

Plan

Partie 1: Rappel

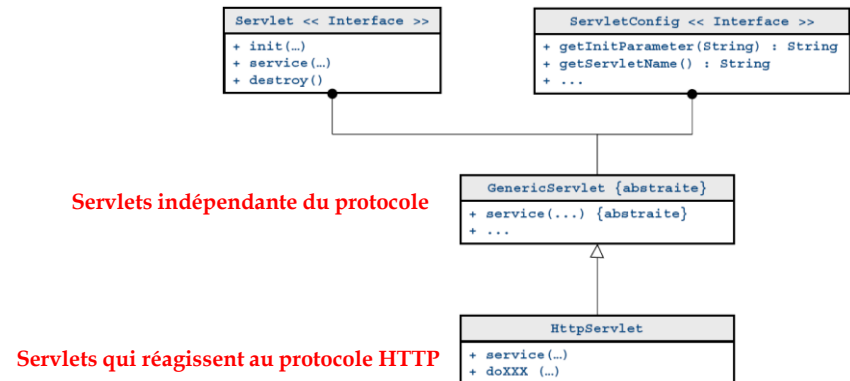
- Chapitre 1 : Plateforme JEE - Introduction
- Chapitre 2 : Servlets
- Chapitre 3 : JSP

Partie 2:

- Chapitre 3 : Java Bean
- Chapitre 4 : Le modèle DAO
- Chapitre 5 : Framework MVC
- Chapitre 6 : Persistance en Java : EJB et JPA

Servlets

☞ Servlets, GenericServlet et HttpServlet



Servlets

👉 HTTPServlet

- Une Servlet est un composant J2EE fonctionnant du côté serveur.
- Permet de développer des pages web dynamiques côté serveurs (comme PHP, ASP)
- Ils s'exécutent dans un conteneurs web comme Tomcat
- Permet l'extension des fonctions du serveur web
- Certaines catégorie de servlets permettent la gestion des requêtes HTTP et de fournir au client une réponse HTTP



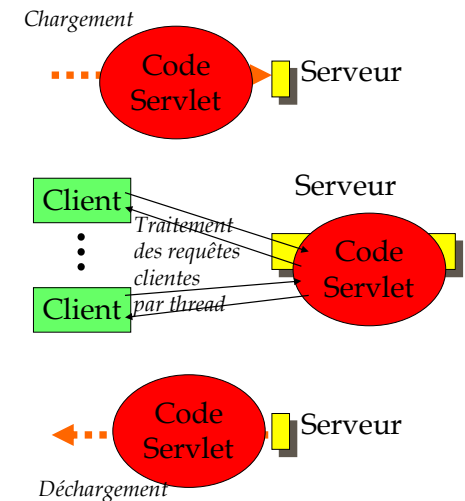
Servlets

👉 Cycle de vie d'une servlet

• **init()** : initialisation de la servlet
chargement du code.
Souvent effectué lors de la première
requête cliente (doGet, doPost)
Allocation d'un pool de threads

- **doGet ()** : Traitement des requêtes HTTP GET
- **doPost ()** : Traitement des requêtes HTTP POST

• **destroy ()** : destruction de la servlet
par le serveur



Servlets

👉 Structure d'une servlet

```
public class HelloWorld extends HttpServlet {
    private int count;
    private String message;

    public void init(ServletConfig config) throws ServletException {
        super.init(config);

        .... // Traitement exécuté au chargement de la servlet
        count = 0;
        message = 0;
    }

    public void destroy() {
        .... // Traitement exécuté à la destruction de la servlet par le serveur
        message=null;
    }
}
```

Servlets

👉 Structure d'une servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        .... // Traitement pour les requêtes de type GET
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        .... // requêtes de type POST
    }
}
```

Objet requête **Objet réponse**



Servlets

HttpServletRequest

- `String getMethod()` : Récupère la méthode HTTP utilisée par le client
- `String getHeader(String Key)` : Récupère la valeur de l'attribut Key de l'en-tête
- `String getRemoteHost()` : Récupère le nom de domaine du client
- `String getRemoteAddr()` : Récupère l'adresse IP du client
- `String getParameter(String Key)` : Récupère la valeur du paramètre Key (clé) d'un formulaire. Lorsque plusieurs valeurs sont présentes, la première est retournée
- `String[] getParameterValues(String Key)` : Récupère les valeurs correspondant au paramètre Key (clé) d'un formulaire, c'est-à-dire dans le cas d'une sélection multiple (cases à cocher, listes à choix multiples) les valeurs de toutes les entités sélectionnées
- `Enumeration getParameterNames()` : Retourne un objet *Enumeration* contenant la liste des noms des paramètres passés à la requête
- `String getServerName()` : Récupère le nom du serveur
- `String getServerPort()` : Récupère le numéro de port du serveur

Servlets

HttpServletResponse

- `String setStatus(int StatusCode)` : Définit le code de retour de la réponse
- `void setHeader(String Nom, String Valeur)` : Définit une paire clé/valeur dans les en-têtes
- `void setContentType(String type)` : Définit le type MIME de la réponse HTTP, c'est-à-dire le type de données envoyées au navigateur
- `void setContentLength(int len)` : Définit la taille de la réponse
- `PrintWriter getWriter()` : Retourne un objet *PrintWriter* permettant d'envoyer du texte au navigateur client. Il se charge de convertir au format approprié les caractères Unicode utilisés par Java
- `ServletOutputStream getOutputStream()` : Définit un flot de données à envoyer au client, par l'intermédiaire d'un objet *ServletOutputStream*, dérivé de la classe *java.io.OutputStream*
- `void sendRedirect(String location)` : Permet de rediriger le client vers l'URL location

Servlets

👉 Déploiement des servlets

Descripteur de déploiement

```
<web-app>
<servlet>
  <servlet-name>helloservlet</servlet-name>
  <servlet-class>test.servlet.HelloWorldExample</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>helloservlet</servlet-name>
  <url-pattern>/hello</url-pattern>
</servlet-mapping>
</web-app>
```

Balise principale → `<web-app>`

Balise de description d'une Servlet → `<servlet>`

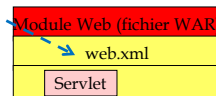
Nom de la Servlet → `<servlet-name>`

Classe de la Servlet → `<servlet-class>`

Définition du chemin virtuel → `<servlet-mapping>`

Nom de la servlet considéré → `<servlet-name>` (dans mapping)

Définition du chemin virtuel → `<url-pattern>`



Une application web peut contenir plusieurs servlets

Servlets

👉 Configuration des servlets à partir de Java 6 via les annotations

- `@WebServlet` permet de marquer une classe comme servlet.

```
@WebServlet(name="simpleservlet", urlPatterns="/myservlet", "/simpleservlet")
```

```
public class SimpleServlet extends HttpServlet { ... }
```

```
@WebServlet(name="simpleservlet", urlPatterns="/myservlet", "/simpleservlet", initParams={
  @WebInitParam(name="param1", value="value1"),
  @WebInitParam(name="param2", value="value2")
})
```

```
public class MyServlet extends HttpServlet { ... }
```

- `@WebInitParam` cette annotation est utilisée pour préciser des paramètres d'initialisation aux Servlets ou aux Filtres


Servlets

☞ Persistances d'instances

Est-ce que les servlets sont instanciées à chaque requêtes ?



- Entre chaque requête du client, les Servlets persistent sous forme d'instances d'objet.
- L'unique instance traite toutes les requêtes.

- 
- ✓ Rapidité : pas de surcoût en temps lié à la création d'objet à chaque requête.
 - ✓ L'espace mémoire réservé reste faible.
 - ✓ La persistance, c'est-à-dire la possibilité de conserver l'état de la servlet, est facilitée.

Servlets

☞ Persistances d'instances

Exemple illustratif

```
public class SimpleCounterServlet extends HttpServlet {  
    private int nbVisite= 0;  
    protected void doGet(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException {  
        PrintWriter out = res.getWriter();  
        nbVisite ++;  
        out.println("Depuis son chargement, c'est le" + nVisite + " eme visite à cette  
servlet " ); }  
}
```

Servlets

☞ Persistances d'instances

✓ Possibilité d'utiliser des paramètres d'initialisation définis dans le fichier de déploiement [web.xml](#)

```
<web-app><servlet>
  <servlet-name>helloservlet</servlet-name>
  <servlet-class>test.SimpleCounterServlet </servlet-class>
  <init-param>
    <param-name>nb_visite_initiale</param-name>
    <param-value>10</param-value>
    <description>Valeur initiale</description>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>helloservlet</servlet-name>
  <url-pattern>/hello</url-pattern>
</servlet-mapping> </web-app>
```

web.xml

Servlets

☞ Persistances d'instances

Et de coté servlet...

```
public class SimpleCounterServlet extends HttpServlet {
  private int nb_visite;

  public void init() throws ServletException {
    String initial = this.getInitParameter("nb_visite_initiale");
    try {
      count = Integer.parseInt(initial);
    } catch (NumberFormatException e) { nb_visite = 0; }
  }

  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    PrintWriter out = res.getWriter();
    nb_visite ++;
    out.println("Depuis son chargement, c'est votre " + nb_visite + " eme visite à
    cette servlet " ); }
}
```

Servlets

☞ Persistances d'instances

...Autre remarques ?



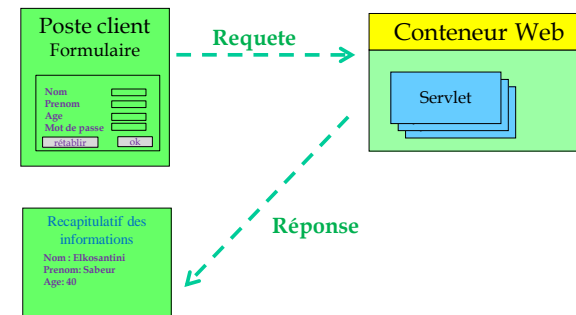
- ✓ A chaque rechargement d'une Servlet :
 - Création d'une nouvelle instance.
 - destruction de l'ancienne servlet.

- ✓ Pas de constructeurs. Toute initialisation se fait dans la méthode `init()`.

- ✓ `init ()` ne possède pas de paramètres.

Servlets

☞ Servlet et les formulaires html



Servlets

Poste client
Formulaire

Nom

Prenom

Age

Mot de passe

```
<FORM Method="POST" Action="MaServlet">
Nom : <INPUT type="text" size=20 name="nom"><BR>
Prénom : <INPUT type="text" size=20 name="prenom"><BR>
Age : <INPUT type="text" size=2 name="age"><BR>
Age : <INPUT type="password" size=20 name="pwd"><BR>

<INPUT type="reset" value="rétablir"> </FORM>
<INPUT type="submit" value="Envoyer"> </FORM>
```

Servlets


```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class UserInfo extends HttpServlet {

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

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<HTML>\n<BODY>\n" +
            "<H1>Recapitulatif des informations</H1>\n" +
            "<UL>\n" +
            "<LI>Nom: " +
            request.getParameter("Nom") + "\n" +
            "<LI>Prenom: " + request.getParameter("Prenom") +
            "\n" +
            "<LI>Age: " +
            request.getParameter("Age") +
            "\n" + "</UL>\n" + "</BODY></HTML>");
    }
}
```

Servlet

Servlets

Servlet et les formulaires html

- `String getParameter(String Key)` : renvoie la valeur du champ Key du formulaire
- `String[] getParameterValues(String Key)` : Récupère les valeurs correspondant champ Key d'un formulaire qui peut avoir plusieurs valeurs (cases à cocher, listes à choix multiples).
- Enumeration `getParameterNames()` : renvoie l'ensemble des noms des champs du formulaire passé à la servlet.
- Object `getAttribute(String Key)` : renvoie la valeur de l'attribut Key sous forme d'un objet de type Object

Servlets

Servlet et les formulaires html

Quelle est la différence entre `getAttribute` et `getParameter` ?



- `getParameter(String Key)` renvoie le contenu d'une variable à partir d'un formulaire.



- Object `getAttribute(String Key)` : renvoie la valeur de l'attribut Key sous forme d'un objet de type Object



Servlets

👉 Exercices

1. Ecrire une servlet qui permet de récupérer le login et le mot de passe à partir d'un formulaire et les affiche dans une autre page.
2. *Amélioration 1*: ajouter le sex au formulaire et améliorer le message affiché :
"Bonjour Mr (ou Mme)"
3. *Amélioration 2*: ajouter la liste des compétences au formulaire et améliorer le message affiché :
"Bonjour Mr (ou Mme) Vos compétences sont : ... et ..."
4. Remplacer la page html par une servlet.
5. A cette dernière version, ajouter un lien qui permet à un nouveau visiteur de d'accéder en tant qu'invité. Un mot de passe lui sera affiché. Cette partie sera traitée par une autre servlet.

Servlets

👉 Exercices

1. Reprendre l'exercice précédent en ajoutant les bases de données. Avant d'afficher les messages, la servlet doit vérifier si le login existe dans la base de données.

Servlets

👉 Exercice

Écrire une application web comportant une première page HTML permettant de choisir un affichage des pièces par catégorie.

1. Écrire une page HTML présentant la compagnie et ayant un lien vers un premier servlet.
2. La première servlet génère une page HTML permettant à l'utilisateur de choisir une catégorie à afficher. Il affiche la liste des noms de catégories de la table catégories et permet de faire un lien vers la deuxième servlet en lui retournant le numéro de catégorie sélectionné par l'utilisateur.
3. La troisième servlet reçoit un numéro de catégorie en paramètre HTTP une troisième page HTML est générée. Celle-ci présente une entête contenant la catégorie et un tableau contenant pour toutes les pièces de cette catégorie, le numéro, la description, le nom du fabricant et le prix de vente. Bonus : une image de la pièce en fonction de l'URL.

Servlets

👉 Première servlet

```
public class Catégorie extends HttpServlet {
    Connection conn=null;

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

            Class.forName ("org.postgresql.Driver");
            conn = DriverManager.getConnection ("jdbc:postgresql://127.0.0.1:5432/test",
            "postgres", "postgresql");
        }
        catch (Exception c) { System.out.println ("problème SQL"+c); }
    }
}
```

Servlets

👉 Première servlet

```
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    try{
        Statement stmt = conn.createStatement();
        String req= "SELECT * FROM categorie ";
        ResultSet rs = stmt.executeQuery(req);
        while(rs.next())
            out.println("<a
href=\"./produit?idc="+rs.getString("idcat")+"\">"+rs.getString("nom")+"</a>
");
    }
    catch (Exception c) { System.out.println ("problème SQL"+c); }
}
```

Servlets

👉 Première servlet

```
<context-param>
    <init-param>
        <param-name>db-driver</param-name>
        <param-value>org.postgresql.Driver</param-value>
    </init-param>

    <init-param>
        <param-name>db-url</param-name>
        <param-value>jdbc:postgresql://127.0.0.1:5432/test</param-value>
    </init-param>

    <init-param>
        <param-name>db-login</param-name>
        <param-value>postgres</param-value>
    </init-param>

    <init-param>
        <param-name>db-passwd</param-name>
        <param-value>postgres</param-value>
    </init-param>
</context-param>
```

Utiliser web.xml pour les paramètres de la base de données.

Servlets

👉 Première servlet

```
public class Categorie extends HttpServlet {
    Connection conn=null;

    public void init(ServletConfig config) throws ServletException {
        try {
            String driver= config.getInitParameter("db-driver");
            String url= config.getInitParameter("db-url");
            String login= config.getInitParameter("db-Login");
            String mdp= config.getInitParameter("db-passwd");

            Class.forName (driver);}
            conn = DriverManager.getConnection (url, login, mdp);
        }
        catch (Exception c) { System.out.println ("problème SQL"+c); }
    }
}
```

La servlet devient alors ...

Servlets

👉 Première servlet

```
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    try{
        Statement stmt = conn.createStatement();

        String req= "SELECT * FROM categorie ";
        ResultSet rs = stmt.executeQuery(req);
        while(rs.next())
            out.println("<a
href='./produit?idc="+rs.getString("idcat")+"\">"+rs.getString("nom")+"</a>
");
    }
    catch (Exception c) { System.out.println ("problème SQL"+c); }
    }
}
```

Servlets

👉 Deuxième servlet

```
public class produit extends HttpServlet {
    Connection conn=null;

    public void init(ServletConfig config) throws ServletException {
        try {
            String driver= config.getInitParameter("db-driver");
            String url= config.getInitParameter("db-url");
            String login= config.getInitParameter("db-Login");
            String mdp= config.getInitParameter("db-passwd");

            Class.forName (driver);}
            conn = DriverManager.getConnection (url, login, mdp);
        }
        catch (Exception c) { System.out.println ("problème SQL"+c); }
    }
}
```

Servlets

👉 Deuxième servlet

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

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    try{
        Statement stmt = conn.createStatement();
        String req= "SELECT * FROM produit where
        idc="+request.getParameter("idc");
        ResultSet rs = stmt.executeQuery(req);
        while(rs.next())
            out.println(rs.getString("nomp"));
        }
        catch (SQLException c) { System.out.println ("problème SQL"+c); }}
```

Servlets

Redirection

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {
    public void doGet( HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        response.sendRedirect("http://www.isima.rnu.tn"); }
    public void doPost( HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        DoGet(request,response);}
}
```

Redirection vers un site

Servlets

Forward et Include

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

    request.getRequestDispatcher("/index.html").forward(request,response);
}
```

La redirection est cachée pour l'utilisateur. La réponse et la requête sont aussi redirigés

Servlets

👉 **Forward et Include**

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

    response.setContentType("text/html");
    request.getRequestDispatcher("/entete.html").include(request,response);
    request.getRequestDispatcher("/index.html").include(request,response);

}
```

L'inclusion permet d'inclure plusieurs fois des ressources

Servlets

👉 **Les sessions**

- HTTP est un protocole sans état et ne permet pas mémoriser des variables dans une requête.

Comment faire alors ?



- ✓ Les champs cachés.
- ✓ Réécriture des URL (contenant les informations mémorisées)
- ✓ Les sessions

Servlets

👉 Les sessions

- Une session est une suite d'interactions entre un client et un serveur Web
- On peut utiliser les sessions pour mémoriser les actions d'un utilisateur unique (achat en ligne)
- Mécanisme très puissant permettant de stocker des objets et non de simples chaînes de caractères.
- L'API *HttpSession* fournit des fonctions pour gérer les sessions.

Servlets

👉 HttpSession

- Méthode de création (de *HttpServletRequest*) :
 - ✓ *HttpSession getSession()* : retourne la session associée à l'utilisateur.
- Gestion d'association (de *HttpSession*) :
 - ✓ *Enumeration getAttributeNames()* : retourne les noms de tous les attributs.
 - ✓ *Object getAttribute(String name)* : retourne l'objet associé au nom.
 - ✓ *setAttribute(String na, Object va)* : donne la valeur va à l'attribut na.
 - ✓ *removeAttribute(String na)* : supprime l'attribut de nom na.
- Destruction (de *HttpSession*) :
 - ✓ *logout()* : termine la session.

Servlets

👉 HttpSession

```
public class HttpSessionServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/plain");
        PrintWriter out = res.getWriter();
        HttpSession session = req.getSession();
        String nom = (String)session.getAttribute("nom");
        out.println("Bonjour Mr. " + nom + ".");
    }
}
```

Servlets

👉 HttpSession

```
public class HttpSessionServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/plain");
        PrintWriter out = res.getWriter();
        HttpSession session = req.getSession();
        Integer count =
            (Integer)session.getAttribute("count");
        if (count == null)
            count = new Integer(1);
        else
            count = new Integer(count.intValue() + 1);
        session.setAttribute("count", count);
        out.println("Vous avez visité cette page " + count + " fois.");
    }
}
```

Problème : un espace nécessaire suffisant dans le serveur

Servlets

👉 Les sessions

- Il est possible de fixer la durée d'une session par application (en minutes)

```
<web-app >
<servlet>
  <servlet-name>helloservlet</servlet-name>
  <servlet-class>test.SimpleCounterServlet </servlet-class>
  ...
  <session-config>
    <session-timeout>30</session-timeout>
  </session-config>
  ...
</servlet>
<servlet-mapping>
  <servlet-name>helloservlet</servlet-name>
  <url-pattern>/hello</url-pattern>
</servlet-mapping> </web-app>
```

Servlets

👉 Cookies

- Les cookies représentent un moyen simple de stocker temporairement des informations chez un client.
- Concrètement : il s'agit de fichiers texte stockés sur le disque dur du client.
- L'envoi du cookie vers le navigateur du client se fait grâce à la méthode de HttpServletResponse:

```
void AddCookie(Cookie cookie)
```

Exemple :

```
Cookie MonCookie = new Cookie("nom", "valeur");
response.addCookie(MonCookie);
```

Servlets

👉 Cookies

- Récupération des cookies du client:

`Cookie[] getCookies()`

- Récupération de la valeur d'un cookie :

`String Valeur = Cookie.getValue()`

- Inconvénients des cookies :

- ✓ Les navigateurs ne les acceptent pas toujours.
- ✓ L'utilisateur peut configurer son navigateur pour qu'il refuse ou pas les cookies.
- ✓ Le nombre et la taille des cookies peuvent être limités par le navigateur.

Servlets

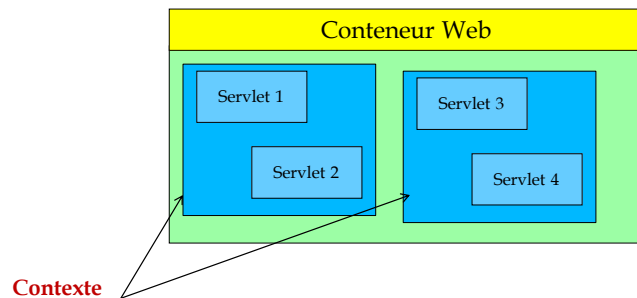
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.net.*;

public class AfficheMonCookie extends HttpServlet {
    public void doGet (HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {
        Cookie[] cookies = request.getCookies();
        for(i=0; i < cookies.length; i++) {
            Cookie MonCookie = cookie[i];
            if (MonCookie.getName().equals(" couleur")) {
                String Valeur = cookie[i].getValue();
            }
        }
        // écriture de la réponse
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><head>");
        out.println("<title>Mon Cookie</title>");
        out.println("</head><body bgcolor ="+Valeur+">");
        out.println("Voici mon premier test");
        out.println("</body></html>"); } }
```

Servlets

Le contexte des servlets

- Un contexte constitue une vue sur le fonctionnement d'une même application web.
- Il est possible de partager des informations entre servlets du même contexte grâce à l'interface *ServletContext*



Servlets

Le ServletContext

- Objet permettant au servlet de communiquer avec le conteneur
- Obtenu avec :
 - `Servlet.getServletContext()`
- Les principales méthodes de ServletContext
 - ✓ `Object getAttribute(String name)`
 - ✓ Retourne un attribut du contexte
 - ✓ `Void setAttribute(String name, Object value)`
 - ✓ Ajoute ou remplace un objet dans le contexte
 - ✓ `String getInitParameter(String name)`
 - ✓ Retourne un paramètre d'initialisation de l'application

Servlets

Le ServletContext

```
out.println("Context 1 "+getServletContext().getInitParameter("context1"));  
out.println("Context 2 "+getServletContext().getInitParameter("context2"));
```

```
<context-param>  
  <param-name>context1</param-name>  
  <param-value>ma valeur 1</param-value>  
</context-param>  
<context-param>  
  <param-name>context2</param-name>  
  <param-value>ma valeur 2</param-value>  
</context-param>
```

Servlet WebContextServlet

Context 1 ma valeur 1 Context 2 ma valeur 2

Servlets

Autres champs du fichier web.xml

- Il est possible de définir les pages à afficher :

- ✓ En fonction d'erreurs http
- ✓ En fonction d'exceptions java

```
<error-page>  
  <exception-type>  
    cours.event.EventException  
  </exception-type>  
  <location>/erreur.html</location>  
</error-page>
```

Pour une
exception
java

```
<error-page>  
  <error-code>404</error-code>  
  <location>/404.html</location>  
</error-page>
```

Pour une
erreur Http

Plan

Partie 1: Rappel

- Chapitre 1 : Plateforme JEE - Introduction
- Chapitre 2 : Servlets
- Chapitre 3 : JSP

Partie 2:

- Chapitre 4 : Java Bean
- Chapitre 5 : Le modèle DAO
- Chapitre 6 : Framework MVC : struts
- Chapitre 7 : Persistance en Java : EJB et JPA