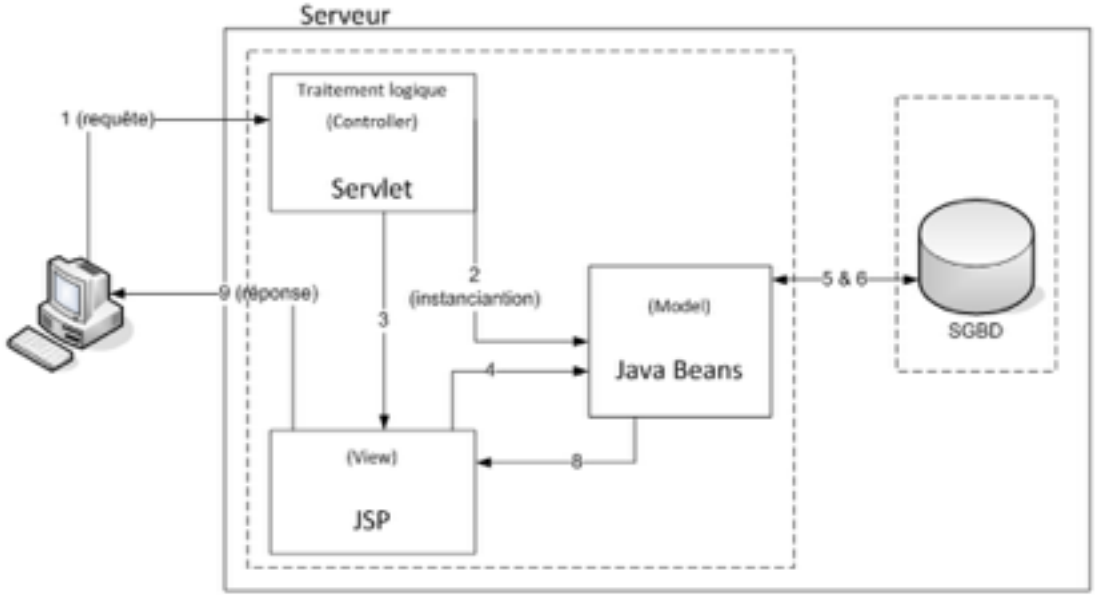
## 11. Décrire le modèle MVC dans le contexte des servlets et des Java Server Pages. Quels sont les principes et méthodes (paramètres, codes, tags, ...) à mettre en œuvre pour l'implémenter ? Décrire le protocole applicatif et son implémentation dans le cas classique d'un caddie virtuel. Donner une structure de base du code (principales variables membres, logique de base, transfert de pages, ...).

MVC = Model View Contrôler.

Le but d'un model MVC est de séparer 3 aspects distincts : L'aspect visuel (la view), c'est a dire la représentation des données et l'encodage des données, l'aspect traitement des données, c'est a dire la partie qui manipule les requêtes, donc le code métier, et enfin l'aspect accès des données, c'est a dire la partie qui s'occupe de rapatrier/exporter les données vers un SGBD.
Dans le cas des servlets et des JSP, ce model MVC sera représenté comme ceci :

- View : Ce seront les JSP qui joueront le rôle de view. En effet, c'est grâce aux pages crée par les JSP que les données seront visible par le client, ou que les formulaires pourront être rempli.
- Controler : ce sont les servlets qui joueront ce rôle. Ces servlets invoqué par une view traitera la demande et commandera les accès au données nécessaire, effectuera le traitement demandé puis redirigera vers la view adapter.
- Model : ce seront les Beans d'accès qui feront office de Model. Ces beans feront le pont entre le contrôler (les servlets) et le SGBD.




Nous voyons ici que le client fait sa requête qui est prise en charge par les servlets, celle-ci instancie les beans d'accès au SGBD, les informations sont récupéré par la JSP et le client reçois une réponse visible par un browser.
Un grand avantage de cette solution, outre l'avantage d'avoir un code plus claire et donc plus facile à développer. Une équipe de design peut travailler sur l'aspect visuelle avec les JSP (création de page HTML, de table CSS) alors que une équipe de développement peut travailler sur le code métier en lui même sur les servlets Java Beans.

Mécanisme de fonctionnement

Les servlets sont un point de passage obligé entre différente JSP. Ainsi un client ne pourra pas passer d'une JSP a l'autre, ce sera une servlet, donc le contrôler qui jouera sont rôle et redirigera le client.

Pour se faire, les JSP invoqueront une servlet avec un tag nommé par exemple « Action », permettant de faire comprendre a la servlet le travail demandé. La servlet pourra rediriger le client vers la JSP voulue grâce à la méthode :

public abstract void sendRedirect(String localisation) throws IOException

de la classe HTTPServletResponse. Cette méthode permet de renvoyer le client vers une ressource dont le chemin correspond au paramètre localisation. Une autre façon de faire est d'utiliser la méthode :

public RequestDispatcher getRequestDispatcher(String path)

de la classe ServletContext (obtenue par la méthode getServletContext sur l'objet Servlet). L'objet RequestDispatcher obtenu après l'appel getRequestDispatcher n'est pas la ressource en elle-même mais une enveloppe de celle-ci contenant sa référence. Sa méthode :

public void forward (ServletRequest request, ServletResponse response) throws ServletException, java.io.IOException

permet de faire chargé cette ressource dans le browser du client.

Un paramètre envoyé d'une JSP à une servlet ne sera pas renvoyé à une servlet ou JSP par la suite, sauf si la servlet qui à reçu ce paramètre le renvoi vers la JSP/Servlet cible. En effet, si tout les paramètres ajouter à une requête restait dans celle-ci, nous nous retrouverions avec une inflations des paramètres.

Pour partager des données de JSP à Servlet, nous utiliserons des Java Bean qui peuvent être mémorisé de différente façon. C'est différente façon correspond à la portée que nous voulons appliquer. La portée est la durée ou le Java Bean est disponible et si il est privé à un seul client. Il existe 3 différentes portées :

- Requête : Le Java Bean sera alors enregistré et géré au sein de l'objet HttpServletRequest grâces aux méthodes :
  - public void setAttribute(String name, Object object)
  - public Object getAttribute(String name)
  - public void removeAttribute(String name)

- Session : Le Java Bean sera alors enregistré et géré au sein de l'objet HttpServletSession grâce aux méthodes :
  - public void setAttribute(String name, Object object)
  - public Object getAttribute(String name)
  - public void removeAttribute(String name)

Cette objet est disponible pour chaque session et reste disponible tout au long de celle-ci (évidement, un objet différent par session).

- Application : Le Java Bean sera alors enregistré et géré au sen de l'objet ServletContext, il sera donc disponible dans toute les servlets d'un même projet WEB, et est donc identique pour tout les clients. Notre Bean sera géré via les méthodes :
  - public void setAttribute(String name, Object object)
  - public Object getAttribute(String name)
  - public void removeAttribute(String name)

**Pour manipuler le bean au sein d'une JSP, il faut ;**

- Avant le tag <html>, spécifier dans un premier temps l'utilisation du bean en ajoutabr le tag <jsp:useBean id="identifiant_du_bean" scope="session" class="LePackage.LaClasse" />.
- Si l'on désire accéder à un bean de portée différente, il faut alors changer la portée.
- Pour récupérer des données et visualiser le contenu, il suffit ensuite d'utiliser les tags <jsp :[get | set]property name … property/>:

<jsp:getProperty name="identifiant_du_bean" property="nom_de_la_property" />

**Ebauche de code illustrant ce qui a été dis ci-dessus:**

*Accueil.jsp*

*<html>*

**…**

*Ebauche de code :*
*// Acceuil.jsp :*

```
<form method="POST" action=http://www.exemple.com/servlet/ServletControle>
<input type="hidden" name="login" value="MonLogin ">
<input type="hidden" name="pass" value="MonPwd">
<input type="submit" name="action" value=Connexion">
</form>
…
</html>
```

*// Servlet :*

```
ServletControle.jsp
Public class ServletControle extends HttpServlet
{
    …
    Protected void processRequest(HttpServletRequest req,HttpservletResponse rep) throws…
    {
        String action = req.getParameter("action") ;
        if(action.equals(Connexion))
        {
            BeanClient bc = new BeanClient(req.getParameter("login"), req.getParameter("pass") ) ;
            //Soit un bean appelé BeanClient
            HttpSession session = req.getSession(true) ; //Porté session
            Session.setAttribute("client",bc) ; //Ajout du bean session pour l'accès aux données

            //Vérification du login/pass (peut être réalisé par un bean dédié à cela)
            …
            …

            //Transfert du bean
            ServletContext sc = getServletContext() ;
            RequestDispatcher rd = sc.getRequestDispatcher("/Bonjour.jsp" ) ;
            Rd.forward(req,rep) ;

        }
        …
    }
    …
}
```

*// Bonjour.jsp*

```
// Bonjour.jsp
…
<jsp :useBean id="client" scope="session" class="Bean.BeanClient"/> //Attribut "client" de la session
<html>
…
<jsp :getProperty name="client" property="login"/>  //Utilisation des parameters du bean « client » de la session
<jsp :getProperty name="client" property="pass"/>
</html>
…
```