J2EE

Collaboration servlets/JSP

Collaboration servlets/JSP

Partage du contrôle

Partage d'informations

J2EE: Collaboration servlets/JSP
Objectifs

Des servlets/JSP qui s'exécutent dans le même serveur peuvent communiquer les unes avec les autres

Elles peuvent partager de l'information :

- un état (état des ventes, des achats, des clients, ...)
- une ressource (connexion BD, un fichier statique, ...)

Elles peuvent partager le contrôle d'une même requête :

- la requête reçue par une servlet peut être traitée en partie par une autre servlet/JSP

Partage du contrôle

- Distributeur de requête
- Partage du contrôle par délégation de servlet
- Partage de contrôle par délégation de JSP
- Partage du contrôle par inclusion

Partage du contrôle
Partage du contrôle d'une requête

2 possibilités :

- Délégation de la requête à une autre servlet

  Une servlet reçoit une requête et laisse à une autre servlet (jsp) la responsabilité de la traiter (en partie ou totalement).
  Dans ce cas le contrôle est passé à la seconde servlet.

- Inclusion de contenu

  Une servlet inclut dans sa propre réponse un contenu généré dynamiquement par une autre servlet.
  La réponse peut être ainsi construite à partir d'un ensemble de contenus générés par divers composants web.
  Dans ce cas la première servlet conserve le contrôle.

Partage du contrôle
Transmission du contrôle par délégation

La délégation est obtenue par un distributeur de requête, instance de la classe `javax.servlet.RequestDispatcher`.

La servlet obtient un distributeur pour la requête vers un composant de l'application (servlet, jsp, fichier statique...) en indiquant son URI par la méthode :

```
public RequestDispatcher
ServletRequest.getRequestDispatcher(String uri)
```

L'URI peut être :
- relatif, on reste dans le contexte de la servlet courante
- absolu, il est interprété relativement à la racine du contexte

Le chemin peut être complété par une chaîne d'interrogation :

```
/reunion/Inscription?nom=albert
```

Partage du contrôle
Distribuer un renvoi

La méthode `forward()` passe la requête complète au composant cible.

```
req.setAttribute("numero","21515");

RequestDispatcher distibuteur
=req.getRequestDispatcher(ServletReponse);

distributeur.forward(req,res);
```

Le contrôle est passé à la servlet `ServletReponse` du même contexte qui peut de ce fait accéder à l'attribut `numero`

```
String num = (String)req.getAttribute("numero");
```

Partage du contrôle
Exemple 1 (1/3)

```
<servlet>
<servlet-name>Annuaire</servlet-name>
<servlet-class>lesServlets.Annuaire</servlet-class>
  <init-param>
      <param-name>urlAnnuaire</param-name>
      <param-value>/index.jsp</param-value>
</init-param>
<init-param>
<param-name>urlListePersonnes</param-name>
      <param-value>/vue-personnes.jsp</param-value>
</init-param>
<init-param>
<param-name>urlListeInvites</param-name>
      <param-value>/vue-invites.jsp</param-value>
</init-param>
</servlet>
```

Partage du contrôle
Exemple 1 (2/3)

```
public void init(ServletConfig config) throws ServletException {

urlListePersonnes =
config.getInitParameter("urlListePersonnes");

urlListeInvites =
config.getInitParameter("urlListeInvites");

urlAnnuaire =
config.getInitParameter("urlAnnuaire");
}
```

Partage du contrôle
Exemple 1 (3/3)

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

int choix = Integer.parseInt((String)request.getParameter("choix"));
RequestDispatcher dispatcher;
switch(choix){
    // choix = 1 => affichage de la liste des personnes
    case 1 :
         dispatcher = request.getRequestDispatcher(urlListePersonnes);
         dispatcher.forward(request,response);
    break;
    // choix = 2 => affichage de la liste des invités
    case 2 :
         dispatcher = request.getRequestDispatcher(urlListeInvites);
         dispatcher.forward(request,response);
    break;
    default:break;
    }
}
```

Partage du contrôle
Transmission du contrôle par délégation de JSP (1/3)

Une JSP peut déléguer le traitement d'une requête à une autre JSP

`<jsp:forward> ... </jsp:forward>`

principale.jsp:

```
<HTML> <BODY>
<H1>JSP principale</H1>
<jsp:forward page="déléguée.jsp" />
Ignoré !!
</BODY> </HTML>
```

déléguée.jsp

```
<HTML> <BODY>
<H1>JSP déléguée</H1>
<P> <%=(int)(Math.random()*5)%> </P>
</BODY> </HTML>
```

Partage du contrôle
Transmission du contrôle par délégation de JSP (2/3)

Une JSP peut transmettre des paramètres aux déléguées :

```
<jsp:param name="..." value="..." />
```

principale.jsp:

```
<HTML> <BODY>
<H1>JSP principale</H1>
<jsp:forward page="déléguée.jsp">
  <jsp:param name="nom" value="Samia" />
  <jsp:param name="age" value="34" />
</jsp:forward>
</BODY> </HTML>
```

déléguée.jsp

```
<HTML> <BODY>
<H1>JSP déléguée</H1>
Nom : <%= request.getParameter("nom") %>
Age : <%= request.getParameter("age") %>
</BODY> </HTML>
```

Partage du contrôle
Transmission du contrôle par délégation de JSP (3/3)

Nom : Samia
Age : 34

Partage du contrôle
Transmission du contrôle par inclusion

```
<jsp:include page="bonjour.jsp"/>
```

Le contenu retourné dépendra des paramètres de la requête

- L'inclusion a lieu au moment de la requête et non au moment de la création de la servlet.
- Le serveur exécute la ressource à inclure et inclut sa sortie au contenu envoyé au client
- Cela permet d'inclure à la fois du contenu statique et du contenu dynamique dans la page JSP.
- Elle n'est pas nécessairement une page HTML (les balises `<html>` peuvent être absentes)

Possibilité de transmettre des informations lors de l'inclusion:

```
<jsp:include page="page.jsp" >
<jsp:param name="paramParDefaut" value="nouvelle" />
</jsp:include>
```

Partage du contrôle
Transmission du contrôle par inclusion

```
<html><head><title>Page Principale</title></head>
<body>
<jsp:forward page="delegue.jsp">
  <jsp:param name="nom" value="Samia" />
  <jsp:param name="age" value="34" />
</jsp:forward>
```

principale.jsp

```
<H1>JSP déléguée</H1>
    <jsp:include page="/page.jsp"  >
      <jsp:param name="param" value="new " />
    </jsp:include>
Nom : <%= request.getParameter("nom") %><br>
Age : <%= request.getParameter("age") %>
</body>
</html>
```

delegue.jsp

page.jsp

```
<h1>Hello  : <%= request.getParameter("param") %> World!</h1>
```

Partage du contrôle
Transmission du contrôle par inclusion

Hello : new World!
Nom : Samia
Age : 34

Partage du contrôle
Inclure un contenu dans une réponse

La méthode `include()` :

- inclut un contenu à la réponse courante
- la servlet appelante garde le contrôle de la réponse

```
// on passe une chaîne à la réponse
RequestDispatcher distributeur =
req.getRequestDispatcher(ServletReponse?numero=21515);
distributeur.include(req,res);

// on passe un attribut (donc un objet à part entière)
RequestDispatcher distributeur =
req.getRequestDispatcher(ServletReponse);
req.setAttribute("objet",new Object());
distributeur.include(req,res);
```

Partage du contrôle
Exemple 2 (1/4)

- Le projet include contient 2 servlets : `Principale` et `Secondaire`

- Une requête émise à `Principale` construit une réponse sous la forme d'un texte formé à partir de `Principale` et d'un contenu inclus fourni par `Secondaire`

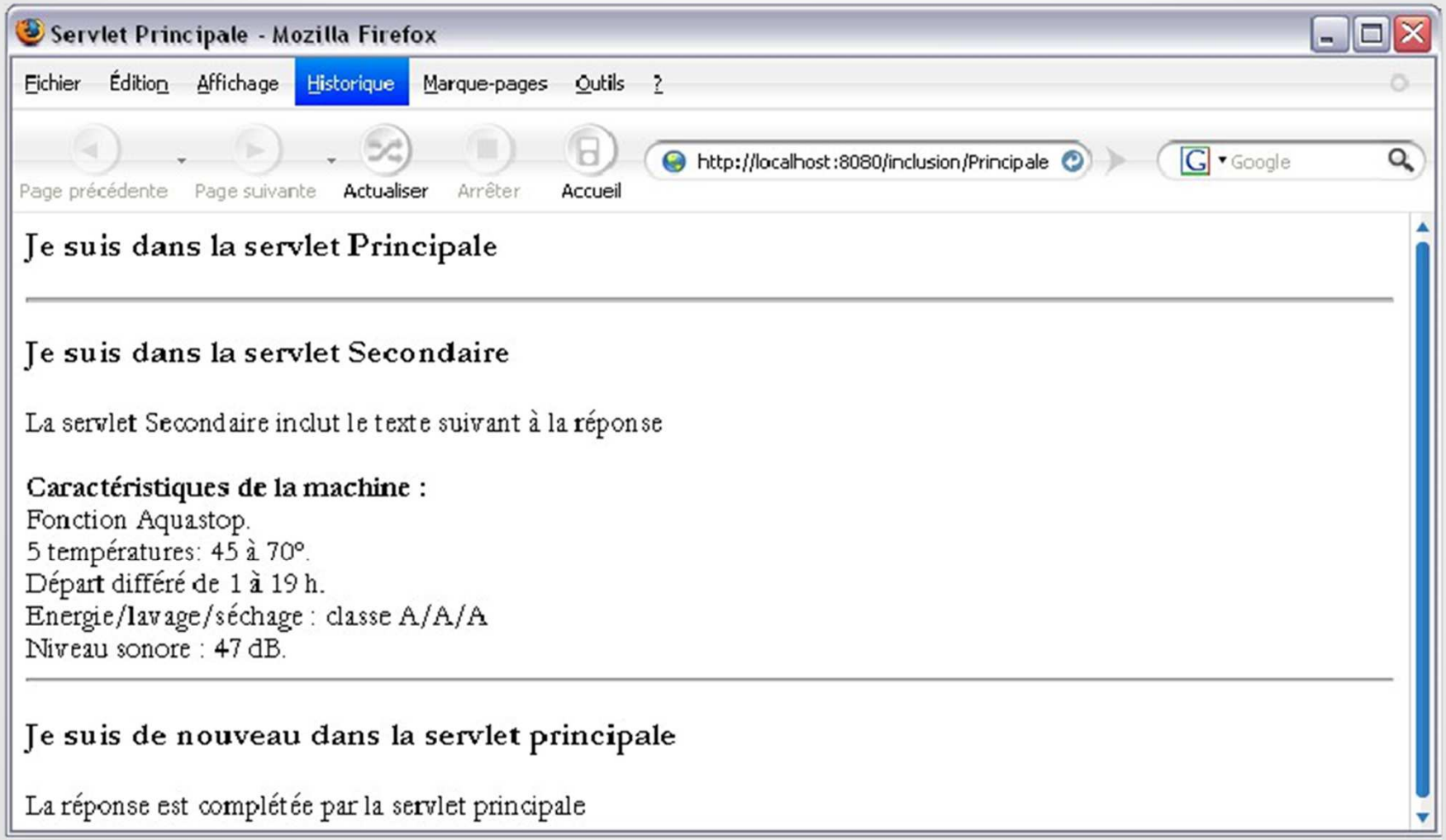
Partage du contrôle
Exemple 2 (2/4)

```
public class Principale extends HttpServlet {

protected void doGet(HttpServletRequest req, HttpServletResponse
res)throws ServletException, IOException {
    res.setContentType("text/html;charset=UTF-8");
    PrintWriter out = res.getWriter();
    out.println("<html><head>");
    out.println("<title>Servlet Principale</title></head>");
    out.println("<body>");
    out.println("<h3>Je suis dans la servlet Principale</h3>");
    out.println("<hr/>");
    RequestDispatcher dist = req.getRequestDispatcher("Secondaire");
    dist.include(req,res);
    out.println("<hr/>");
    out.println("<h3>Je suis de nouveau dans la servlet principale</h3>");
    out.println("La réponse est complétée par la servlet principale");
    out.println("</body></html>");
    out.close();
    }
}
```

Partage du contrôle
Exemple 2 (3/4)

```
public class Secondaire extends HttpServlet {

protected void doGet(HttpServletRequest req, HttpServletResponse
res) throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    out.println("<h3>Je suis dans la servlet Secondaire</h3>");
    out.println("La servlet Secondaire inclut le texte suivant à
la réponse<br/><br/>");
    out.println("<b>Caractéristiques de la machine :</b><br/>");
    out.println("Fonction Aquastop.<br/>");
    out.println("5 températures: 45 à 70°.<br/>");
    out.println("Départ différé de 1 à 19 h.<br/>");
    out.println("Energie/lavage/séchage : classe A/A/A<br/>");
    out.println("Niveau sonore : 47 dB.");
    }
}
```

Partage du contrôle
Exemple 2 (4/4)



Partage du contrôle
Remarques sur le partage du contrôle

Le partage du contrôle par des balises actions ne permettent pas le transfert d'attributs objet autres que des chaînes de caractères.

Il faut alors utiliser un objet `RequestDispatcher` et l'objet implicite `request`.

Rappel :

```
<% Date date = new Date();
RequestDispatcher dispatch=
    request.getRequestDispatcher("/suite.jsp");
request.setAttribute("Date",date);
dispatch.include(request,response);
%>
```

Transmission d'un paramètre servlet -> jsp via la requête (1/3)

```
protected void doGet(HttpServletRequest req, HttpServletResponse
res) throws ServletException, IOException {
    res.setContentType("text/html;charset=UTF-8");
    PrintWriter out = res.getWriter();
    out.println("<html><head>");
    out.println("<title>Servlet Principale</title></head>");
    out.println("<body>");
    req.setAttribute("date", new Date());
    RequestDispatcher dist=req.getRequestDispatcher("suite.jsp");
    dist.forward(req, res);
    out.println("</body></html>");
    out.close();
    }
```

Transmission d'un paramètre servlet -> jsp via la requête (2/3)

```
<html>
<head>
<title>Page suite</title>
</head>
<body>
<h2>Le paramètre transmis est : </h2>
<%= request.getAttribute("date") %>
</body>
</html>
```

suite.jsp

Transmission d'un paramètre servlet -> jsp via la requête (3/3)

Transmission d'un paramètre jsp -> servlet via la requête (1/3)

suite.jsp

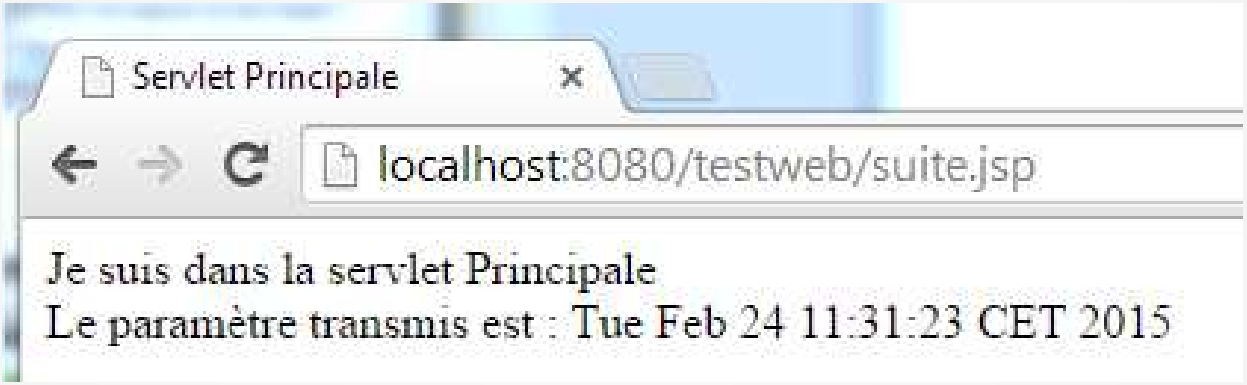
```
<html>
<head>
<title>Page suite</title>
</head>
<body>
<% request.setAttribute("date", new java.util.Date()); %>
<jsp:forward page="Principale" />
</body>
</html>
```

Transmission d'un paramètre jsp -> servlet via la requête (2/3)

Servlet Principale

```
protected void doGet(HttpServletRequest req,
HttpServletResponse res) throws ServletException, IOException {

    res.setContentType("text/html;charset=UTF-8");
    PrintWriter out = res.getWriter();
    out.println("<html><head>");
    out.println("<title>Servlet Principale</title></head>");
    out.println("<body>");
    out.println("Je suis dans la servlet Principale\n");
    Date date= (Date)req.getAttribute( "date");
    out.println("Le paramètre transmis est : "+date);
    out.println("</body></html>");
    out.close();
    }
```

Transmission d'un paramètre jsp -> servlet via la requête (3/3)



Transmission du contrôle: Remarques

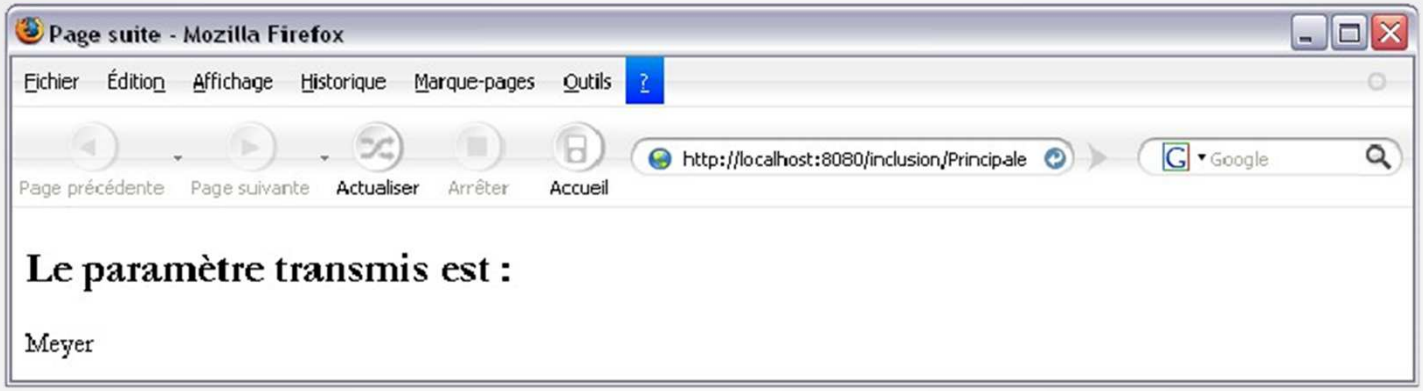
Les choses à ne pas faire :

- Effectuer des modifications sur la réponse avant un renvoi. Elles seraient ignorées.
- Placer du code en séquence à la suite d’un renvoi.

Réponse modifiée avant renvoi (1/2)

```
protected void doGet(HttpServletRequest req,HttpServletResponse res)
    throws ServletException, IOException {

    res.setContentType("text/html;charset=UTF-8");
    PrintWriter out = res.getWriter();
    out.println("<html><head><title>Servlet Principale</title></head>");
    out.println("<body>");
    out.println("<h3>Je suis dans la servlet Principale</h3><hr>");
    RequestDispatcher dist = req.getRequestDispatcher("Secondaire");
    dist.include(req,res);
    out.println("<hr><h3>Je suis à nouveau dans servlet principale</h3>");
    out.println("La réponse est complétée par la servlet principale</h3>");
    out.println("</body></html>");

    ServletContext application = getServletConfig().getServletContext();
    application.setAttribute("nom", "Meyer");
    dist= req.getRequestDispatcher("suite.jsp");
    dist.forward(req, res);
    out.close();
    }
}
```

Réponse modifiée avant renvoi (2/2)



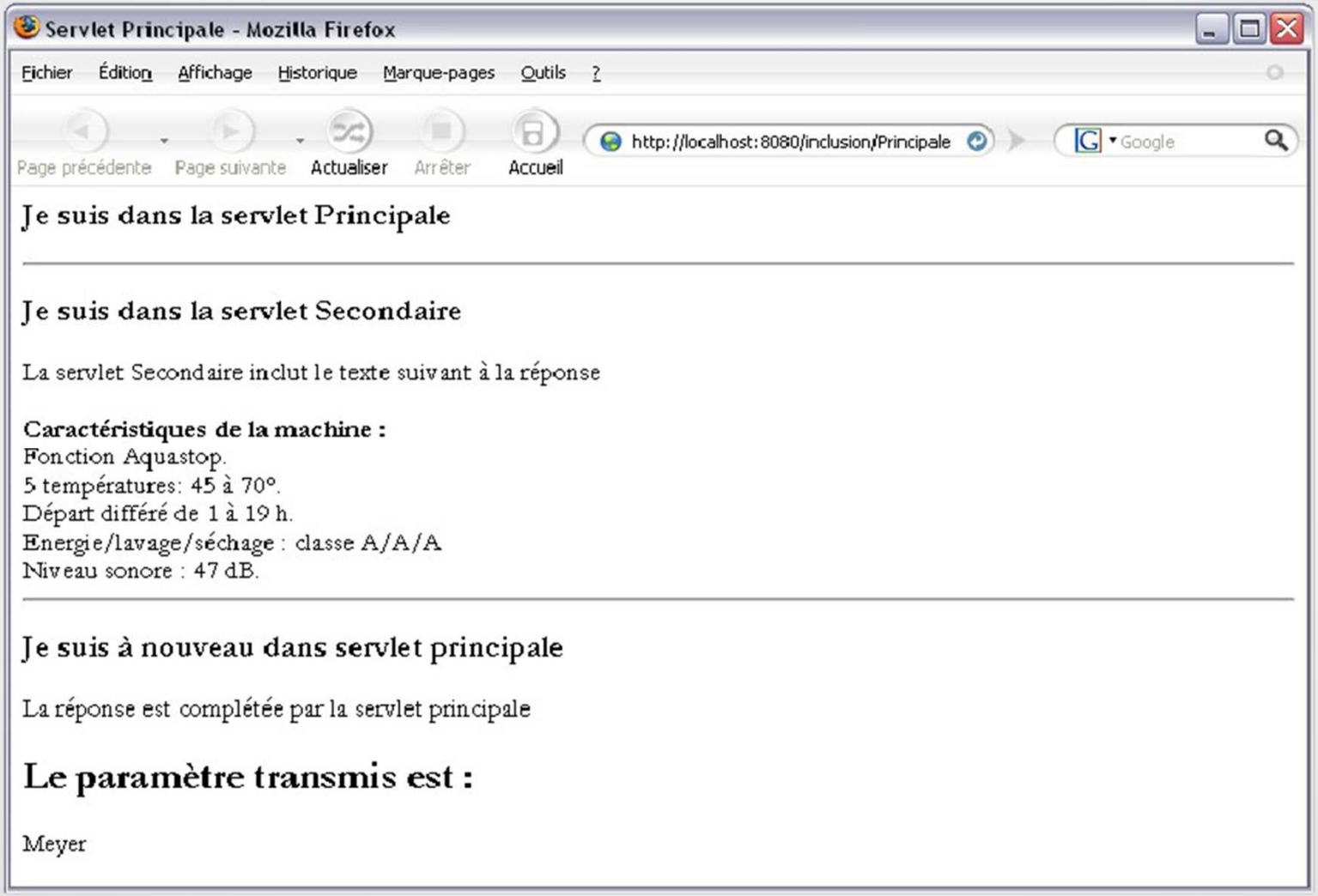
Remarque :

Des modifications (`include`) ont été effectuées sur la réponse avant un renvoi (`forward`).
Elles sont ignorées.

2 inclusions, contrôle conservé (1/2)

```
protected void doGet(HttpServletRequest req,
HttpServletResponse res) throws ServletException, IOException {

    res.setContentType("text/html;charset=UTF-8");
    PrintWriter out = res.getWriter();
    out.println("<html><head><title>Servlet Principale</title></head>");
    out.println("<body>");
    out.println("<h3>Je suis dans la servlet Principale</h3><hr>");
    RequestDispatcher dist = req.getRequestDispatcher("Secondaire");
    dist.include(req,res);
    out.println("<hr><h3>Je suis à nouveau dans servlet principale</h3>");
    out.println("La réponse est complétée par la servlet principale</h3>");
    out.println("</body></html>");
    ServletContext application = getServletConfig().getServletContext();
    application.setAttribute("nom", "Meyer");
    dist= req.getRequestDispatcher("suite.jsp");
    dist.include(req, res);
    out.close();
}
```

2 inclusions, contrôle conservé (2/2)



Transmission d'information jsp vers servlet

- Via un champ caché
- Par réécriture d'URL
- Via la requête
- Via la session
- Via le contexte
- Via l'application

Transmission d'informations
Via un champ caché

Une page JSP peut transmettre de l'information à une servlet via un champ de formulaire caché

```
<form action=Annuaire method=get>
<input type=hidden name=choix value=3>
```

Intérêt : la valeur du paramètre caché permet à la servlet d'aiguiller les actions à entreprendre

```
int choix=Integer.parseInt((String)request.getParameter("choix"));
switch(choix){
      // choix = 1 => affichage de la liste des personnes
      case 1 : action 1;break;
      // choix = 2 => affichage de la liste des invités
      case 2 : action 2;break;
      // choix = 3 => ajout d'une personne dans l'annuaire
      case 3 : action 3;break;
      default:break;
      }
```

Transmission d'informations
Par réécriture d'URL

Une page JSP peut transmettre de l'information à une servlet via la réécriture d'URL

```
<html>
<body>
</body>
    <%
    out.println("<a href='/projet/Annuaire?choix=1'>
           Liste des personnes</a>");
    %>
<br>
    <%
    out.println("<a href='/projet/Annuaire?choix=2'>
           Liste des invit&eacute;s</a>");
    %>
<br>
</html>
```

Transmission d'informations
JSP: Objets implicites

- `Request`

Contient les attributs de la requête, les entêtes HTTP, les cookies

- `Response`

Contient la réponse de la page JSP

- `Session`

Contient les attributs de session disponibles pour toutes les pages qui participent à la session

- `Out`

Représente le flot de sortie pour communiquer la réponse à l'utilisateur

- `Application`

Contient les données partagées par toutes les pages de l'application

Transmission d'informations
JSP: Objets implicites

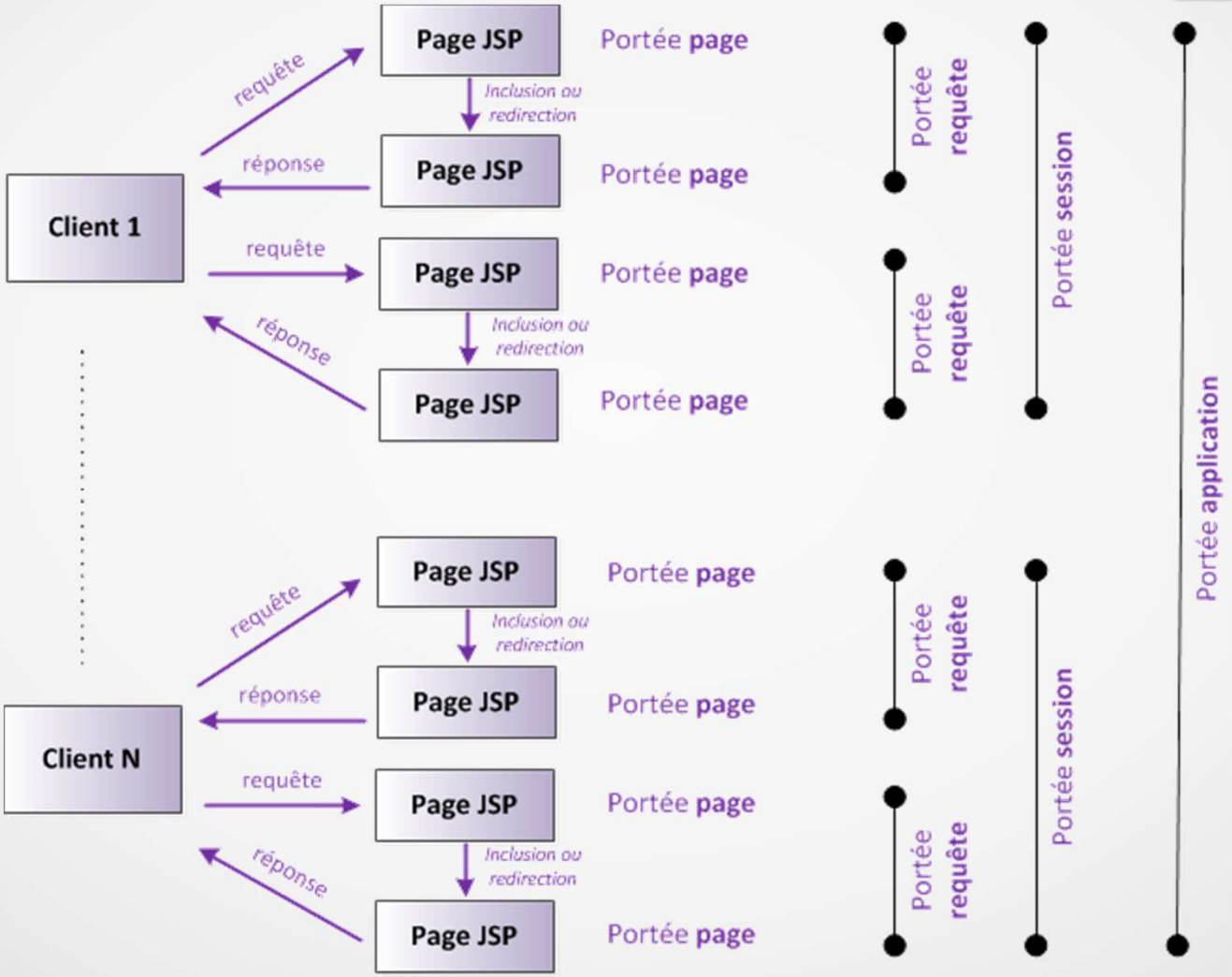
- `pageContext`

Contient les attributs de la page

- `Config`

Contient des informations sur la configuration de la servlet

Transmission d'informations
Portée des objets implicites



Transmission d'informations
Servlets: partage d'information

Une JSP possède des variables définies implicitement :
`request`
`response`
`session`
`application`

Elles permettent de partager des attributs entre JSP et servlets

Les servlets ont accès à ces attributs en créant les objets correspondants :
`request, response` paramètres formels des méthodes GET et POST représentent la requête et la réponse

Pour créer l'objet représentant une session :

```
javax.servlet.http.httpSession session=request.getSession();
```

Transmission d'informations
Informations partagées avec ServletContext(1/2)

A toute application web est associé un contexte auquel les servlets ont accès

Un contexte d'application web est représenté par une instance de la classe `ServletContext` (à partir de l'API Servlet 2.1)

Les servlets accèdent au contexte courant par la méthode `getServletContext()`

```
ServletContext contexte = getServletContext();
```

Les éléments du contexte (variable implicite application dans les JSP) se présentent comme des couples (nom, objet)

Les accesseurs aux éléments du contexte sont :

```
public void setAttribute(String name,Object o)
public Object getAttribute(String name)
public Enumeration getAttributeNames()
public void removeAttribute(String name)
```

Transmission d'informations
Informations partagées avec ServletContext(2/2)

Chaque application web possède son propre contexte
Pour partager des informations avec une autre application web (située sur le même serveur), il faut accéder à un autre contexte.

Marche à suivre :

```
// accès au contexte courant
ServletContext contexteCourant = getServletContext();


// accès à un autre contexte
ServletContext contexteExterieur =
getServletContext("/autreAppli/index.jsp");

// récupération d'un attribut externe
Float qte = (Float)contexteExterieur.getAttribute("quantite");
```

Le chemin doit être absolu

Transmission d'informations
Paramètres d'initialisation partagés avec ServletContext

Permet aux ressources (servlets, jsp, etc...) d'une même application web de partager les mêmes informations initiales (contexte commun).

La classe `ServletContext` fournit 2 méthodes :

- Accès aux paramètres d'initialisation du contexte :

```
public String ServletContext.getInitParameter(String nom)
```

- Accès aux noms des paramètres d'initialisation du contexte

```
public Enumeration ServletContext.getInitParameterNames()
```

Transmission d'informations
Context ou Application ?

Pas de différence entre l'objet `application` implicite et le context retourné par la méthode `getServletContext()`.

```
ServletContext application=getServletContext();
```

Transmission d'informations
Exemple 3 (1/3)

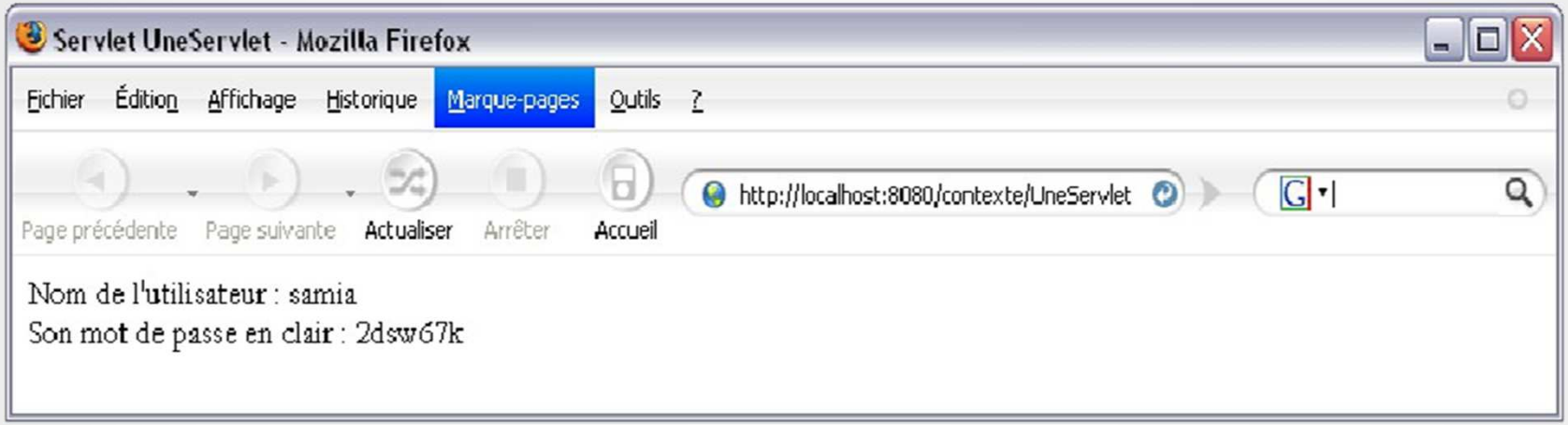
Déploiement du contexte initial : il contient les informations d'authentification pour l'accès à une BD

```
<web-app>
<!-- le contexte de l'application web -->
<context-param>
   <param-name>user</param-name>
      <param-value>samia</param-value>
   </context-param>
<context-param>
   <param-name>passwd</param-name>
    <param-value>2dsw67k</param-value>
</context-param>
</web-app>
```

Transmission d'informations
Exemple 3 (2/3)

```
public class UneServlet extends HttpServlet {
  protected void doGet HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    ServletContext contexte = getServletContext();
    String user = contexte.getInitParameter("user");
    String passwd = contexte.getInitParameter("passwd");
    try {
        out.println("<html><head>");
        out.println("<title>Servlet UneServlet</title>");
        out.println("</head><body>");
        out.println("Nom de l'utilisateur : "+user+"<br>");
        out.println("Son mot de passe en clair : "+passwd);
        out.println("</body></html>");
    } finally {
        out.close();
    }
}
```

Transmission d'informations
Exemple 3 (3/3)



Transmission d'informations
Exemple 4 (1/5)

2 servlets partagent une même information : un vecteur contenant le nom de personnes.

Cette information est enregistrée sous la forme d’un attribut dans le contexte de l’application.

La servlet Inscription enregistre des participants à une réunion.

Un autre servlet Invites accède à la composition de la réunion et affiche la liste des inscrits.

Transmission d'informations
Exemple 4 (2/5)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app SYSTEM "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
<servlet><servlet-name>Inscription</servlet-name>
        <servlet-class>reunion.Inscription</servlet-class>
</servlet>
<servlet><servlet-name>Invites</servlet-name>
        <servlet-class>reunion.Invites</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>Inscription</servlet-name>
        <url-pattern>/Inscription</url-pattern>
</servlet-mapping>
<servlet-mapping>
        <servlet-name>Invites</servlet-name>
        <url-pattern>/Invites</url-pattern>
</servlet-mapping>
</web-app>
```

Transmission d'informations
Exemple 4 (3/5)

```
public class Inscription extends HttpServlet {
    private String nom;
    private Vector invites = new Vector();

    public void doGet(HttpServletRequest req, HttpServletResponse
        res) throws IOException, ServletException{
        res.setContentType( "text/html" );
        PrintWriter out = res.getWriter();
        ServletContext contexte = getServletContext();
        out.println("<FORM ACTION='/reunion/Inscription' METHOD='get'>
        out.println("<INPUT NAME='nom' TYPE='text'/>");
        out.println("<INPUT TYPE='submit‘ VALUE='nouvel inscrit'/>" );
        out.println("</FORM>");

        invites.add(req.getParameter("nom"));
        contexte.setAttribute("invites",invites);
    }
}
```

Transmission d'informations
Exemple 4 (4/5)

```
public class Invites extends HttpServlet {
    public void doGet(HttpServletRequest req,
    HttpServletResponse res) throws IOException, ServletException {

    res.setContentType("text/plain");
    PrintWriter out = res.getWriter();
    ServletContext contexte = getServletContext();
    Vector invites = (Vector)contexte.getAttribute( "invites" );
    Enumeration e = invites.elements();
    out.println("liste des inscrits : ");
    while(e.hasMoreElements())
        out.println(e.nextElement());
    }
}
```

Transmission d'informations
Exemple 4 (5/5)

Servlet `Inscription`

servlet `Invites`

Transmission d'informations
Session

A chaque utilisateur d'un site web est associé un objet de session dans lequel il peut sauvegarder toute donnée utile disponible pendant toute la session.

Obtenir un objet de session dans un programme java :

- l'objet implicite session pour les JSP
- une instance de `javax.servlet.http.HttpSession` pour les servlets

```
HttpSession session = request.getSession();
```

Ajouter une donnée (par exemple une date) dans un objet de session :

```
Date date = new Date();
session.setAttribute("hier",date);
```

Retourner l'objet de nom donné :

```
Date date = (Date)session.getAttribute("hier");
```

Transmission d'informations
Cycle de vie d'une session

Une session expire en cas d'invalidation par une servlet:

```
session.invalidate();
```

On peut obtenir à la suite une nouvelle session:

```
session.request.getSession();
```

En cas de timeout:

```
// on peut fixer la valeur du timeout en secondes
session.setMaxInactiveInterval( 3600 );
// fixer une valeur par défaut dans web.xml
<session-config>
    <session-timeout>
        60<!--en minutes-->
    </session-timeout>
</session-config>
```

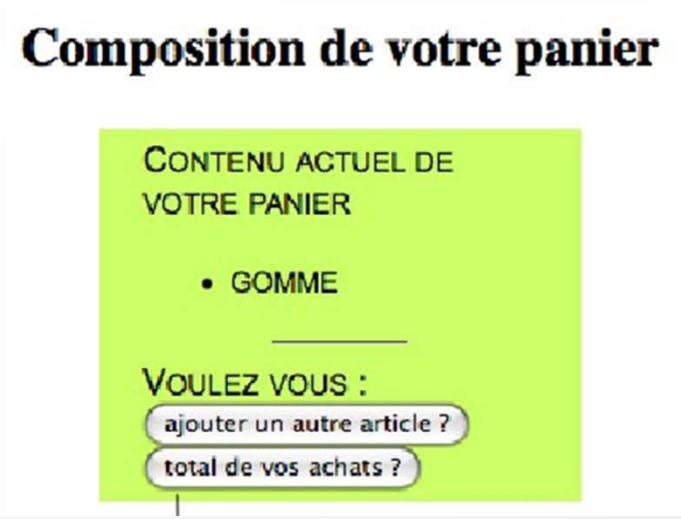
Les données de la session sont supprimées à l'expiration de la session.

Transmission d'informations
Session : exemple 1/7

Une boutique web propose une collection d'articles.

Chaque client de la boutique ouvre une session puis peut ajouter des articles à son panier avant de régler le montant de ses achats.

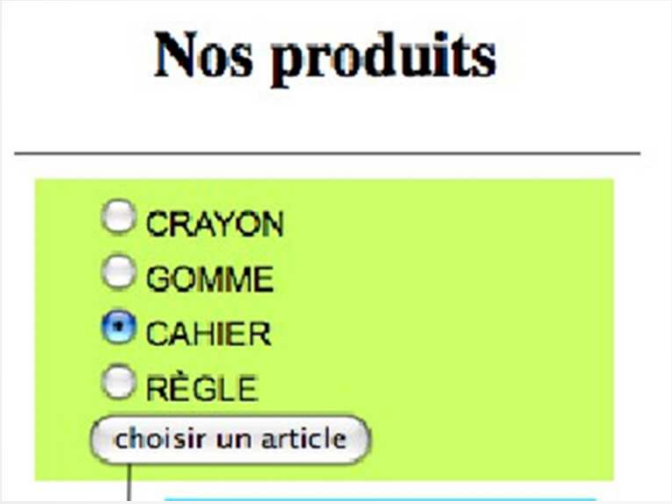
formulaire.jsp



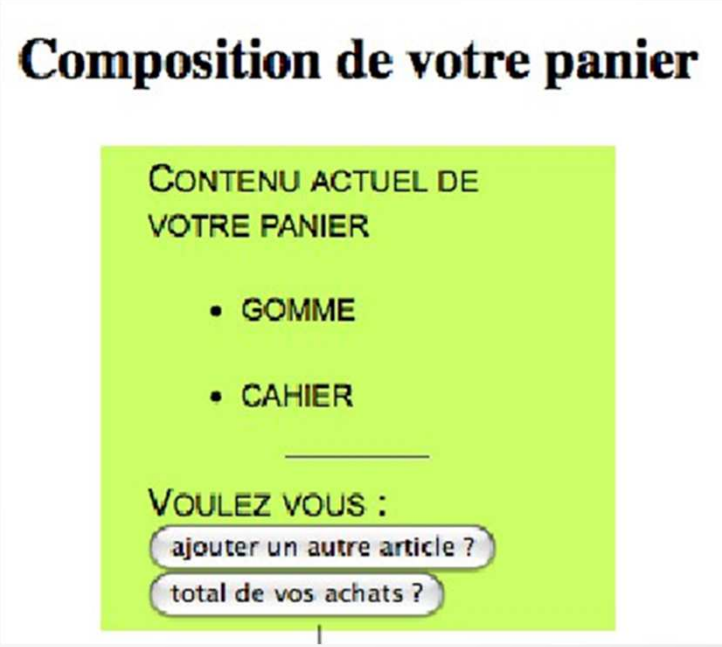
panier.jsp

Transmission d'informations
Session : exemple 2/7

L'utilisateur continue ses achats (suite de requêtes). Les données saisies précédemment sont conservés.



formulaire.jsp

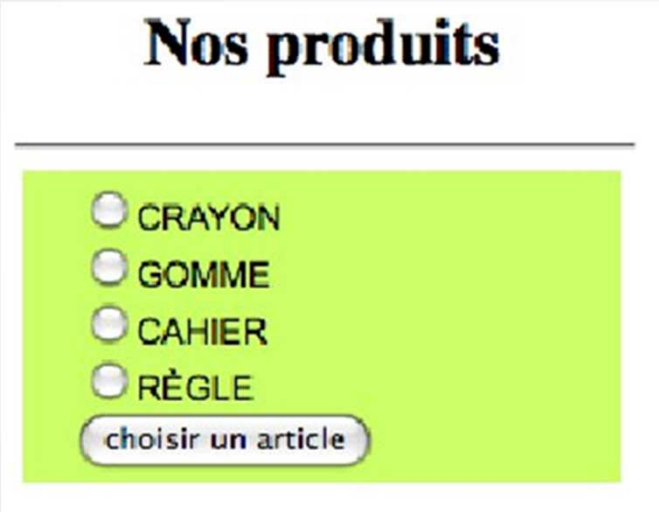


panier.jsp

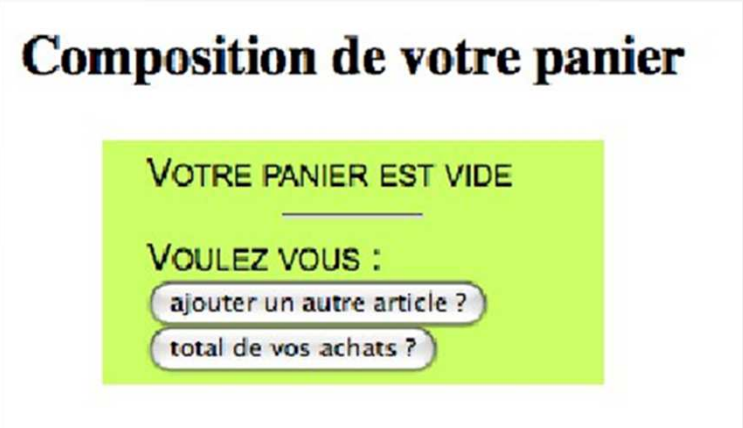
Puis finalement la session est fermée (invalidée).

Transmission d'informations
Session : exemple 3/7

Si on lance de nouveau l'application, le bouton "total des achats" ayant provoqué l'invalidation de la session précédente, on constate que le panier est de nouveau vide.

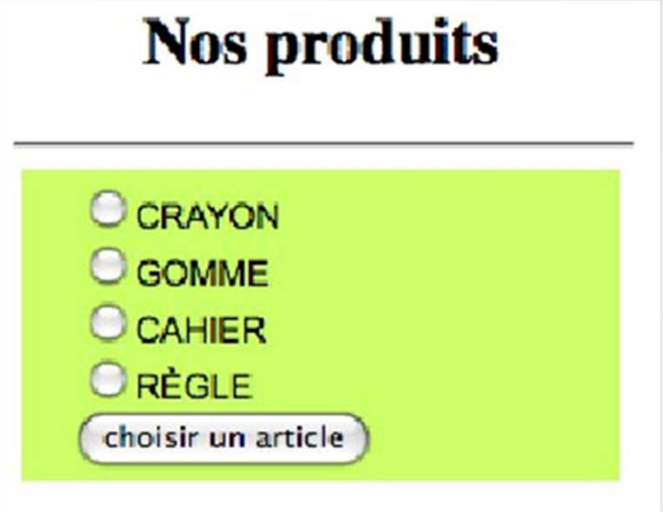


formulaire.jsp



panier.jsp

Transmission d'informations
Session : exemple 4/7




```
<html><!-- page jsp : formulaire.jsp -->
    <head>
        <title>JSP Page</title>
    </head>
<body>
    <h1 align='center'>web boutique</h1>
    <h2 align='center'>Nos produits</h2><hr>
    <div id='zone'>
        <form action='Boutique' method='get'>
            <input type='radio' name='radio' value='crayon'><h7>crayon</h7><br>
            <input type='radio' name='radio' value='gomme'><h7>gomme</h7><br>
            <input type='radio' name='radio' value='cahier'><h7>cahier</h7><br>
            <input type='radio' name='radio' value='règle'><h7>règle</h7><br>
            <input name='choix' type='submit' value='choisir un article'>
        </form>
    </div>
</body>
</html>
```

Transmission d'informations
Session : exemple 5/7

Servlet Boutique

```
// Servlet Boutique : méthode doGet
protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    HttpSession session = request.getSession();
    String choix = request.getParameter("choix");
    if( choix!=null){
        String article = request.getParameter("radio");
        if(article!=null){
            panier.add(article);
            session.setAttribute("panier", panier);
            }
        RequestDispatcher dispatcher=
        request.getRequestDispatcher("panier.jsp");
        }else{
        RequestDispatcher dispatcher=
        request.getRequestDispatcher("formulaire.jsp");
        }
    dispatcher.forward(request,response);
}
```

variable d'instance
ArrayList<String> panier = new ArrayList<String>();

Transmission d'informations
Session : exemple 6/7

panier.jsp

```
<h2 align='center'>Composition de votre panier</h2>
<% ArrayList<String> articles=
(ArrayList<String>)session.getAttribute("panier");
if( articles != null ){
%>
        <h7>Contenu actuel de votre panier</h7>
<%  for(String article:articles){%>
<ul><li><h7><%= article%></h7></li></ul>
<% }}else{ %>
        <h7>Votre panier est vide</h7>
<%}%>
<hr/><h7>Voulez vous :</h7><br/>
<form action='Boutique' method='get' >
<input type='submit' name ='ajouter' value='ajouter un article ?'>
</form><br/>
<form action='Boutique' method='post' >
<input type='submit' name='total' value='total de vos achats ?'>
</form>
```

Transmission d'informations
Session : exemple 7/7

```
// Servlet Boutique : méthode doPost
@Override
protected void doPost(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    HttpSession session = request.getSession();
    String total = request.getParameter("total");
    if(total!=null) {
       if(total.equals("total de vos achats ?")){
            panier.clear();
            session.invalidate();
            }
       }
}
```

Transmission d'informations
Partage par la variable implicite `application`

Contexte d'exécution = ensemble de couples `(name,value)` partagées par toutes les JSP/servlets instanciées

Objet prédéfini `application`

Méthodes invocables sur l'objet prédéfini `application`

- `void setAttribute(String name, Object value)`

ajoute un couple `(name, value)` dans le contexte

- `Object getAttribute(String name)`

retourne l'objet associé à la clé `name` ou `null`

- `void removeAttribute(String name)`

enlève le couple de clé `name`

- `java.util.Enumeration getAttributeNames()`

retourne tous les noms d'attributs associés au contexte

Transmission d'informations
Partage par la variable implicite `application`

Entre jsp : index.jsp

```
<html>
<head><title>Page index.jsp</title></head>
<body>
<% application.setAttribute("nom","Meyer");%>
<jsp:forward page="suite.jsp" />
</body>
</html>
```

suite.jsp

```
<html>
<head><title>Page suite.jsp</title></head>
<body>
<h2>Le paramètre transmis est : </h2>
<%= application.getAttribute("nom")     %>
</body>
</html>
```

Transmission d'informations
Partage par la variable implicite `application`

Transmission d'informations
Partage par la variable implicite application

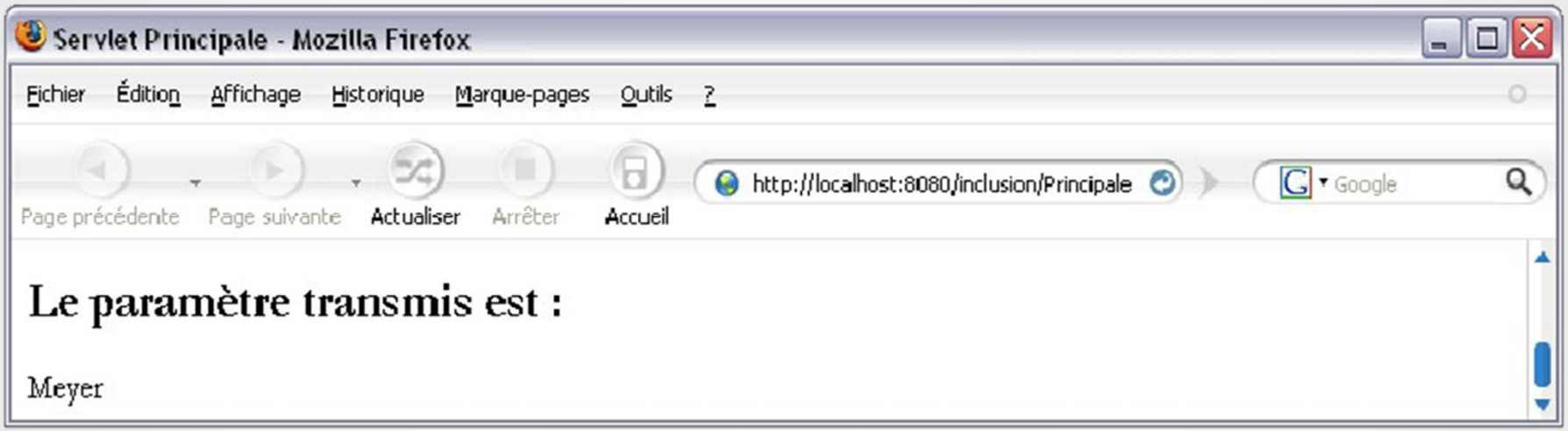
Entre servlet et jsp :

```
public class Principale extends HttpServlet {
protected void doGet(HttpServletRequest
req,HttpServletResponse res)
throws ServletException, IOException {
    res.setContentType("text/html;charset=UTF-8");
    ServletContext contexte =
    getServletConfig().getServletContext();
    contexte.setAttribute("nom", "Meyer");
    RequestDispatcher
    dist=req.getRequestDispatcher("suite.jsp");
    dist.include(req, res);
    }
}
```

```
<html><head><title>Page
suite.jsp</title></head>
<body><h2>Le paramètre transmis est : </h2>
<%= application.getAttribute("nom") %>
</body></html>
```

Transmission d'informations
Partage par la variable implicite `application`

Entre servlet et jsp



ServletContext vs ServletConfig

- `ServletConfig`

1 objet par servlet.
Fourni par le serveur web après instanciation de la servlet.
Utilisé par le container de servlet pour transmettre les infos d'initialisation à la servlet.
Les paramètres d'initialisation doivent être fixés dans le descripteur de déploiement (web.xml)
Accessible par tout utilisateur accédant la servlet.

- `ServletContext`

1 objet par application.
Défini un ensemble de méthodes pour communiquer avec le container de servlets.
Portée application.
Objet inclus dans l'instance de ServletConfig.