T.P. 2
Servlet

1. Création d'un projet Web:

A l'aide d'Eclipse, créer un nouveau projet «sampleServlet» avec comme environnement d'exécution le serveur Tomcat installé dans le TP1, vous pouvez suivre les étapes suivantes:

Eclipse propose un type de projet particulier pour les applications Web : 'projet Web dynamique'. Il permet de créer une structure conforme au format .WAR et d'accéder aux APIs J2EE.
L'ouverture de l'assistant de création de projet Web dynamique peut se faire à partir du menu contextuel (clic droit) de la vue 'Explorateur de projet' dans la perspective J2EE.

La première page de l'assistant demande plusieurs informations, les plus importantes sont :

- Le nom du projet : ***sampleServlet***

- L'environnement d'exécution cible : il s'agit du serveur auquel sera associé ce projet.

- Appartenance à un EAR : le projet Web dynamique correspond à un WAR, lors du déploiement, la spécification J2EE prévoit qu'un WAR puisse être stocké dans un fichier EAR (WAR + EJB). Eclipse propose un type de projet particulier pour les EAR : projet d'applications d'entreprise. L'association d'un WAR avec un EAR est facultative. Dans le cadre de ce TP nous ciblons Tomcat qui ne supporte pas le format EAR donc la création d'un projet d'application d'entreprise n'est pas demandé.

La troisième page de l'assistant permet de renseigner deux informations importantes :
1 - Le nom de contexte : conformément à la spécification J2EE, chaque application à son propre nom de contexte qui apparaît dans les URL permettant d'accéder à l'application. Par défaut, Eclipse propose le même nom que le projet. **Modifier la valeur par défaut pour avoir «*helloWord*» comme nom de contexte.**
2 - Le nom du 'répertoire de contenu' : cette information est propre à Eclipse. Un projet Web Dynamique n'a pas directement la structure d'un WAR. La structure du WAR se trouve dans un sous-répertoire du projet, c'est le nom de ce sous-répertoire que ce champ permet de modifier (La valeur par défaut est 'WebContent'. L'intérêt de cette approche est de pouvoir stocker dans le projet des fichiers qui ne seront pas visibles par le serveur de test et qui ne seront pas exportés lors de la création du fichier WAR correspondant au projet. Par exemple, le code source n'est pas stocké dans le WAR il se trouve dans un sous-répertoire du projet nommé, par défaut, 'src' (voir la page précédente de l'assistant).

2. Développement d'une Servlet:

Le but de cette section est de refaire l'exercice 6 du TP 1 en utlisant une Servlet au lieu d'une JSP:

- La création d'une servlet est possible en utilisant l'assistant de création de classe, mais pour simplifier les choses Eclipse propose un assistant spécifique. L'ouverture de cet assistant peut se faire à partir du menu contextuel associé au projet Web (clic droit sur le nom du projet), dans ce menu choisir 'Nouveau->Servlet':
Dans la première page de l'assistant, les informations importantes sont le nom de package (« com.tp2 » par exemple) et le nom de classe de la servlet : «***HelloServlet***»

- La seconde page de l'assistant permet de saisir des informations qui seront utilisées pour déclarer la servlet dans le descripteur de déploiement de l'application (fichier web.xml). La plus importante de ces informations et la liste des alias (champ 'Mappage d'URL'). Ces alias (pointeurs sur la ressource de type servlet) apparaîtront dans les URL permettant d'accéder à la servlet (la forme de ces URLs sera http://nomDeServeur:port/nomDeContexte/Alias). Une servlet peut avoir plusieurs alias, par défaut Eclipse définit un alias en utilisant le nom de la classe de la servlet. **Modifier la valeur par défaut pour avoir «/hello» comme alias.**

- La dernière page de l'assistant permet de sélectionner les méthodes qui devront être générées lors de la création de la classe du servlet**.**

- Après avoir cliqué sur 'Terminer', l'assistant crée la classe. Cette nouvelle classe apparaît dans la vue 'Explorateur de projet' et est ouverte en édition.

- Outre la création de la classe du servlet, l'assistant de création de servlet a mis à jour le fichier web.xml. **Editer ce fichier, et à l'aide de l'onglet « Source », vérifier que Eclipse a bien associer la classe de servlet «HelloServlet» à l'alias «/hello».**

- Modifier la classe de la servlet «HelloServlet» pour refaire l'exercice 6 du TP 1.

- Pour tester la servlet, Eclipse propose un moyen rapide de lancer le serveur Tomcat (s'il n'est pas déjà démarré) et d'invoquer la servlet. Cette action se fait à partir du menu contextuel associé à la classe de la servlet, dans ce menu choisir 'Exécuter en tant que->Exécuter sur le serveur'. Le serveur peut aussi être lancé en mode debug en utilisant 'Deboguer en tant que->Deboguer sur le serveur'. Lors de la première utilisation de cette opération une boîte de dialogue permet d'indiquer le serveur sur lequel l'exécution doit se faire. Eclipse ouvre ensuite un navigateur Web dans la zone d'édition et invoque la servlet. Dan notre exemple, il faut modifier l'URL déclenchée automatiquement par Eclipse pour ajouter les paramètres attendus par notre servlet.

3. Développement d'une servlet AdminPizza (nouveau projet TP2_Pizza)

L'objectif de cette section est de mettre en place un site web pour l'administration de la vente (en ligne) de pizzas : une pizza est déterminée par son type et son prix à l'unité.
Plus techniquement, l'objectif est de vous faire pratiquer l'utilisation de l'Objet Session: instance de la classe HttpSession, voir la javadoc de cette classe à l'adresse: http://docs.oracle.com/javaee/5/api/javax/servlet/http/HttpSession.html.

La Session va servir à stocker les objets de type «Pizza» (par exemple la session peut contenir une liste (ou une HashMap) stockant tous les pizzas crées, ...). L'appel «request.getSession()» permer de récupérer la Session utilisateur courante.

Dans un nouveau projet TP2_Pizza, utiliser le code en annexe pour créer la Servlet d'administration des pizzas «*AdminPizza*» *(package administration)*. L' URL pour appeler cette Servlet doit être de la forme: «*http://localhost:8080/TP2_Pizza/Admin_Pizza*».

Dans le répertoire «src», créer les classes JavaBeans «Pizza.java» et «Stock.java», il suffit de compléter le code des deux JavaBeans suivants (getter et setter pour tous les champs):

```
package entityPizza;

public class Pizza implements Serializable {

    private String pizzaId;
    private Integer prix;

    public Pizza() {
    }

   public String getPizzaId() {
        return pizzaId;
    }
    ...
    ...
    public void setPrix(Integer prix) {
        this.prix = prix;
     }
}
```

```
package entityPizza;

public class Stock {
    private Integer quantite;
    private String pizzaId;

    public Stock() {
    }

    public Integer getQuantite() {
        return quantite;
    }
   ...
   ...

    public void setPizzaId(String pizzaId) {
        this.pizzaId = pizzaId;
    }
}
```

Ensuite, créez les deux classes (DAO) PizzaFacade.java et StockFacade.java. Ces deux classes vont jouer le rôle d'interface entre la Servlet et la source de stockage des données (ici la Session). PizzaFacade.java doit contenir les méthodes appelées par la Servlet pour gérer le stockage et la récupération des JavaBeans «Pizza.java» (de même pour StockFacade.java et les JavaBeans «Stock.java»).

4. En s'inspirant du formulaire de la Servlet «*AdminPizza*», développer et tester une Servlet qui permet la gestion de commandes de pizzas « *Commande_Pizza*.java ».

Annexe: Code pour Servlet «AdminPizza»:

```
import entityPizza.Pizza;
import entityPizza.PizzaFacade;
...
public class AdminPizza extends HttpServlet {
    private StockFacade stockFacade = new StockFacade();
    private PizzaFacade pizzaFacade = new PizzaFacade();

      @Override
        protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
            response.setContentType("text/html;charset=UTF-8");
            PrintWriter out = response.getWriter();
            try {
                out.println("<html>");
                out.println("<head>");
                out.println("<title>Administration Pizza</title>");
                out.println("</head>");
                out.println("<body>");
                out.println("<h1>Ajouter un nouveau type de pizza : </h1>");

                List lPizza = pizzaFacade.getAllPizzas();
                for (Iterator it = lPizza.iterator(); it.hasNext();) {
                    Pizza pizzaBean = (Pizza) it.next();
                    out.println("Type : <b>" + pizzaBean.getPizzaId() + " </b> ");
                    out.println("Prix : " + pizzaBean.getPrix() + "<br/>");
                }
                String type=null;
                type=request.getParameter("type");
                if (type!=null) {
                    try {
                        int prix=0;
                        prix=new Integer(request.getParameter("prix"));
                        int quantite=new Integer(request.getParameter("quantite"));
                        //On ajoute un type de pizza
                        Pizza pizzaBean = new Pizza();
                        pizzaBean.setPizzaId(type);
                        pizzaBean.setPrix(prix);
                        pizzaFacade.create(pizzaBean);
                        Stock stockBean = new Stock();
                        stockBean.setPizzaId(type);
                        stockBean.setQuantite(quantite);
                        stockFacade.create(stockBean);
                        //La pizza et son Stock sont sauvegardés
                        //Donc, on recharge la page
                        response.sendRedirect("Admin_Pizza");
                    } catch (Exception ex) {
                        ex.printStackTrace();
                    }
                } else {
                    out.println("<form method='POST'>");
                    out.println("Type: <input type='text' name='type'><br/>");
                    out.println("Prix: <input type='text' name='prix'><br/>");
                    out.println("Quantité: <input type='text' name='quantite'><br/>");
                    out.println("<input type='submit'><br/>");
                    out.println("</form>");
                }
                out.println("<a  href='Commande_Pizza'> commande pizza</a>");
                out.println("</body>");
                out.println("</html>");
            } finally {
                out.close();
            }
        }
...
```