

# TD n°1, éléments de correction

## Exercice 1

Corrigé dans l'énoncé

## Exercice 2

Exemple d'énumération des variables passées dans l'url :

```
<%@page import="java.util.Enumeration"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Récupération GET</title>
</head>
<body>
    <table border="1">
        <thead>
            <tr>
                <th>Nom</th>
                <th>Valeur</th>
            </tr>
        </thead>
        <%
            Enumeration<String> names = request.getParameterNames();
            while (names.hasMoreElements()) {
                String name = names.nextElement();
                String value = request.getParameter(name);
                out.print("<tr>" + name + "<td></td><td>" + value + +
"</td></tr>");
            }
        %>
    </table>
</body>
</html>
```

## Exercice 3

Récupération des variables d'un formulaire posté dans une servlet

```
@WebServlet(name="Submit", urlPatterns = { "/submit.do" })
public class Submit extends HttpServlet {
```

```

private static final long serialVersionUID = 1L;

/**
 * Retourne le flux de sortie
 * @param response
 * @return
 * @throws IOException
 */
private PrintWriter getOut(HttpServletRequest response) throws IOException{
    response.setCharacterEncoding("UTF8");
    response.setContentType("text/html");
    PrintWriter out=response.getWriter();
    return out;
}
/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    PrintWriter out=getOut(response);
    out.print("Méthode GET interdite sur cette page");
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    PrintWriter out=getOut(response);
    if(request.getParameter("name")!=null){
        out.print("Informations saisies :<br>");
        out.print("<div>Nom : "+request.getParameter("name")+"</div>");
        if(request.getParameter("password")!=null)
            out.print("<div>Mot de passe :<br>"+request.getParameter("password")+"</div>");
    }
}
}
}

```

## Exercice 4

Gestion des informations temporaires, sessions et cookies  
JSP et servlets, mise en place d'un contrôleur

Package	Classe	Description
net.td.metier	Link	Représente un lien Internet

### Classe métier :Link

```
package net.td.metier;
```

```

public class Link {
    private String nom;
    private String adresse;
    public Link() {
        this("", "http://");
    }
    public Link(String nom, String adresse) {
        super();
        this.nom = nom;
        this.adresse = adresse;
    }
    public String getNom() {
        return nom;
    }
    public void setNom(String nom) {
        this.nom = nom;
    }
    public String getAdresse() {
        return adresse;
    }
    public void setAdresse(String adresse) {
        this.adresse = adresse;
    }
    public boolean isValid(){
        if(adresse==null || "".equals(adresse))
            return false;
        Pattern p=Pattern.compile("^((http|https|ftp){1}(://){1}).+?$");
        Matcher m=p.matcher(adresse);
        return m.matches();
    }
    @Override
    public String toString() {
        return "Link [nom=" + nom + ", adresse=" + adresse + "]";
    }
    public void copyFrom(Link aLink){
        nom=aLink.getNom();
        adresse=aLink.getAdresse();
    }
}

```

## Classes techniques

### Gateway : Passerelle persistance/métier

```

package net.td.utils;

public class Gateway {

    /**
     * Retourne la liste des liens contenue dans la session
     * Si la liste n'existe pas encore en Session, celle-ci est créée
     * @param request Requête Http

```

```

 * @return Liste des liens
 */
public static ArrayList<Link> getLinks(HttpServletRequest request){
    ArrayList<Link> result;
    if(request.getSession().getAttribute("links")!=null){
        result=(ArrayList<Link>) request.getSession().getAttribute("links");
    }else
    {
        result=new ArrayList<Link>();
        request.getSession().setAttribute("links", result);
    }
    return result;
}
/**
 * Retourne un lien à partir du paramètre id passé dans la requête
 * @param request
 * @return
 */
public static Link getLink(HttpServletRequest request){
    Link result=null;
    if(request.getParameter("id")!=null){
        try{
            int id=Integer.valueOf(request.getParameter("id"));
            result=getLinks(request).get(id);
        }catch(Exception e){
            result=null;
        }
    }
    return result;
}
/**
 * Mise à jour ou ajout à partir des paramètres de request du lien aLink
 * @param aLink
 * @param request
 */
public static void updateLink(Link aLink,HttpServletRequest request){
    ArrayList<Link> links=getLinks(request);
    Link theLink=getLink(request);
    if(theLink!=null){
        theLink.copyOf(aLink);
    }else{
        links.add(aLink);
    }
}
}

```

## Gui : Interfaces

```

package net.td.utils;

public class Gui {
    /**
     * Retourne un lien au format HTML

```

```

    * @param aLink
    * @param index
    * @return
    */
public static String showLink(Link aLink,int index){
    String result=<div><a href='"+aLink.getAdresse()+"'
target='openLink'>"+aLink.getNom()+"</a>&nbsp;<a
href='update.do?id='"+index+"'>...</a></div>";
    return result;
}
/***
 * Retourne une liste de lien au format HTML
 * @param links
 * @return
 */
public static String ShowLinks(ArrayList<Link> links){
    String result="";
    for(int i=0;i<links.size();i++){
        result+=showLink(links.get(i),i);
    }
    return result;
}
}

```

## Vues

```

<%@page import="net.td.utils.Gateway"%>
<%@page import="net.td.metier.Link"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
String ck="";
String id=request.getParameter("id");
Link link=Gateway.getLink(request);
if(link!=null){
    ck=<input type='checkbox' name='delete' id='delete'>;
    ck+="Supprimer le lien</label>";
}else
    link=new Link();
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Ajout/modification de lien</title>
</head>
<body>
<form action="update.do" method="post">
    <input type="hidden" id="id" name="id" value="<%=id%>">
    <div>
        <label for="nom">Nom :</label>
        <input type="text" name="nom" id="nom" value="<%=link.getNom()%>">
    </div>

```

```

<div>
    <label for="adresse">Adresse :</label>
    <input type="text" name="adresse" id="adresse"
value="<%=link.getAdresse()%>">
</div>
<div><%=ck %></div>
    <input type="submit" value="Continuer >>">
</form>
</body>
</html>

```

## Contrôleur

Le contrôleur gère :

- La logique applicative (réponses aux requêtes, redirections, chargement des vues)
- Le contrôle des données

```

package net.td.controller;

/**
 * Servlet implementation class SLink
 */
@WebServlet({ "/SLink", "*.do" })
public class SLink extends HttpServlet {
    private static final long serialVersionUID = 1L;
    /**
     * Retourne l'action sollicitée par la requête
     * @param request
     * @return
     */
    private String getAction(HttpServletRequest request){
        String result="";
        String[] parts=request.getRequestURI().split("/");
        if(parts.length>0)
            result=parts[parts.length-1];
        return result;
    }
    /**
     * Charge une vue (jsp située dans le dossier protégé WEB-INF)
     * @param viewName
     * @param request
     * @param response
     * @throws ServletException
     * @throws IOException
     */
    private void loadView(String viewName,HttpServletRequest
request,HttpServletResponse response) throws ServletException, IOException{
        request.getRequestDispatcher("WEB-INF/"+viewName+".jsp").forward(request,
response);
        System.out.println("redirection vers WEB-INF/"+viewName+".jsp");
    }
    /**

```

```
* @see HttpServlet#HttpServlet()
*/
public SLink() {
    super();
}

/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    PrintWriter out=response.getWriter();
    String action=getAction(request);
    switch (action) {
        case "list.do":
            loadView("listLinks", request, response);
            break;
        case "update.do":
            loadView("frmLink", request, response);
            break;
        default:
            System.out.println("Aucune action du nom de "+action+"(GET) à
effectuer");
            break;
    }
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    String action=getAction(request);
    switch (action) {
        case "delete.do":
            break;
        case "update.do":
            updateLink(request, response);
            break;
        default:
            System.out.println("Aucune action du nom de "+action+"(POST) à
effectuer");
            break;
    }
}

/**
 * Procédure de contrôle de la mise à jour/ajout d'un lien
 * @param request
 * @param response
 * @throws ServletException
 * @throws IOException
 */
private void updateLink(HttpServletRequest request,HttpServletResponse
```

```
response) throws ServletException, IOException {
    Link aLink=new Link();
    if(request.getParameter("nom")!=null &&
request.getParameter("adresse")!=null){
        aLink.setNom(request.getParameter("nom"));
        aLink.setAdresse(request.getParameter("adresse"));
        if(aLink.isValid()){
            Gateway.updateLink(aLink, request);
            loadView("listLinks", request, response);
        }else{
            request.setAttribute("link", aLink);
            loadView("error", request, response);
        }
    }
}
```

From:  
<http://slamwiki2.kobject.net/> - **SlamWiki 2.1**



Permanent link:  
<http://slamwiki2.kobject.net/javaee/td1?rev=1383637119>

Last update: **2019/08/31 14:42**