CHAPITRE II : Les Servlets

2.1 DEFINITION :

Les servlets sont des applications fonctionnant du côté serveur au même titre que les langages de script côté serveur tels que ASP ou bien PHP. Les servlets permettent donc de gérer des requêtes HTTP et de fournir au client une réponse HTTP dynamique (donc de créer des pages web dynamiques).

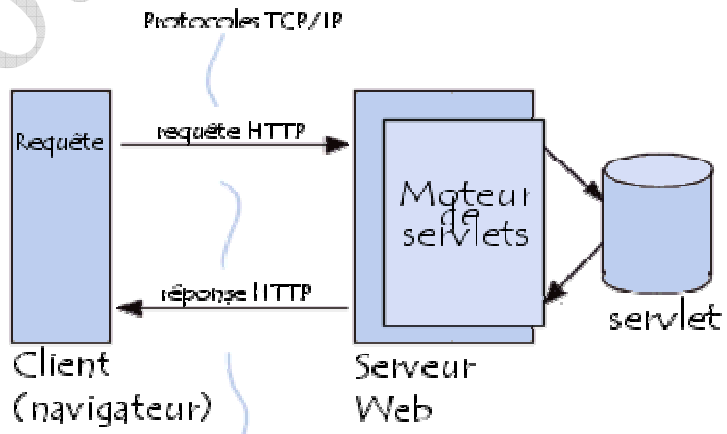
Une servlet est généralement une classe Java qui permet de créer dynamiquement des données au sein d'un serveur HTTP. Ces données sont le plus généralement présentées au format HTML, mais elles peuvent également l'être au format XML ou tout autre format destiné aux navigateurs web. Les servlets utilisent l'API Java Servlet (package javax.servlet).

Une servlet s'exécute dynamiquement sur le serveur web et permet l'extension des fonctions de ce dernier, typiquement : accès à des bases de données, transactions d'e-commerce, etc. Une servlet peut être chargée automatiquement lors du démarrage du serveur web ou lors de la première requête du client. Une fois chargées, les servlets restent actives dans l'attente d'autres requêtes du client.

L'utilisation de servlets se fait par le biais d'un *conteneur de servlets* (framework) côté serveur (ex : TomCat). Celui-ci constitue l'environnement d'exécution de la servlet. L'API définit les relations entre le conteneur et la servlet. Le conteneur reçoit la requête du client, et sélectionne la servlet qui aura à la traiter. Le conteneur fournit également tout un ensemble de services standards pour simplifier la gestion des requêtes et des sessions.

2.2 INFRASTRUCTURE D'UN SERVLET :

Les servlets ont de nombreux avantages par rapport aux autres technologies côté serveur. Tout d'abord, étant donné qu'il s'agit d'une technologie Java, les servlets fournissent un moyen d'améliorer les serveurs web sur n'importe quelle plateforme (portabilité) , d'autant plus que les servlets sont indépendantes du serveur web (contrairement aux modules , exemple apache). En effet, les servlets s'exécutent dans un moteur de servlet utilisé pour établir le lien entre la servlet et le serveur web. Ainsi le programmeur n'a pas à se soucier de détails techniques tels que la connexion au réseau, la mise en forme de la réponse HTTP, ...



D'autre part les servlets sont beaucoup plus performantes que les scripts, car il s'agit de pseudo-code, chargé automatiquement lors du démarrage du serveur ou bien lors de la connexion du premier client. Les servlets sont donc actives (résidentes en mémoire) et prêtes à traiter les demandes des clients grâce à des threads, tandis qu'avec les langages de script traditionnels un nouveau processus est créé pour chaque requête HTTP. Cela permet donc une charge moins importante au niveau du processeur du serveur (d'autant plus qu'un système de cache peut permettre de stocker les calculs déjà accomplis), ainsi que de prendre une place moins importante en mémoire.

2.3 EXEMPLE DE SERVLET :

Voici un exemple simple de servlet dont le but est d'afficher du texte sur le navigateur du client :

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class PremiereServlet extends HttpServlet {

  public void init() {
  }
  public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {

        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("<HTML>");
        out.println("<HEAD><TITLE> Titre </TITLE></HEAD>");
        out.println("<BODY>");
        out.println("Ma première servlet");
        out.println("</BODY>");
        out.println("</HTML>");
        out.close();
  }
}
```

La première étape consiste à importer les packages nécessaires à la création de la servlet, il faut donc importer *javax.servlet*, *javax.servlet.http* et *javax.io*

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

Afin de mettre en place l'interface *Servlet* nécessaire au conteneur de servlet, il existe plusieurs possibilités :

- Définir manuellement chaque méthode
- Dériver la classe *GenericServlet* et redéfinir les méthodes dont on a besoin
- Dériver la classe *HttpServlet* et redéfinir les méthodes dont on a besoin

Dans la servlet ci-dessus, la classe *HttpServlet* a été étendue

```
public class PremiereServlet extends HttpServlet {
}
```

Lorsque la servlet est instanciée, il peut être intéressant d'effectuer des opérations qui seront utiles tout au long du cycle de vie de la servlet (se connecter à une base de données, ouvrir un fichier, ...). Pour ce faire, il s'agit de surcharger la méthode *init()* de la servlet.

```
public void init() {}
```

A chaque requête, la méthode *service()* est invoquée. Celle-ci détermine le type de requête dont il s'agit, puis transmet la requête et la réponse à la méthode adéquate (*doGet()* ou *doPost*). dans notre cas, on ne s'intéresse qu'à la méthode GET, c'est la raison pour laquelle la méthode *doGet()* a été surchargée

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

L'objet *HttpServletRequest* permet de connaître les éventuels paramètres passés à la servlet (dans le cas d'un formulaire HTML par exemple), mais l'exemple ci-dessus n'en a pas l'utilité.
Par contre l'objet *HttpServletResponse* permet de renvoyer une page à l'utilisateur. La première étape consiste à définir le type de données qui vont être envoyées au client. Généralement il s'agit d'une page HTML, la méthode *setContentType()* de l'objet *HttpServletResponse* doit donc prendre comme paramètre le type MIME associé au format HTML (*text/html*) :

```
res.setContentType("text/html");
```

Ensuite la création d'un objet *PrintWriter* grâce à la méthode *getWriter()* de l'objet *HttpServletResponse* permet d'envoyer du texte formatté au navigateur (pour envoyer un flot de données, il faudrait utiliser la méthode *getOutputStream()*)

```
PrintWriter out = res.getWriter();
```

Enfin il faut utiliser la méthode *println()* de l'objet *PrintWriter* afin d'envoyer les données textuelles au navigateur, puis fermer l'objet *PrintWriter* lorsqu'il n'est plus utile avec sa méthode *close()*

```
        out.println("<HTML>");
        out.println("<HEAD><TITLE> Titre </TITLE></HEAD>");
        out.println("<BODY>");
        out.println("Ma première servlet");
        out.println("</BODY>");
        out.println("</HTML>");
        out.close();
```