**Commandes en Java - EE**

**Lier Eclipse au server TomCat :**

window > Preferences > Server > Runtime Environment > Add

choisir la version de TomCat téléchargée

indiquer l'emplacement du dossier de TomCat (à la racine du lecteur C:)

> Finish

**Pour avoir accès au serveur local depuis le navigateur :** localhost:8080/NomProjet/url-pattern

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Symbole | Signification | Code | Utilisation | | |
| Servlet |  |  |  | |  |
| web.xml | fichier de configuration indispensable faisant le lien entre servlet et url  ----------  partie à répéter dans le même fichier à la suite si plusieurs servlets  ----------- | <?xml version="1.0" encoding="UTF-8"?>  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd" version="3.0">  <servlet>  <servlet-name>maServlet</servlet-name>  <servlet-class>monPackage.maServlet</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>maServlet</servlet-name>  <url-pattern>/monUrl</url-pattern>  </servlet-mapping>  </web-app> | | | fichier à placer dans le dossier WEB-INF  Remarque : si plusieurs URL sont gérées par la même Servlet :  <url-pattern>/monUrlA</url-pattern>  <url-pattern>/monUrlB</url-pattern>  … |
| doGet( )  doPost( ) | méthode permettant de traiter la requête et de renvoyer une page ou un élément en réponse | public void doGet(  this.getServletContext().getRequestDispatcher("/WEB-INF/maPage.jsp").forward(request, response); | | | envoi en réponse à la requête une page (vue) stockée dans WEB-INF  Rq: .getRequestDispatcher(" ") se base sur le contexte de l'appli, cad le dossier de l'appli ou son nom de domaine si elle est déployée. Le forwarding ne peut se faire en dehors de l'appli. |
| .setAtribute( , ) | permet de transmettre un attribut et sa valeur à la vue concernée par la requête | public void doGet(…) {  request.setAtribute("nomAtribut", valeurAtribut);  this.getServletContext…  } |  | | Remarque : pour récupérer cette valeur : voir JSP |
| .sendRedirect( ) | redirige vers une autre page (du même site ou non) | response.sendRedirect(request.getContextPath() + "/WEB-INF/monAutrePage.jsp");  response.sendRedirect(request.getContextPath() + "**/url/deMa/servlet**"); | response.sendRedirect("http://autreSite.com"); | | Rq: la redirection est transparente (cad la nouvelle adresse s'affiche dans le navigateur du client) contrairement au forwarding (interne au site) |
| .getContextPath() | méthode de l'objet request permettant d'obtenir le contexte de l'appli, cad  monProjet/ ou www.monSite.com/ | response.sendRedirect(request.getContextPath() + /WEB-INF/monAutrePage.jsp); |  | | ici la redirection peut se faire dans un autre contexte que celui de l'appli. Pour rester à l'intérieur de manière automatique sans avoir à modifier toutes les adresses une fois l'appli déployée, on utilise cette méthode |
| .getServletConfig() | récupère les paramètres de la serlet définis dans le fichier *web.xml* | public void doGet(…) … {  String auteur = this.getServletConfig().getInitParameter("auteur"); | *web.xml :*  <init-param>  <param-name>auteur</param-name>  <param-value>maValeur</param-value>  </init-param> | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  | Paramètres URL |  |  | |  |
| url? = & = | insère des paramètres dans une adresse URL | monURL?nomParametre=valeur  monURL?nomParametre1=valeur1&nomParametre2=valeur2&… |  | |  |
| .getParametre(" ") | récupère la valeur d'un paramètre placé dans l'URL | String monParametre = request.getParameter("nomParametre"); | try {  montant = Double.parseDouble( request.getParameter( "montantCommande" ) );  } catch ( NumberFormatException e ) {  */\* valeur si n'est pas un nombre correct \*/*  montant = -1;  } | | récupère la valeur d'un parameter et le caste en nombre en capturant l'erreur si ce n'est pas un vrai nombre |
|  | Session |  |  | |  |
| HttpSession | créé un objet de type HttpSession qui sera conservé durant toute la visite de l'utilisateur ainsi que les informations qu'on lui aura associées | HttpSession maSession = request.getSession();  maSession.setAttribute("nomAttribut", monObjet); | *pour récupérer l'objet depuis une JSP :*  ${ sessionScope.nomAttribut } | | ATTENTION : dans la Servlet il n'y a pas besoin d'associer l'objet maSession à l'objet request  Dans la JSP le sessionScope recherchera l'attribut nomAtt uniquement dans la session de l'utilisateur associé à la requête en cours (et non pas dans toutes les sessions existantes)  Ces variables de session peuvent être récupérées dans une JSP via l'objet sessionScope *(voir JSP > EL)* |
| .getId() | retourne l'identifiant permettant de déterminer à qui appartient la session | maSession.getId(); |  | | Rq: cet identifiant est créé avec l'objet HttpSession lors de la 1re requête du client. Il est transmis dans la réponse et enregistré par le navigateur. A la 2e requête, l'Id est fourni par le client et .getSession() retrouve la session en question. |
| .invalidate() | méthode permettant d'effacer volontairement la session en question | maSession.invalidate(); |  | | Rq : nécessaire pour un logout volontaire de l'utilisateur, mais sinon les sessions sont automatiquement effacées par le serveur après un certain temps d'inactivité du visiteur |
|  |  |  |  | |  |
|  | Cookies |  |  | |  |
| new Cookie( , ) | création d'un nouveau cookie | Cookie monCookie = new Cookie("nomCookie", maVariable); |  | | Rq : une seule variable stockée par cookie ! |
| .setMaxAge( ) | définie une date d'expiration pour le cookie (en seconde) | monCookie.setMaxAge(60 \* 60 \* 24 \* 30); | nb positif => date expiration en sec.  0 => suppr du cookie  nb négatif => suppr à la fin de la session | | Rq : par défaut, la date d'expiration d'un cookie est la fin de la session |
| response.addCookie( ) | enregistre le cookie placé en paramètre sur la machine de l'utilisateur | response.addCookie(monCookie); |  | | ATTENTION : à utiliser dans doPost() de la Servlet ! |
| request.getCookies() | récupère tous les cookies liés à notre site et enregistrés sur la machine de l'utilisateur | Cookie[] mesCookies = request.getCookies(); | *Trouver le cookie qui nous intéresse :*  if (mesCookies != null) {  for (Cookie cookieTmp : mesCookies) {  if (cookieTmp.getName().equals("nomAttribut")) {  String maValeur = cookieTmp.getValue();  *ou*  request.setAttribute("nomAttribut", cookieTmp.getValue());  }  }  } | | ATTENTION : à utiliser dans doGet() de la Servlet ! |
|  |  |  |  | |  |
|  |  |  |  | |  |
| Filtre |  |  |  | |  |
| web.xml |  | <filter>  <filter-name>monFiltre</filter-name>  <filter-class>monPackage.MonFiltre</filter-class>  </filter>  <filter-mapping>  <filter-name>monFiltre</filter-name>  <url-pattern>/monUrl</url-pattern>  </filter-mapping> | | | Rq: si une requête doit passer par plusieurs filtres, l'ordre dans lequel ils se présenteront (FilterChain) est défini par l'ordre dans lequel ils sont déclarés dans le fichier *web.xml* |
|  |  |  | | |  |
| doFilter( , , ) | méthode de traitement appliquée à la requête/reponse | public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain) throws IOException, ServletException {  /\* Cast des objets request et response \*/  HttpServletRequest request = (HttpServletRequest) req;  HttpServletResponse response = (HttpServletResponse) resp; | | | Rq: la classe mère Filtre implémentée n'est pas spécialisée dans les requêtes HTTP. Il faut donc caster les objets request et response à chaque fois ! |
| .doFilter( , ) | méthode à appeler sur l'objet de type FilterChain pour passer la requête/rep au filtre suivant dans la liste ou bien à la servlet dédiée si c'est le dernier filtre | chain.doFilter(request, response); |  | |  |
| UTF-8 | filtre obligatoire à déclarer en premier dans *web.xml* précisant l'encodage en UTF-8 de toutes les données entrantes dans le serveur | public void doFilter( … ) {  HttpServletRequest request = (HttpServletRequest) req;  HttpServletResponse response = (HttpServletResponse) resp;  request.setCharacterEncoding("UTF-8");  chain.doFilter(request, response);  } | | | Rq: possibilité d'utiliser à la place le filtre natif mis à disposition par Tomcat : *voir* *xml-Commandes* |
|  |  |  |  | |  |
| JSP | JSP = Java Server Page |  |  | |  |
| .jsp | les fichiers .jsp sont les vues de notre projet.  mélange de code HTML et Java |  |  | | fichiers à placer dans WEB-INF |
| <% %> | balises de code Java dans une JSP  le code est exécuté mais pas transmis lors de l'envoi de la page HTML en réponse |  |  | |  |
| <%-- --%> | commentaire |  |  | | Rq: contrairement aux commentaires HTML, les commentaires JSP ne seront pas visible sur la page transmise au client. |
| <%! %> | déclaration de variable(s) | <%! String maChaine = "valeur";  public String maMethode() {  instruction(s);  return maValeur;  }  %> |  | |  |
| <%= %> | balises d'expression | <%= "Mon message à afficher" %> | equivalent à :  <% out.println("Mon message à afficher"); %> | |  |
| <%@ %> | directives  doivent être placées au debut de la JSP (sauf directive d'inclusion qui peut être placée n'importe où) |  |  | |  |
| <%@ taglib | charge une bibliothèque personnalisée | <%@ taglib uri="maTagLib.tld" prefix="tagExemple" %> |  | | Rq: utilisé pour inclure la bibliothèque JSTL (voir plus bas) |
| <%@ page | directive de page  définit des informations relatives à la page JSP | <%@ page import="java.util.List, java.util.Date, monPackage.MaClasse" %> |  | |  |
| <%@ include | inclusion d'une autre page (JSP, HTML, …) à l'intérieur de notre JSP | <%@ include file="uneAutreJSP.jsp" %> |  | | Rq: ici le code de la page incluse est "copié" dans la page puis l'ensemble est ensuite compilé  Rq: les fichiers doivent être placés dans le même dossier : ici WEB-INF |
| <jsp:include /> | inclusion d'une autre page JSP à l'intérieur de notre JSP | <jsp:include page="uneAutreJSP.jsp" /> |  | | Rq: contrairement à l'inclusion ci-dessus, ici les deux pages JSP sont d'abord compilées avant d'être incluse l'une dans l'autre |
|  |  |  |  | |  |
|  |  |  |  | |  |
| .getAttribute( ) | récupère la valeur d'un atribut placé sur l'objet request par la Servlet | <p>  <%  type monAtribut = (type) request.getAttribute("nomAtribut");  out.println(monAtribut);  %>  </p> |  | |  |
|  |  |  |  | |  |
| Expression Language (EL) | simplifie le code Java dans une page JSP |  |  | |  |
| ${ } | permet de définir une EL dans une JSP |  | <p>Résultat du calcul : ${ 6 \* 7 } </p>  <p>Bonjour ${ name } </p>  <p>2e valeur de mon tableau : ${ table[1] } </p> | | Résultat du calcul : 42  Bonjour Alex  2e valeur du tableau : B  Rq : si name et table lui ont été adressés en Attribut par la Servlet. |
|  | conditions en EL | A = 8  A = true  A = '' | ${ A == 8 } True  ${ A > 5 } True  ${ A } False  ${empty A } False  ${ A } True  ${ A == true } True  ${ empty A } True | |  |
| empty A | renvoi true si objet vide ou null | ${ empty maVaribla } | ${ empty maVariable ? 'Vide' : maVariable } | |  |
| ${ ' ' } | déclaration d'une chaine de caractère | ${ 'Ma chaine de caractère' } |  | | Rq: en EL il est possible et même préférable de déclarer des String avec ' ' plutôt que " " afin de simplifier les expressions et ne pas mélanger les " " qui apparaitrons en JSTL |
| ${ maVar[ ] } | récupération d'un élément dans un tableau, liste, collection, … | ${ maListe[2] }  ${ maListe.get(2) } | ${ maListe['monElement'] } | | Rq : bien que les deux notations soient valables, il est préférable d'utiliser les [' '] pour les listes/tableaux/collections et le . pour les objets |
| pageScope.  requestScope.  sessionScope.  applicationScope. | objet Map permettant de récupérer des variables de chaque portée (définies avec JSTL ou avec Java : Beans) | ${ sessionScope.nomVariable }  ${ sessionScope.nomAttribut } |  | |  |
| param. | objet Map qui associe le nom et la valeur des paramètres de la requête  permet d'accéder à la valeur de chaque paramètre | ${ param.nomParametre } |  | |  |
| paramValues. | objet Map qui associe nom et valeurs multiples des paramètres  permet d'accéder aux valeurs en renvoyant un tableau de String | ${ paramValues.nomParametre[2] } | *JSTL*  <p>Pays visité(s) :</p>  <ul>  <c:forEach var="paysLi" items="${ paramValues.pays }" >  <li><c:out value="${ paysLi }" /></li>  </c:forEach>  </ul> | | code JSTL dans le cas d'un formulaire <select=… multiple="multiple"> |
| cookie. | objet Map qui associe le nom et l'instance des cookies | ${ cookie.nomCookie } |  | |  |
| Map | lorsqu'on utilise une Map qui renvoi dans l'item en cours le couple [nom, valeur] on obtient la valeur avec  value | ${ maMap[3].value.maProp } | <c:forEach var="clientTmp" items="${ sessionScope.clientsMap }">  <p>${ clientTmp.value.nom }</p>  <p>${ clientTmp.value.prenom }</p> | | *exemple : extraire les valeurs des prop. de tous les Beans* client *listés dans une Map* |
|  |  |  |  | |  |
|  |  |  |  | |  |
| JSTL | balises semblable à HTLM mais permettant un traitement similaire à Java |  |  | | ATTENTION : avec TomCat il faut inclure une copie du fichier de la bibliothèque dans WEB-INF/lib |
| <%@ taglib uri… %> | code en-tête de page JSP pour pouvoir utiliser la bibliothèque de balises JSTL | <%@ page pageEncoding="UTF-8" isELIgnored=“false” %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %> | *Format des balises JSTL alors associée:*  <c:instruction /> | | ATTENTION : code indispensable pour utiliser JSTL dans la page  Rq: isELIgnore="false" permet de s'assurer que les EL sont reconnues dans les JSP (au cas où elles soient ignorées pas défaut)  Rq : le préfixe choisit sera celui à utiliser dans les balises JSTL |
| web.xml | pour ne pas avoir à copier cet en-tête dans toutes nos pages JSP on peut l'enregistrer dans une page JSP indépendante et ajouter le code suivant à web.xml | <jsp-config>  <jsp-property-group>  <url-pattern>\*.jsp</url-pattern>  <include-prelude>/WEB-INF/taglibs.jsp</include-prelude>  </jsp-property-group>  </jsp-config> |  | | Rq : il n'est alors plus nécessaire d'inclure d'en-tête dans les pages JSP ! |
| <%@ page session="false" %> | supprime la création et l'accés aux données de type session sur toute la page | <%@ page session="false" %> |  | | utile si le contenu de la page est indépendant des sessions client et afin d'éviter la création session inutile par la balise <c:url value="/maPage" /> |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
| <c:out value=" " | affiche un élément (texte, valeur d'une variable, contenu d'un champs, …) | <c:out value="${ maVariable }" default="A afficher si variable vide" />  <c:out value="${ maVariable }">A afficher si variable vide</c:out> | <c:out value="${ monBean.maPropriete } /> | | Rq : avantage de value=" " est d'échapper les caractères spéciaux de HTML |
| escapeXml=" " | attribut à rajouter à la précédente  pour ne plus échapper les caractères spéciaux HTML | escapeXml="false" /> |  | |  |
| <c:set var=" " | crée/modifier une variable | <c:set var="maVariable" value="valeur1" scope="page" />  <c:set var="maVariable" scope="page">valeur1</c:set> | <c:set var="maVariable scope="session">  <c:out value="${ monBean.prop } default="valeurDéfaut" }" />  </c:set> | | création d'une variable prenant pour valeur celle d'une variable d'un Bean/d'un attribut/d'un paramètre/… |
| scope=" " | attribut déterminant la portée  de la variable  uniquement dans la page  lors des requêtes sur la même page  session de l'utilisateur (=visite sur site)  pour tous les utilisateurs utilisant le site | "page" (plus faible)  "request"  "session"  "application" (plus forte) |  | |  |
| <c:set target=" " | modifie un Bean (objet Java) envoyé sur la page | <c:set target="${ monBean }" property="maPropriete" value="valeur2" /> | value="${ null }" /> | | pour rendre une valeur nulle |
| <c:remove var=" " | supprime une variable | <c:remove var="maVariable" scope="scopeDeLaVariable" /> |  | |  |
|  |  |  |  | |  |
|  | Conditions |  |  | |  |
| <c:if test=" " | condition simple | <c:if test="${ condition }>Code HTML à utilizer si condition = true</c:if> |  | |  |
| var=" " | attribut permettant d'enregistrer le résultat de la condition (booléen) pour le réutiliser ensuite sans refaire le test | <c:if test="${ condition } var="maVariable">Code HTML</c:if> | <c:if test="${ condition } var="maVariable" scope="session">Code HTML</c:if> | | Rq : par défaut, la variable créée à une portée de "page" qui peut être modifiée avec l'attribut scope |
| <c:choose>  <c:when test= | condition multiple du type  if  else if  else if  else | <c:choose>  <c:when test="${ condition1 }>Code</c:when>  <c:when test="${ condition2 }>Code</c:when>  <c:otherwise>Code</c:otherwise>  </c:choose> |  | | à la 1re condition vérifiée, le code en question s'exécute et le test s'arrête (sortie de choose).  Si aucune des conditions n'est vérifiée, c'est le code de otherwise qui est exécuté par défaut.  Rq : otherwise est facultatif. |
|  |  |  |  | |  |
|  | Boucles |  |  | |  |
| <c:forEach | boucle de type  for( i=0, i <= 10, i++) | <c:forEach begin="0" end="10" step="1">Code</forEach> | <c:forEach var="A" begin="0" end="10" step="1"><p>Tour n°<c:out value="${ A }" /></p></forEach> | | Rq : on peut créer une variable (facultative) pour enregistrer le numéro de tour et s'en servir si besoin. |
| <c:forEach | boucle de type  for( var : tableau) | <c:forEach var="varTemp" items="${ monTableau }" >  <p><c:out value="${ varTemp }" /></p>  </c:forEach> | <c:forEach items="${ monTableau }" var="varTemp" begin="2" end="5"> | | Rq : pour parcourir tableau/liste/collection/…  Rq : on peut ajouter un rang initial et final si on ne veut pas parcourir la totalité du tableau |
| varStatus= | attribut permettant de générer dans un seul objet plusieurs informations sur la boucle en cours (valeur, n° de ligne, …) | <c:forEach items="${ monTableau }" var="varTemp" varStatus="monStatus"> | ${ monStatus.index } => index (init. à 0)  ${ monStatus.count } => n° ligne (init. à 1)  ${ monStatus.current } => valeur elt en cours  ${ monStatus.first } => true si 1e elt de liste  ${ monStatus.last } => true si dernier elt | |  |
| <c:forTokens | boucle permettant de découper une chaine de caractère selon un élément spécifique | <c:forTokens var="morceau" items="maChaineADécouper" delims="z">  <p><c:out value="morceau" /><p>  </c:forTokens> | <c:forTokens var="morceau" items="maChaineADécouper"  delims="z% !"> | | ici, la chaine sera découpée tout les z  ATTENTION : si plusieurs symboles sont placés dans l'attribut delims (y compris un espace), alors la chaine sera découpé à chacun de ses symboles pris individuellement ! |
|  |  |  |  | |  |
|  | URL |  |  | |  |
| <c:url | génère une URL valide dans la page | <a href=<c:url value="/maPage.jsp" />>Cliquez ici</a>  <a href=<c:url value="/monUrl" />>Cliquez ici</a> | *Code HTPM généré :*  <a href="/test/maPage.jsp">Cliquez ici</a>  ou  <a href="www.monSite.com/maPage.jsp">Cliquez ici</a> | | ATTENTION : si les paramètres sont bien encodés avec les balises <c:param> en revanche, le reste de l'adresse ne le sera pas avec <c:url> !  "Contexte" : l'adresse utilisée lors du développement n'est pas l'URL définitive du site (choisie lors de sa mise en ligne). <c:url > adaptera à l'URL définitive automatiquement  "Cookies" : en cas d'absence de cookie sur la machine du client, elle modifie l'adresse URL pour intégrer l'identifiant de session  Rq: si ce principe ne nous intéresse pas (page indépendante des sessions client) pour éviter création de session inutile : <%@ page session="false" %> |
|  |  | <a href=<c:url value="maPage.jsp" />>Cliquez ici</a> | <a href="maPage.jsp">Cliquez ici</a> | | ATTENTION : en cas d'URL relative (sans le / en début), elle n'est pas modifiée |
| <c:param | permet d'ajouter des paramètres à cette URL en les encodant comme il faut | <a href=<c:url value="/maPage.jsp">  <c:param name="param1" value="val1" />  <c:param name="param2" value="val2" />  </c:url>>Cliquez ici</a> |  | | valable uniquement entre les balises : <c:url>, <c:import> et <c:redirect> |
| <c:redirect | redirige l'utilisateur vers une autre page (éventuellement un autre site) de manière transparente | <c:redirect url="http://www.autreSite.com" /> |  | | Rq: balise <c:param> compatible |
|  |  |  |  | |  |
|  | Importation/Inclusion |  |  | |  |
| <c:import | importe une page (du même site ou non) et l'inclus dans une autre page JSP | <c:import url="maPage.jsp" />  <c:import url="www.autreSite.com/pageCible" /> |  | | Rq: balise <c:param> compatible |
|  | peut aussi permetre de stocker l'élément importé et le convertir en un autre format (voir bibliothèque xml de JSTL) | <c:import url="monFichier.xml" varReader="maVarBrute">  <x:parse var="maVarConvertie" doc="${ maVarBrute } />  </c:import> |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
| Upload fichiers |  |  |  | |  |
| <form | code HTTP pour un formulaire d'envoi de fichier | <form method="post" action="adressePage" enctype="multipart/form-data">  <input type="file" name="fichier" id="fichier" />  <input type="submit" />  </form> |  | |  |
| web.xml | code de configuration à ajouter au fichier web.xml pour configurer la réception des fichiers | <servlet>  <servlet-name>Test</servlet-name>  <servlet-class>com.octest.servlets.Test</servlet-class>  *<init-param>*  *<param-name>chemin</param-name>*  *<param-value>c:/fichierstmp</param-value>*  *</init-param>*  <multipart-config>  <location>c:/fichierstmp</location>  <max-file-size>10485760</max-file-size> <!-- 10 Mo -->  <max-request-size>52428800</max-request-size> <!-- 5 x 10 Mo -->  <file-size-threshold>1048576</file-size-threshold> <!-- 1 Mo -->  </multipart-config>  </servlet> | | <init-param>  <location>  <max-file-size>  <max-request-size>  <file-size-treshold> | (facultatif)  déclare un paramètre pouvant alors être récupéré dans la Servlet avec la méthode this.getInitParameter("nomParam");  adresse du dossier où stocker temporairement les fichiers  adresse absolue commencent par le disque (c: ou d: )  taille maximale autorisée pour chaque fichier (en octet)  taille max de la requête (peut contenir plusieurs fichiers)  taille minimale d'un fichier pour être stocké dans le dossier temporaire |
| Java | Classe Java (partie Modèle) qui traite le(s) fichier(s) contenu(s) dans la requête envoyé par la page en les enregistrant (après vérification de leur existence) dans un dossier | public static final String CHAMP\_FICHIER = "fichier";  public static final int TAILLE\_TAMPON = 10240;  public static final String CHEMIN\_FICHIERS = "/Users/…/fichiers/";  public String getFile(HttpServletRequest request) {  // On récupère le champ du fichier  Part part = request.getPart( CHAMP\_FICHIER );  // On vérifie qu'on a bien reçu un fichier  String nomFichier = getNomFichier(part);  // Si on a bien un fichier  if (nomFichier != null && !nomFichier.isEmpty()) {  String nomChamp = part.getName();  // Corrige un bug du fonctionnement d'Internet Explorer  nomFichier = nomFichier.substring(nomFichier.lastIndexOf('/') + 1).substring(nomFichier.lastIndexOf('\\') + 1);  // On écrit définitivement le fichier sur le disque  ecrireFichier(part, nomFichier, CHEMIN\_FICHIERS);  }  }  private static String getNomFichier( Part part ) {  for ( String contentDisposition : part.getHeader( "content-disposition" ).split( ";" ) ) {  if ( contentDisposition.trim().startsWith( "filename" ) ) {  return contentDisposition.substring( contentDisposition.indexOf( '=' ) + 1 ).trim().replace( "\"", "" );  }  }  return null;  }  private void ecrireFichier( Part part, String nomFichier, String chemin ) throws IOException {  BufferedInputStream entree = null;  BufferedOutputStream sortie = null;  try {  entree = new BufferedInputStream(part.getInputStream(), TAILLE\_TAMPON);  sortie = new BufferedOutputStream(new FileOutputStream(new File(chemin + nomFichier)), TAILLE\_TAMPON);  byte[] tampon = new byte[TAILLE\_TAMPON];  int longueur;  while ((longueur = entree.read(tampon)) > 0) {  sortie.write(tampon, 0, longueur);  }  } finally {  try {  sortie.close();  } catch (IOException ignore) {  }  try {  entree.close();  } catch (IOException ignore) {  }  }  } | | | ATTENTION : le chemin de fichiers a pour contexte le disque sur lequel tourne le serveur (c: sur l'ordinateur)  et doit obligatoirement finir par /  CHEMIN\_FICHIERS = "c:/Users/…/fichiers/";  CHEMIN\_FICHIERS = "d:/Users/…/fichiers/";  Rq : la Servlet doit rediriger la requête vers cette Classe pour que les fichiers soient traités.  Une fois traité, il est bon de renvoyer un message vers la Servlet qui la renverra à son tour vers la page actualisée précisant que le fichier a ou n'a pas été chargé convenablement.  Code avec vérifications et gestion des erreurs : voir *Exemple complet de code - Envoie de fichier* |
| IllegalStateException | type d'exception retournée si le/les fichier(s) transmis dépassent les limites de tailles définies dans <multipart-config> de la servlet dans *web.xml* |  |  | |  |
| IOException | si problème de localisation du dossier de stockage temporaire défini dans <multipart-config> de la servlet dans *web.xml* |  |  | |  |
| ServletException | si requête pas de type "multipart/form-data" si Servlet contacté par autre formulaire que celui prévu |  |  | |  |
| Téléchargement de fichiers |  |  |  | |  |
| .getPathInfo() | méthode de l'objet HttpServletRequest retournant la portion d'URL entre l'adresse de la servlet et les paramètres éventuels :  http://monsite.com/maservlet/mondossier/monfichier.txt?param1=valeur1 | String fichierDemande = request.getPathInfo(); |  | | Rq: lors du téléchargement d'un fichier, le nom du fichier demandé est placé dans l'URL lui-même et pas dans un paramètre afin d'éviter les comportements différents selon les navigateurs |
|  |  | Code complet de la Servlet : voir *Exemple complet de code - Téléchargement de fichier* | URLDecoder.decode(maChaine, "UTF-8"); | | retranscrits les caractères spéciaux placés dans l'URL (%E9, …) en caractères normaux (é, ê, à, espace, …) |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
| Basse de Données |  |  |  | |  |
| Class.forName( ) | chargement du Driver : MySQL/JDBC pour pouvoir utiliser JDBC et communiquer avec la BdD MySQL | //Chargement du driver  try {  Class.forName("com.mysql.jdbc.Driver");  } catch (ClassNotFoundException e) {  } | | | Rq : ce chargement doit se faire une seule fois lors de l'ouverture de l'appli (et non pas à chaque requête à la BdD)  ATTENTION : le fichier "mysql-connector-java-5.1.41.zip" doit être placé dans WEB-INF>lib |
| Connection  Statment  ResultSet | objets permettant   * la connexion à la BdD * la requête à la BdD * la récupération temporaire des données | //Connexion à la base  Connection maConnexion = null;  Statement monStatement = null;  ResultSet monResultat = null;  try {  maConnexion = DriverManager.getConnection("jdbc:mysql://localhost:33060/nomBdD", "user", "password");  monStatement = maConnexion.createStatement();  //Exécution de la requite  monResultat = monStatement.executeQuery("SELECT nomColonneA, nomColonneB FROM nomTableau;");  //Récupération des données  while (monResultat.next()) {  String maValeurA = monResultat.getString("nomColonneA");  int maValeurB = monResultat.getInt("nomColonneB");  //envoi de ces valeurs dans objets spécifiques pour garder en mémoire avant nouveau tour de boucle  instructions  }  } catch (SQLException e) {  } finally {  //Fermeture de la connexion  try {  if (monResultat != null)  monResultat.close();  if (monStatement != null)  monStatement.close();  if (maConnexion != null)  maConnexion.close();  } catch (SQLException ignore) {  }  } | | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  |  |  | |  |