1. **Servlets** : composant coté serveur, exécuté dans un container de servlets(comme tomcat) lui-même s’exécutant via un serveur web (comme apache).
   1. Dériver de javax.servelet.http.HttpServlet
   2. Les classes javax.servlet.\* ne sont pas partie du Java standard, il faut rajouter la lib de tomcat.
   3. HttpServlet : Hérite de GenericServlet : classe définissant une servlet utilisant le protocole http
   4. RequestDispatcher : Définition d'un objet qui permet le renvoi d'une requête vers une autre ressource du serveur (une autre servlet, une JSP ...), La méthode forward() fait suivre une requête d'une servlet à une autre sur le même serveur. Des attributs peuvent être passés grâce à la méthode setAttribute().
   5. HttpServletRequest : Hérite de ServletRequest : définit un objet contenant une requête selon le protocole http
   6. HttpServletResponse : Hérite de ServletResponse : définit un objet contenant la réponse de la servlet selon le protocole http
   7. Cycle de vie d’une servlet :

* la méthode *init()*
* la méthode *service()*
* la méthode *getServletConfig()*
* la méthode *getServletInfo()*
* la méthode *destroy()*
  1. 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.
  2. 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*). Il ne faut pas redéfinir la méthode service()
  3. web.xml : utilisé par Tomcat pour faire une relation entre une requête HTTP et une servlet.
  4. le conteneur crée un objet **ServletContext** qui sera partagé par toutes nos servlets, La méthode getServletContext() de l'objet ServletConfig nous retourne donc cet objet. ais comme les concepteurs de la technologie JEE sont des gens super gentils, ils ont fait en sorte que nos servlets puissent invoquer une méthode getServletContext() sans passer par l'objet ServletConfig. (voir exemple servletContext.java)
  5. getServletContext().getRequestDispatcher("/formulaire2.jsp").forward(request,response) : permet d’appeler la page formulaire2.jsp.
  6. Il y a un objet **ServletConfig** par servlet et un objet **ServletContext** par application.
  7. La méthode setAttribute de **ServletContext sert à passer des attributs accessibles par toutes l’application**
  8. **le moyen d'initialiser des objets en dehors de toute servlet, on utilise  (ServletContextListener), voir le répertoire ServletContextListener.**

1. **web.xml** : La servlet est tout d'abord définie dans l'application via l'élément <servlet></servlet> . Nous trouvons deux éléments dans ce dernier :

* <servlet-class></servlet-class> : code réel de la servlet ;
* <servlet-name></servlet-name> : nom interne à l'application.
* <init-param> </init-param> : permet de déclarer des variable avec des valeurs qui peuvent être récupéré dans la servlet (getInitParameter(name) de l’objet **ServletConfig**, voir l’exemple : initParam.java)

Ensuite, nous trouvons <servlet-mapping></servlet-mapping> contenant les informations de paramétrage client :

* <servlet-name></servlet-name> : nom interne à l'application ;
* <url-pattern></url-pattern> : nom qui apparaît côté client.