Java EE

Table des matières

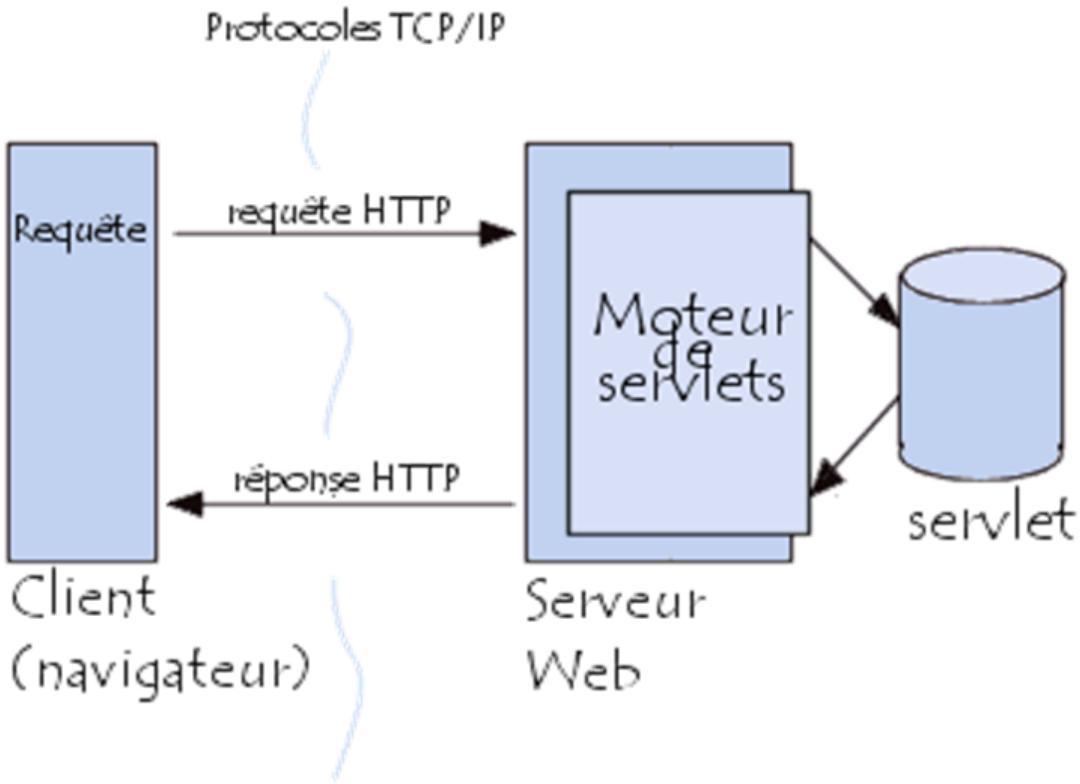
Architecture Java EE

1

A. Principe

Définition : servlet

C'est une application Java qui tourne dans une VM sur le serveur.



Intérêts d'une servlet

- exécuté sur le serveur Web
- envoie des pages HTML générées à la volée
- séparation du code du programme des données
- portabilité de l'application

Mise en place

- Il faut disposer d'un serveur Java EE.

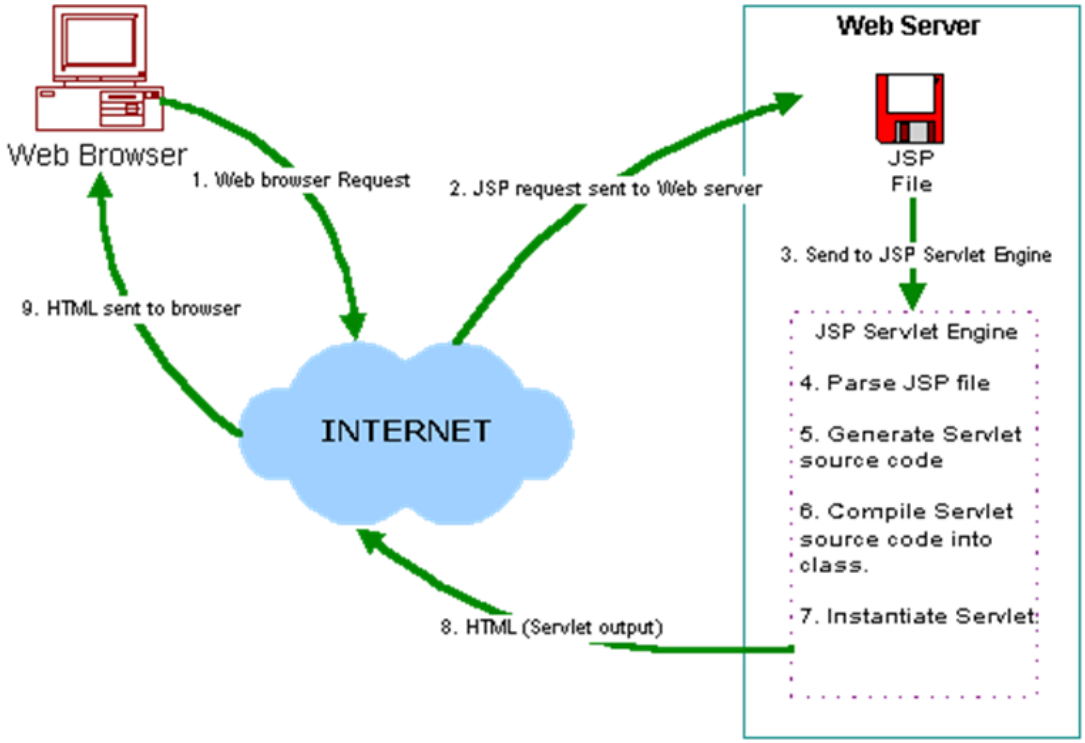
Syntaxe : langage Java

- API standard
- API Java EE : classes particulières pour les interactions avec le serveur et avec http/html

  par exemple classe javax.servlet.HttpServlet

Définition : JSP (Java Server Page)

C'est une page HTML avec des balises supplémentaires, traduite à la volée en servlet, compilée dynamiquement et exécutée dans une VM sur le serveur.



Intérêts dune page JSP

- indépendance de la plate forme
- utilisation de Javabeans et Enterprise Java Bean (EJB)
- séparation entre le code HTML et le code applicatif

Mise en place

- Il faut disposer d'un serveur de pages JSP

Syntaxe

Le code jsp se place dans les pages html ou dans des pages jsp propres.

- C'est un langage avec des tags qui s'implante au sein du html. ·
- Commentaires <%-- %> ·

- Declaration tag <%! %>
- Expression tag <%= %>
- Directive tag <%@ %>
- Scriptlet tag <% %>
- Actions

```
1  <html>
2  <head>
3  <title>Ma premiere page JSP </title>
4  </head>
5  <body>
6  <%@ page language="java" %>
7  <% out.println("Bonjour"); %>
8  </body>
9  </html>
```

B. Moteurs Java EE

Certifié Java EE 8

- Oracle GlassFish Server Open Source
- IBM WebSphere Application Server
- RedHat Wildfly

Certifié Java EE 7

- Oracle GlassFish Server Open Source Edition 4.04
- TmaxSoft TMAX JEUS 85
- RedHat Wildfly 8.0.06

Certifié Java EE 6

- Oracle GlassFish Enterprise Server v3, basé sur le serveur open-source GlassFish
- Oracle WebLogic Server 12c de Oracle Corporation
- JBoss AS 7.x
- TmaxSoft JEUS 7
- Apache Geronimo 3.0
- IBM WebSphere Application Server 8.0
- IBM WebSphere Application Server Community Edition 3.0, basé sur Apache Geronimo

Certifié Java EE 5 / J2EE 1.5

- Sun Java System Application Server Platform Edition 9.0, basé sur le serveur open-source GlassFish
- Oracle WebLogic Server 10gR3
- SAP NetWeaver Application Server, Java EE 5 Edition
- TmaxSoft JEUS 6
- Apache Geronimo 2.0
- IBM WebSphere Application Server Community Edition 2.0, basé sur Apache Geronimo
- IBM WebSphere Application Server v7

- Oracle Containers for Java EE 11
- GlassFish
- JBoss 5.0.0
- JOnAS 5,

C. Versions

Java Enterprise Edition

Version	Année
J2EE 1.2	1999
J2EE 1.2	2001
J2EE 1.4	2003
J2EE 1.5 / Java EE 5	2006
Java EE 6	2009
Java EE 7	2013
Java EE 8	2017

Web profile

Sous ensemble de Java EE

- Servlet
- JavaServer Pages
- JavaServer Pages Standard Tag Library
- JavaServer Faces
- ...

API servlet

II

A. Paquetage et classe

- interface javax.servlet.Servlet
- classe javax.servlet.GenericServlet
  service()
- classe javax.servlet.HttpServlet
  doGet()
  doPost()

B. Servlet minimale

Classe dérivée de GenericServlet

- implémentation de service(), appelée par le serveur quand on appelle la servlet
- appel de getWriter pour obtenir un PrintWriter
- appel de setContent pour définir le type de sortie

```
1  import javax.servlet.*;
2  import java.io.*;
3
4  public class BasicServlet extends GenericServlet {
5
```

```
6     public void service(ServletRequest req, ServletResponse resp)
7       throws ServletException, IOException {
8
9       resp.setContentType("text/plain");
10      PrintWriter out = resp.getWriter();
11
12      out.println("Hello.");
13    }
14  }
15
```

Classe dérivée de HttpServlet

- implémentation de doGet(), appelée par la méthode service() par défaut de HttpServlet
- envoi de données HTML
- getLastModified() permet d'indiquer au serveur si la page a été modifiée

```
1   import javax.servlet.*;
2   import javax.servlet.http.*;
3   import java.io.*;
4
5   public class HelloWorldServlet extends HttpServlet {
6
7     public void doGet(HttpServletRequest req, HttpServletResponse resp)
8       throws ServletException, IOException {
9
10      resp.setContentType("text/html");
11      PrintWriter out = resp.getWriter();
12
13      out.println("<HTML>");
14      out.println("<HEAD><TITLE>Have you seen this before?
    </TITLE></HEAD>");
15      out.println("<BODY><H1>Hello, World!
    </H1><H6>Again.</H6></BODY></HTML>");
16    }
17  }
18
```

C. Cycle de vie d'une servlet

Objet persistant

- Un objet est instancié lors du premier appel à la servlet.
- Cet objet est persistant tant qu'il n'est pas arrêté par le serveur J2E (arrêt du serveur, arrêt de l'application)

Deux méthodes liées au cycle de vie

- init
- destroy

Méthode init(ServletConfig svcfg)

- elle est appelée au démarrage de la servlet par le serveur, qui attend sa fin avant d'appeler service()
- il ne faut pas oublier d'appeler la méthode de la classe de base : super.init(svcfg)
- elle effectue les réservations de ressources (connexion base de données, lancement de threads, ...)

- la classe ServletConfig permet l'accès aux informations de configuration du serveur et aux paramètres définis par l'administrateur

Méthode destroy()

- elle est appelée par le serveur lors de l'arrêt
- il est inutile d'appeler la méthode de la classe de base
- elle effectue la libération des ressources

```
1  import javax.servlet.*;
2  import javax.servlet.http.*;
3  import java.io.*;
4
5  public class LifeCycleServlet extends HttpServlet {
6    int timesAccessed;
7    public void init(ServletConfig conf) throws ServletException {
8      super.init(conf);
9     try {
10       timesAccessed =
Integer.parseInt(getInitParameter("defaultStart"));
11     }
12     catch(NullPointerException e) {
13    timesAccessed = 0;
14     }
15     catch(NumberFormatException e) {
16        timesAccessed = 0;
17     }
18     try {
19       File r = new File("./data/counter.dat");
20       DataInputStream ds = new DataInputStream(new
FileInputStream(r));
21       timesAccessed = ds.readInt();
22       ds.close();
23     }
24     catch (FileNotFoundException e) {
25       // Handle error
26     } catch (IOException e) {
27       // This should be logged
28     }
29   }
30    public void doGet(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
31     resp.setContentType("text/html");
32     PrintWriter out = resp.getWriter();
33     timesAccessed++;
34     out.println("<HTML>");
35     out.println("<HEAD>");
36     out.println("<TITLE>Life Cycle Servlet</TITLE>");
37     out.println("</HEAD><BODY>");
38     out.println("I have been accessed " + timesAccessed + "
time[s]");
39    out.println("</BODY></HTML>");
40   }
41   public void destroy() {
42     // Write the Integer to a file
43     File r = new File("./data/counter.dat");
44     try {
45       DataOutputStream dout = new DataOutputStream(new
FileOutputStream(r));
46       dout.writeInt(timesAccessed);
47       dout.close();
48     }
49     catch(IOException e) {
50       // This should be logged
51     }
52   }
53 }
```

Classe ServletConfig

- getInitParameter(String Name)
- getInitParameterName()
- getServletContext() fournit un objet ServletContext utilisé par la Sservlet pour dialoguer avec le Serveur et les autres servlets

Classe ServletContext

- getMimeType(String file)
- getServerInfo()
- log(String Message)
- getAttribute(String name), getAttributesNames()
- setAttribute(String name, Object object)
- removeAttribute(String name)

D. Traitement d'une requête

1. Gestion d'une requête

HttpServlet

- doGet
- doPost
- doPut
- ...

```
1  import javax.servlet.*;
2  import javax.servlet.http.*;
3  import java.io.*;
4
5  public class HelloServlet extends HttpServlet {
6    public void doGet(HttpServletRequest req, HttpServletResponse resp)
7      throws ServletException, IOException {
8
9      resp.setContentType("text/html");
10     PrintWriter out = resp.getWriter();
11 out.println("<HTML>");
12     out.println("<HEAD><TITLE>Finally, interaction!</TITLE></HEAD>");
13     out.println("<BODY><H1>Hello, " + req.getParameter("username") +
   "!</H1>");
14     out.println("</BODY></HTML>");
15   }
16 }
17
```

GenereicServlet

- service

```
1  import javax.servlet.*;
2  import java.io.*;
3
4  public class BasicServlet extends GenericServlet {
5
6    public void service(ServletRequest req, ServletResponse resp)
```

```
7       throws ServletException, IOException {
8
9       resp.setContentType("text/plain");
10      PrintWriter out = resp.getWriter();
11
12      out.println("Hello.");
13    }
14  }
15
```

2. Information sur la requête

ServletRequest, HttpServletRequest

- getProtocol
- getRemoteHost
- getServerName, getServerPort
- getParameter
- getParameterValues() : fournit un tableau de chaines

HttpServletRequest

- getHeader
- getHeaderNames()

3. Réponse d'une servlet

ServletResponse, HttpServletResponse

- getOutputStream()
  fournit une instance de ServletOutputStream
  pour écriture de texte ou de données binaires
- getWriter()
  fournit une instance de java.io.PrintWriter pour écriture de texte
  Attention : il faut appeler setContentType() avant getWriter()

HttpServletResponse

- setHeader, setIntHeader, setDateHeader
- containsHeader
- setStatus(), sendError()
- sendRedirect()

E. Cookie

classe Cookie

HttpServletRequest.getCookies() fournit un tableau de Cookie
HttpServletResponse.addCookie ajoute un cookie à la réponse
constructeur du cookie
méthodes getName, getValue

```
1  import java.io.*;
```

```
2  import javax.servlet.*;
3  import javax.servlet.http.*;
4  public class CookieExample extends HttpServlet {
5     public void doGet(HttpServletRequest request,
       HttpServletResponse response)
6     throws IOException, ServletException {
7
8       response.setContentType("text/html");
9       PrintWriter out = response.getWriter();
10
11   // print out cookies
12       Cookie[] cookies = request.getCookies();
13 for (int i = 0; i < cookies.length; i++) {
14         Cookie c = cookies[i];
15         String name = c.getName();
16         String value = c.getValue();
17         out.println(name + " = " + value);
18       }
19 // set a cookie
20       String name = request.getParameter("cookieName");

21       if (name != null && name.length() > 0) {
22         String value = request.getParameter("cookieValue");
23         Cookie c = new Cookie(name, value);
24         response.addCookie(c);
25       }
26     }
27   }
28
```

F. Session

classe HttpSession

HttpServletRequest.getSession fournit l'instance de HttpSession

Accès aux variables de session :

- getValue
- putValue

```
1  public class VisitCounterServlet extends HttpServlet {
2
3    public void doGet(HttpServletRequest req, HttpServletResponse resp)
4      throws ServletException, IOException {
5
6      PrintWriter out = resp.getWriter();
7      resp.setContentType("text/html");
8
9      HttpSession thisUser = req.getSession(true);
10     Integer visits;
11
12     if(!thisUser.isNew()) {          //Don't check newly created
 sessions
13       visits = (Integer)thisUser.getValue("visitcounter.visits");
14       if(visits == null)
15         visits = new Integer(1);
16       else
17 visits = new Integer(visits.intValue() + 1);
18     }
19     else
20       visits = new Integer(1);
21
22     // Put the new count in the session
23     thisUser.putValue("visitcounter.visits", visits);
```

```
24
25      // Finally, display the results and give them the session ID too
26      out.println("<HTML><HEAD><TITLE>Visit Counter</TITLE></HEAD>");
27      out.println("<BODY>You have visited this page " + visits + "
        time[s]");
28      out.println("since your last session expired.");
29      out.println("Your Session ID is " + thisUser.getId());
30      out.println("</BODY></HTML>");
31    }
32  }
33
34
```

G. Répartition de requêtes

RequestDispatcher

ServletContext.getRequestDispatcher() fournit une instance de RequestDispatcher
forward() : passe le traitement à une autre servlet
include() : inclut la sortie d'une autre servlet dans la sortie principale

Passage de paramètres

méthodes de ServletContext

- setAttribute(String name, Object obj)
- getAttribute(String Name)

```
1   import javax.servlet.*;
2   import javax.servlet.http.*;
3
4   public class servletToJsp extends HttpServlet {
5
6       public void doGet (HttpServletRequest request,
7             HttpServletResponse response) {
8
9    try {
10      // Set the attribute and Forward to hello.jsp
11      request.setAttribute ("servletName", "servletToJsp");
12
    getServletConfig().getServletContext().getRequestDispatcher("/jsp/jsp
    toserv/hello.jsp").forward(request, response);
13   } catch (Exception ex) {
14      ex.printStackTrace ();
15   }
16      }
17  }
18
```

H. Synchronisation

Par défaut, un thread pour chaque requête

- Instance unique
- Pool d'instance maintenu par le serveur

Interface SingleThreadModel

- Une instance ne peut être exécutée que par un thread à la fois
- une instance unique implique la création d'une file d'attente
- Le pool d'instance effectue les opérations d'extraction, d'exécution et de retour

Interface class MaServlet extends HttpServlet implements SingleThreadModel

Il n'y a pas de méthode à écrire !

Attention : SingleThreadModel ne résout pas tous les cas de synchronisation

- Utilisation de synchronized peut être plus efficace
- Il faut veiller aux accès concurrents à une base de donnée

I. Filtres

Principes

- Traitements des requêtes avant leur passage à la servlet
- Traitements des réponses fournies par la servlet
- Exemples
  - Compression/décompression
  - Transformation XSLT

Définition : Filtre

Classe qui implémente l'interface javax.servlet.Filter

Méthodes à définir

init() // initialisation

doFilter() // travail effectif

destroy() // libération des ressources

```
1  import javax.servlet.*;
2  import java.io.*;
3
4  public class MonFiltre implements Filter {
5    private FilterConfig filterConfig = null ;
6    public void init(FilterConfig filterConfig) throws ServletException
   {
7      this.filterConfig = filterConfig ;
8      ...
9    }
10
11 public void doFilter(ServletRequest req, ServletResponse res,
   FilterChain chain) throws ServletException, IOException {
12     if(filterConfig == null) return ;
13     ...
14     chain.doFilter(request, response) ;
15     ...
16   }
17
18   public void destroy() {
19     this.filterConfig = null ;
```

```
20          ...
21      }
22  }
23
24
```

Configuration avec l'Interface FilterConfig

getFilterName()
getServletContext()
getInitParameter(String nom)
getInitParameterNames()

Chaînage de filtres avec l'Interface FilterChain

Méthode doFilter pour invoquer le filtre ou la servlet suivante

Déploiement avec le fichier web.xml

```
1  <filter>
2    <filter-name>...</filter-name>
3    <description>...</description>
4    <filter-class>...</filter-class>
5    <init-param>
6      <param-name>...</param-name>
7      <param-value>...</param-value>
8    </init-param>
9  </filter>
10 <filter>
11   ...
12 </filter>
13
14 <filter-mapping>
15   <filter-name>...</filter-name>
16   <url-pattern>...</url-pattern>
17 </filter-mapping>
18 <filter-mapping>
19   <filter-name>...</filter-name>
20   <servlet-name>...</servlet-name>
21 </filter-mapping>
22 <filter-mapping>
23   ...
24 </filter-mapping>
25
```

J. Événements

1. Événements de contexte

Interface ServletContextListener

Création, destruction du contexte

Interface ServletContextAttributeListener

Création, modification, destruction d'un attribut

a) Interface ServletContextListener

Méthodes à définir

- contextInitialized(ServletContextEvent sce)
- contextDestroyed(ServletContextEvent sce)

ServletContextEvent.getServletContext fournit l'instance de ServletContext de la servlet

b) Interface ServletContextAttributeListener

Méthodes à définir

- attributeAdded(ServletContextAttributeEvent eve)
- attributeReplaced(ServletContextAttributeEvent sce)
- attributeRemoved(ServletContextAttributeEvent sce)

c) ServletContextAttributeEvent

Méthodes

- getName
- getValue

2. Événements de session

Interface HttpSessionListener

Création, destruction d'une session

Interface HttpSessionActivationListener

Activation, passivation d'une session

Interface HttpSessionAttributeListener

Création, modification, destruction d'un paramètre

a) Interface HttpSessionListener

Méthodes à définir

- sessionCreated(HttpSessionEvent eve)
- sessionDestroyed(HttpSessionEvent eve

HttpSessionEvent.getSession fournit l'instance de HttpSession

b) Interface HttpSessionActivationListener

Méthodes à définir

- sessionWillPassivate(HttpSessionEvent eve)
- sessionDidActivate(HttpSessionEvent eve)

HttpSessionEvent.getSession fournit l'instance de HttpSession

c) Interface HttpSessionAttributeListener

Méthodes à définir

- attributeAdded(HttpSessionBindingEvent eve)

- attributeReplaced(HttpSessionBindingEvent sce)
- attributeRemoved(HttpSessionBindingEventsce)

d) HttpSessionBindingEvent

Méthodes

- getName
- getValue
- getSession

e) Exemple

```
1  import javax.servlet.*;
2  import javax.servlet.http.*;
3
4  public class GestionEvtHttpSession implements HttpSessionListener,
   HttpSessionAttributeListener {
5
6   public void sessionDestroyed(HttpSessionEvent hse){}
7   public void attributeReplaced(HttpSessionBindingEvent hsbe){}
8   public void attributeRemoved(HttpSessionBindingEvent hsbe){}
9
10  public void sessionCreated(HttpSessionEvent hse){
11    HttpSession session = hse.getSession();
12    session.setAttribute("caddie",new java.util.Hashtable());
13  }
14 public void attributeAdded(HttpSessionBindingEvent hsbe){
15    String nomAttribut = hsbe.getName();
16    Object valAttribut = hsbe.getValue();
17    HttpSession session = hsbe.getSession();
18    String idSession = session.getId();
19    ServletContext contextApp = session.getServletContext();
20
21    StringBuffer message = new StringBuffer();
22    message.append("\t\tAttribut ajouté dans la session (id=" +
   idSession + "),");
23    message.append(" avec le nom '" + nomAttribut + "' et la valeur '"
   + valAttribut + "'.");;
24    contextApp.log(message.toString());
25  }
26 }
27
```

3. Déploiement dans web.xml

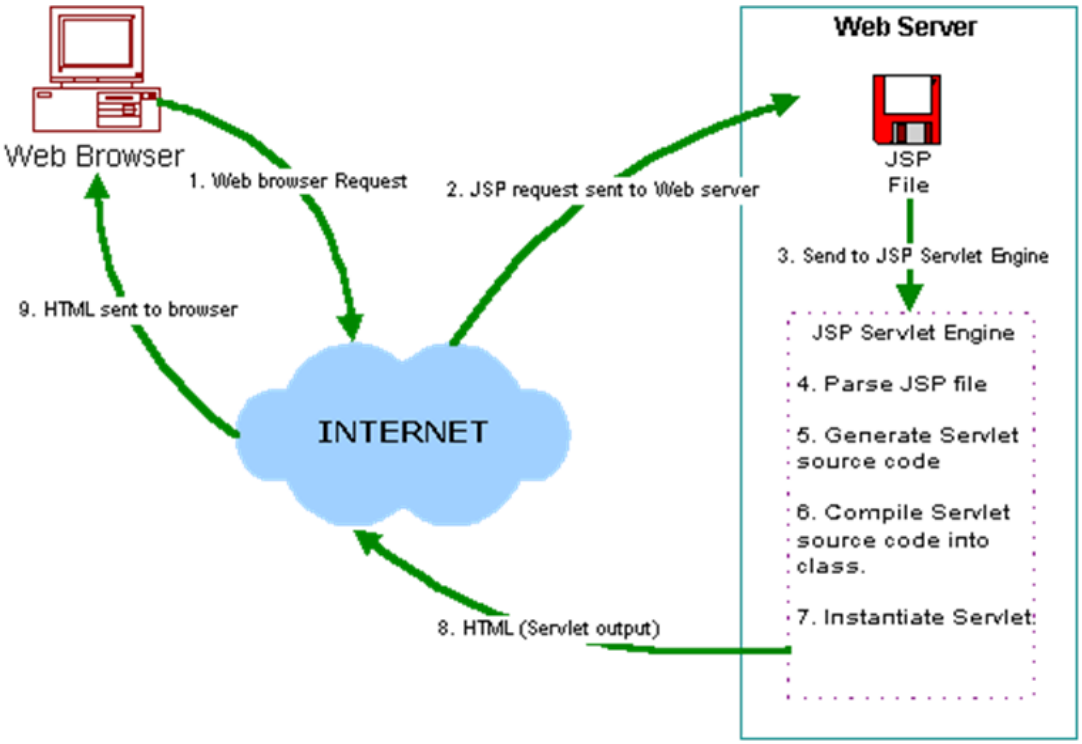
```
1  ...
2  </filter-mapping>
3  <listener>
4   <listener-class>nom</listener-class>
5  </listener>
6  <servlet>
7  ...
8
```

Page JSP

III

A. Traitement d'une requête de page JSP

Cycle de vie



B. Structure d'une page

```
1  <html>
2  <head>
3  <title>My first JSP page
4  </title>
5  </head>
6  <body>
7  <%@ page language="java" %>
8  <% out.println("Hello World"); %>
9  </body>
10 </html>
11
```

Balises

- Commentaires <%-- %>
- Declaration tag <%! %>
- Expression tag <%= %>
- Directive tag <%@ %>
- Scriptlet tag <% %>
- Actions

C. Declaration tag (<%! %>)

définition de variables ou de méthodes

```
1  <%!
2  private int counter = 0 ;
3  private String get Account ( int accountNo) ;
4  %>
5
```

- Il faut respecter la syntaxe Java
- Les instructions sont terminées par un ;

D. Expression tag (<%= %>)

abréviation pour out.println()

- On peut placer n'importe quelle expression
- Il ne faut pas mettre de ;

```
1  Date : <%= new java.util.Date() %>
```

E. Directive tag (<%@ directive ... %>)

définition d'informations spécifiques

- page : traitement d'information pour la page
- include : fichier à inclure
- tag library : tag library à utiliser pour la page

1. page

language

Langage employé

```
1  <%@ page language = "java" %>
```

extends

Classe de base utilisée par JSP pour la Servlet

```
1  <%@ page extends = "com.taglib... %>
```

import

Importe toutes les classes d'un paquetage java dans la page courante. Permet d'utiliser d'autres classes java dans la page

```
1  <%@ page import = "java.util.*" %>
```

session

Indique si la page utilise des sessions.
Par défaut, true.
Passer à false améliore les performances.

buffer

Contrôle l'utilisation du buffer de sortie, 8 kb par défaut

```
1 <%@ page buffer = "none" %>
```

autoFlush

vidage automatique du buffer quand il est plein

```
1 <%@ page autoFlush = "true" %>
```

isThreadSafe

indique si la servlet peut gérer des requêtes multiples.
Si true, un nouveau thread est lancé pour gérer des requêtes simultanées

infoDeveloper

ajoute des informations sur la page. (Auteur, version, copyright et date)

```
1 <%@ page info = "societe.com test page, copyright 2013. " %>
```

errorPage

URL de la page pour traiter les erreurs

```
1 <%@ page errorPage = "/error/error.jsp" %>
```

IsErrorPage

Si true, cette page est une Error Page spéciale, qui a accès à l'objet implicite exception.

contentType

Définit le type mime et le jeu de caractères

2. include

insertion d'une page

```
1 <%@ include file = "include/privacy.html" %>
2 <%@ include file = "navigation.jsp" %>
3
```

3. tag

tags personnalisés utilisés dans la page

```
<%@ taglib uri = "tag library URI" prefix = "tag Prefix" %>
```

F. Scriptlet tag (<% ... %>)

Définition d'instructions Java à exécuter

```
<%
  String username = "Dupond" ;
 out.println ( username ) ;
  %>

```

G. Objets implicites

Variable	Type
request	javax.servlet.http.HttpServletRequest
response	javax.servlet.http.HttpServletResponse
out	javax.servlet.jsp.JspWriter
session	javax.servlet.http.HttpSession
pageContext	javax.servlet.jsp.PageContext
application	javax.servlet.http.ServletContext
config	javax.servlet.http.ServletConfig
page	java.lang.Object

page : page JSP, utilisée pour appeler n'importe quelle méthode de la servlet
config : configuration de la Servlet
request : informations associées à une requête

```
<% String devStr = request.getParameter("dev"); %>
Development language = <%= devStr %>

```

out : accès à des méthodes d'écriture (print, println)
application : références à des objets partagés
session : références aux données de session

H. Cycle de vie

Méthodes appelées au lancement et à l'arrêt

jspInit() au lieu de init()
jspDestroy() au lieu de destroy()

I. Actions

1. appels de Javabean

Définition : Javabean

classe contenant des méthodes de traitement et des propriétés (conteneur)

Utilisation d'un javabean

```
1 <jsp:usebean id="nom" scope="application" class="cl" />
```

- nom : identificateur de l'instance
- cl : classe du javabean
- scope : portée de l'instance
  - page : valide jusqu'à la fin de cette page
  - request : valide pour la requête du client
  - session : valide pour la session du client
  - application : valide jusqu'à la fin de l'application

2. Définition de propriétés

Utilisation d'un javabean et de ses propriétés

nom : nom de l'instance du javabean
propriete : nom d'une propriété du javabean

Définition d'une valeur

```
1 <jsp:set Property name="nom" property="propriete" value="valeur" />
```

accès à la valeur

```
1 <jsp:getProperty name="nom" property="propriete" />
```

3. passage à d'autres pages

transfert vers une autre page

```
1 <jsp:forward page="nouvelle" >
```

inclusion d'une autre page

```
1  <jsp:include page="incluse">
```

4. définition de paramètres

Paramètres accessibles dans les pages appelées

```
1  <jsp:param name="nom" value="valeur" />
2
```

5. plugin

Création d'une applet

<jsp:plugin type="applet" code="MonApplet.class" codebase="applets">

```
1  <jsp:plugin type="applet" code="MonApplet.class" codebase="applets"
2     jreversion="1.1" width="200" height="200" >
3    <jsp:params>
4       <jsp:param name="couleur" value="eeeeee" />
5    </jsp:params>
6  </jsp:plugin>
```

Paramètres obligatoires

- code : nom de la classe
- codebase : chemin du répertoire contenant la classe
- type : applet ou bean

Paramètres optionnels

- align
- height
- width
- ...

Passage de paramètres

```
1    <jsp:params>
2       <jsp:param name="var1" value="val1" />
3       <jsp:param name="var2" value="val2" />
4       ....
5    </jsp:params>
```