# JAVA EE – Modèle : java Bean // Vue : .jsp // Contrôleur : Servlet

## **I - Web.xml**: *Associer les url aux servlet*

|  |  |  |
| --- | --- | --- |
| En-tete | <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"> **[INSTRUCTION]** </web-app> | |
| <servlet> </servlet> : **décrire les servlet**, à l’intérieur on utilisera : | | |
| <servlet-name>NomServlet</servlet-name> | | <servlet-class>com.octest.servlets.NomServlet</servlet-class> |
| <servlet-mapping> </servlet-mapping> : **lier le nom d’une servlet à une url** | | |
| <servlet-name>NomServlet</servlet-name> | | <url-pattern>/acceuil</url-pattern> |

## **II – Servlet** : *doGet est appeler et doPost seulement en cas de <form method=«post »>*

|  |  |  |  |
| --- | --- | --- | --- |
| Lien vers jsp | this.getServletContext().getRequestDispatcher("/WEB-INF/bonjour.jsp").forward(request, response); | | |
| Envoyer une variable contenue dans l’objet request (pour la jsp, java bean etc) | | | *request.setAttribute("heure", time);* |
| Récupérer un paramètre de l’objet request et le stock dans name | | | String name = request.getParameter(« name ») |
| Créer un objet HttpSession  Stocker une variable de session  Récupérer une variable de session  Supprime les données de session | | HttpSession session = request.getSession(); // penser a recréer l’objet dans doPost et doGet  session.setAttribute("var", var);  String prenom = (String) session.getAttribute(«prenom») ;  session.invalidate() ; | |
| Créer un objet cookie  Définie la durée de vie cookie (en seconde)  Ajoute le cookie a notre objet response  Récupérer un tableau de tous les cookies  Recuperer nom cookie / valeur cookie | | Cookie cookie = new Cookie("prenom", prenom);  cookie.setMaxAge(60 \* 60 \* 24 \* 30);  response.addCookie(cookie);  Cookie[] cookies = request.getCookies();  cookie.getName() // cookie.getValue() | |

## **III – JSP**

|  |  |
| --- | --- |
| Inclure du code java | Entre <% %> |
| Expression language | Entre ${ } // exemple : ${ variable } ${ empty name ? ‘ ’ : name} ${ auteur.nom } |
| Inclusion | <%@ include file= « menu.jsp » %> |
| Utiliser variable session | ${ sessionScope.var } |

## **IV – JSTL**: *à importer jstl-1.2.jar dans le dossier lib de web-inf*

|  |  |  |  |
| --- | --- | --- | --- |
| Inclure la bibliotheque jstl avec comme prefixe c | | | <%@ **taglib** uri="http://java.sun.com/jsp/jstl/core" prefix="c" %> |
| Affiche une variable Expression language avec echappement | | | **<c:out** value="${ i }" **/>** |
| Condition unique (boolean stocker dans «varBool»/ si true affiche ok ! | | | **<c:if** test=«${1>2}» var=«varBool»**>** ok ! **</c:if>** |
| Condition multiple | | **<c:choose>** <c:when test= «${1<2}»> si ok </c:when>  <c:otherwise>Si aucun when n’est true</c:otherwise> **<c:choose>** | |
| Boucle découper chaine caractères | **<c:forTokens** var="morceau" items="Un élément/Encore un élément/Un dernier" delims="/ "**>**  <p>${ morceau }</p> **</c:forTokens>** | | |
| Boucle (6 tour ici) | | **<c:forEach** var="i" begin="0" end="10" step="2"**>**  message n°<c:out value="${ i }" /> **</c:forEach>** | |
| Boucle sur tableaux | | **<c:forEach** items="${ titres }" var="titre" varStatus="status"**> //** Items = tableau[] // var = prend la valeur  <p>N°<c:out value="${ status.count }" /> : <c:out value="${ titre }" /> !</p> **</c:forEach>** | |

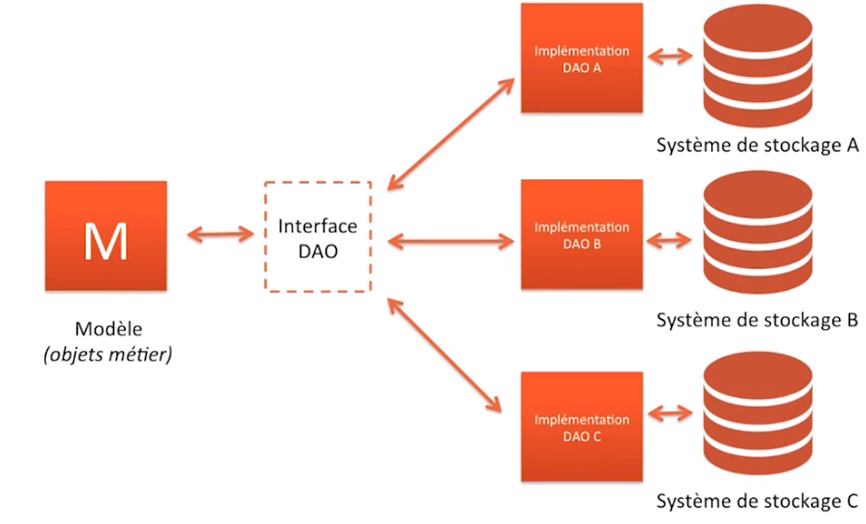
## **V – Java Bean**

|  |  |
| --- | --- |
| Récupère l’objet request dans une méthode | Public void verificationIdentifiants( **HttpServletRequest request**){…} |
| Tester l’égalité d’un string | maVarString**.equals(**laVarChaineAComparer**)** ; true si égal |

## **VI – JDBC***: connecteur entre java et la Base de données*

|  |  |  |  |
| --- | --- | --- | --- |
| Charger le driver pour connecter à la BDD (pour tomcat ) | | | try { Class.forName("com.mysql.jdbc.Driver") ;} catch (ClassNotFoundException e) {} |
| Connection base de données  (chemin , nom , pass) | | Connection connexion = null ; // initialise la valeur par null  try {connexion = DriverManager.getConnection("jdbc:mysql://localhost:3306/javaee", "root", "");  } catch (SQLException e) { e.printStackTrace();} } | |
| Créer un array (pour stocker)  Initialise les objets pour requête  Récupère le résultat brut  Boucle (résultat suivant)  Récupère le champs nom en String  Créer un utilisateur (java bean)  Remplis les champs avec setNom  Ajoute utilisateur à utilisateurs  Fermeture des connexions  |  |  |  Renvois l’array des utilisateur | | List<Utilisateur> utilisateurs = new ArrayList<Utilisateur>();  Statement statement = null; ResultSet resultat = null;  try {statement = connexion.createStatement();  resultat = statement.executeQuery("SELECT nom, prenom FROM noms;");  while (resultat.next()) {  String nom = resultat.getString("nom"); prenom = resultat.getString("prenom");  Utilisateur utilisateur = new Utilisateur();  utilisateur.setNom(nom); utilisateur.setPrenom(prenom);  utilisateurs.add(utilisateur); }  } catch (SQLException e) {  } finally {  try {if (resultat != null) resultat.close();  if (statement != null) statement.close();  if (connexion != null) connexion.close();  } catch (SQLException ignore) {} }  Return utilisateurs ; | |
| Requête préparer afin de faire un insert dans notre base de données | try {  PreparedStatement preparedStatement = connexion.prepareStatement("INSERT INTO noms(nom, prenom) VALUES(?, ?);");  preparedStatement.setString(1, utilisateur.getNom()); ***// 1 = premier ?***  preparedStatement.setString(2, utilisateur.getPrenom());  preparedStatement.executeUpdate();} **// méthode appeler lors d’une modification dans la bdd**  catch (SQLException e) {e.printStackTrace();}}} | | |

## **VII – Modele DAO** : *segmenter la partie modele*



Créer une classe DaoFactory : initialise le dao : connexion base de données

Créer une interface UtilisateurDao qui sera implémenté

Créer une classe UtilisateurDaoImpl : les requetes propres à notre implémentation

Dans notre servlet ont doit créer une méthode public void init() qui initiera la connexion à notre base de données et stocker la connexion dans un champs.

Utiliser les transaction (définir dans notre DaoFactory sur notre objet de connexion .setAutoCommit(false) puis avec la gestion des exceptions si la requête ne renvoies pas d’exception on fait un .commit() si une exeption est levé on fera un .rollback() pour annuler de notre objet connexion) et bien gérer les Exceptions…