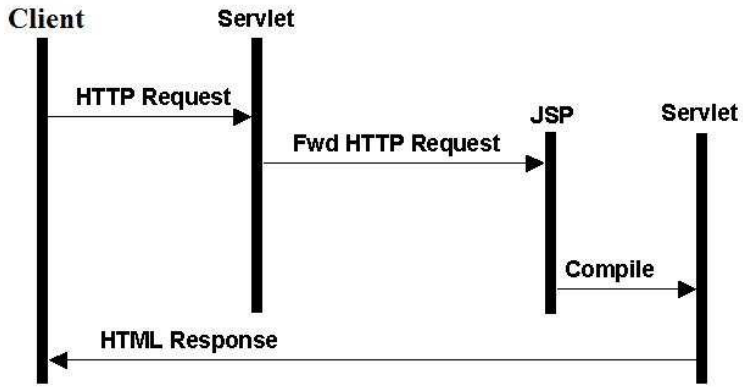
Expliquez puis montrez sur un exemple concret la différence entre l'utilisation de ces deux mécanismes (vous pourrez vous appuyer sur des diagrammes de séquence faisant intervenir le client, le servlet et la JSP, pour étayer votre réponse).

Forward :

C'est une action effectuée de manière interne à la servlet dans son conteneur.

Le navigateur du client n'est jamais mis au courant des changements effectués. Ainsi l'URL du navigateur ne change pas.

Si le navigateur client actualise la page, la requête initiale sera répétée, avec l'URL originale.



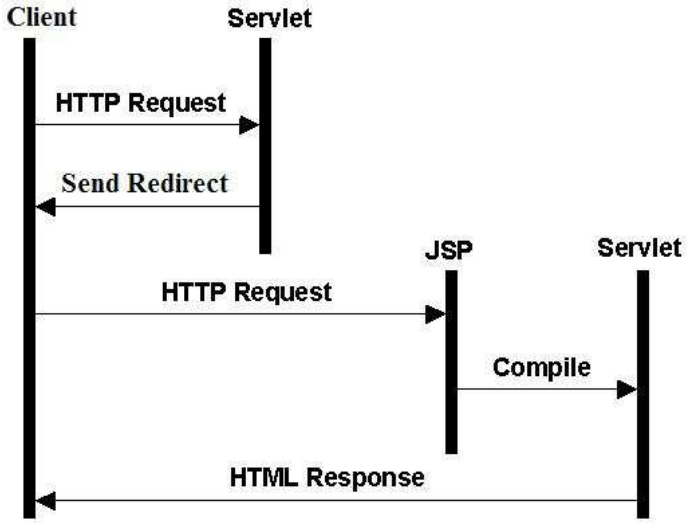
Redirect :

C'est un processus en 2 étapes. Après avoir reçu la première requête, le Servlet envoi au navigateur client une demande de redirection vers une autre URL qu'il effectue.

Si le navigateur client actualise la page, la requête initiale ne sera pas répétée. Au contraire c'est la nouvelle URL qui sera rechargée.

Les Objets/Beans placés dans le request Scope initial ne seront plus disponible dans la seconde requête.

La redirection est toujours plus lente qu'un forward, car il nécessite une seconde requête du navigateur client.



Exemple :

Les Forward doivent être utilisés si l'opération peut être répétée lorsque l'on recharge la page. Sinon il faut utiliser un redirect. Par exemple, si l'opération modifie une base de donnée, alors il faut utiliser un redirect.

Q3 : QCM (8 points)

Entourez la ou les bonnes réponses (seulement la lettre).

1) Quelles sont les méthodes utilisées par un servlet pour gérer les données envoyées par un client via un formulaire HTML ?

A. HttpServlet.doHead()
B. HttpServlet.doPost()
C. HttpServlet.doForm()
D. ServletRequest.doGet()
E. ServletRequest.doPost()
F. ServletRequest.doForm()

2) Comment un développeur gère-t-il la méthode service() de son servlet, lorsque ce dernier étend HttpServlet ?

A. Il redéfinit la méthode service().
B. Il redéfinit une méthode doXXX() (par ex. goGet ou doPost).
C. Il appelle la méthode service() depuis une méthode doXXX() (par ex. doGet() ou doPost()).
D. Il appelle la méthode service() depuis la méthode init().
E. Il n'a rien à faire…

3) Sur quels types d'objets peut-on utiliser les méthodes getAttribute() et setAttribute() ?

A. HttpSession
B. ServletRequest
C. ServletResponse
D. ServletContext
E. ServletConfig
F. SessionConfig

4) Soit le code suivant :

```
<html>
 <body>
Le chiffre à deviner est : <%= Math.random();>
 </body>
</html>
```

Qu'affiche cette JSP à la suite de « Le chiffre à deviner est : » ?

A. Un nombre aléatoire.
B. <%= Math.random() ;>
C. out.println("Math.random() ;") ;
D. Elle n'affiche rien… (il manque le « % » dans « %> » + pas de « ; » dans ce tag).

5) Parmi les balises suivantes quelles sont celles que l'on peut utiliser dans une JSP pour afficher la valeur d'une expression Java sur la sortie ?

A. <@ >
B. <% > (<% out.println(…) %>)
C. <%= >
D. <%! >
E. <%$ >

Note : Il manque les % sur la balise fermante.

6) Soit le code de la JSP test.jsp suivant :

```
1 <html>
2    <head><title>A Comment Test</title></head>
3    <body>
4       <h2>A Test of Comments</h2>
5       <!-- This is Html Hidden Comment  -->
6       <%-- This is JSP Hidden Comment --%>
7    </body>
8 </html>
```

A l'exécution de test.jsp, devinez quel sera la sortie correspondante ?

A. La ligne 5 sera insérée dans la réponse mais pas la ligne 6.
B. Les lignes 5 et 6 seront insérées dans la réponse.
C. Les lignes 5 et 6 ne seront pas insérées dans la réponse.
D. La ligne 6 sera insérée dans la réponse mais pas la ligne 5.

7) Quelle(s) sont(est) les EL valables qui permettent de retourner la property « nom » du bean Etudiant ayant pour scope la session :

A. ${Etudiant.nom}
B. ${Etudiant.getNom()}
C. ${sessionScope.Etudiant["nom"]}
D. ${session["Etudiant"].nom}

8) Qu'affiche le code suivant ?

```
<c:forEach step="2" begin="3" end="8" varStatus="status">
<c:if test="${status.first}">[</c:if>
${status.index},
<c:if test="${status.last}">]</c:if>
</c:forEach>
```

A. 3,5,7,8,
B. [3,5,7,]
C. [3,5,7,8,]
D. [3,5,7]
E. 2,3,4,5,6,7,8

Q4 : MVC (6 points)

On considère l'application MVC que nous avons étudiée en cours : AREL V 6.0. Cette application permet à l'utilisateur de sélectionner une promotion pour afficher la liste des étudiants correspondante. Voici les fichiers sources de cette application :

index.html :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>AREL V6.0</title>
</head>
<body>
<h1 align="center">AREL : L'école virtuelle de l'EISTI</h1>
<form method="GET" action="select-promo">Sélectionner la promo à
afficher : <select name="promo" size="1">
      <option>ing1</option>
      <option>ing2</option>
</select><input type="SUBMIT" /></form>
</body>
</html>
```

arel/SelectPromo.java :

```
package arel;

import java.io.IOException;
import java.util.List;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class SelectPromo extends javax.servlet.http.HttpServlet
                         implements javax.servlet.Servlet {

  protected void doGet(HttpServletRequest request,
                       HttpServletResponse response)
                throws ServletException, IOException {

      String  promoName = request.getParameter("promo");
      Promo promo = new Promo();
      List<String> result = promo.getPromo(promoName);
      request.setAttribute("promo", result);
      RequestDispatcher view = request.getRequestDispatcher("result.jsp");
      view.forward(request, response);
  }
}
```

arel/Promo.java :

```
package arel;

import java.util.ArrayList;
import java.util.List;

public class Promo {

      public List<String> getPromo(String promo) {
            List<String> promoList = new ArrayList<String>();
            if (promo.equals("ing1")) {
                  promoList.add("Donald Duck");
                  promoList.add("Minnie Mouse");
                  promoList.add("Pluto");
                  //...
            } else if (promo.equals("ing2")) {
                  promoList.add("Mickey Mouse");
                  promoList.add("Daisy Duck");
                  promoList.add("Goofy");
                  //...
            } else {
                  return null;
            }

            return promoList;
      }
}
```

result.jsp :

```
<%@ page import="java.util.*"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Result</title>
</head>
<body>
<%
      List<String> promoList = (List<String>)request.getAttribute("promo");
      Iterator it = promoList.iterator();
      while(it.hasNext()) {
            out.print("<br />" + it.next());
      }
%>
</body>
</html>
```

1) Complétez le descripteur de déploiement suivant en fonction des liens et autres nommages exprimés dans les fichiers sources donnés ci-dessus.

web.xml :

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                        http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
         id="WebApp_ID" version="2.5">
  <display-name>MVC</display-name>
  <servlet>
    <servlet-name> X </servlet-name>
    <servlet-class> arel.SelectPromo </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name> X </servlet-name>
    <url-pattern> /select-promo </url-pattern>
  </servlet-mapping>
</web-app>
```

Avec X quelconque, mais identique pour les 2 <servlet-name />

On désire modifier l'application pour afficher la liste résultat dans la même page que celle du formulaire initial. L'option choisie est de remplacer le formulaire HTML index.html par une JSP index.jsp servant à la fois de formulaire de départ et de vue dans l'application MVC pour afficher le résultat en dessous du formulaire.

index.jsp :

```
<%@ page import="java.util.*" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>AREL V6.0</title>
</head>
<body>
<h1 align="center">AREL : L'école virtuelle de l'EISTI</h1>
<form method="GET" action="select-promo">Sélectionner la promo à
afficher : <select name="promo" size="1">
      <option>ing1</option>
      <option>ing2</option>
</select><input type="SUBMIT" /></form>
<h2>Liste des étudiants :</h2>
<%
      List<String> promoList = (List<String>)request.getAttribute("promo");
      Iterator it = promoList.iterator();
      while(it.hasNext()) {
            out.print("<br />" + it.next());
      }
%>
</body>
</html>
```

2) Que faut-il modifier pour que index.jsp devienne la vue dans notre application MVC ? (Donnez le(s) fichier(s) et les nouvelles instructions)

arel/SelectPromo.java

RequestDispatcher view = request.getRequestDispatcher("index.jsp") ;

3) Lorsque l'on tente d'accéder à la JSP index.jsp, une erreur survient. Trouvez cette erreur et corrigez la.

Au premier lancement d' index.jsp, l'attribut promo n'existe pas dans le requestScope. Il faut donc tester s'il n'est pas null.
2 possibilités pour corriger cette erreur :

Ajouter un test de nullité :

```
<%
	List<String> promoList = (List<String>)request.getAttribute("promo");
		if(promoList!=null)
		{
			Iterator it = promoList.iterator();
			while(it.hasNext())
			{
				out.print("<br />" + it.next());
			}
		}
%>
```

Ou bien avec les JSTL :
Ajouter dans les déclarations de la JSP :

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

Puis modifier le scriptlet <% … %> en :

```
<c:forEach var="X" items="${promo}">
<br />${X}
</c:forEach>
```

Avec X quelconque.
L'EL ${promo} peut aussi s'exprimer ${requestScope.promo} ou bien ${requestScope["promo"]}