2025/10/20 06:51 1/9 Mise en place de MVC

Mise en place de MVC

Activités du référentiel concernées :

• D4.1 - Conception et réalisation d'une solution applicative

Contexte

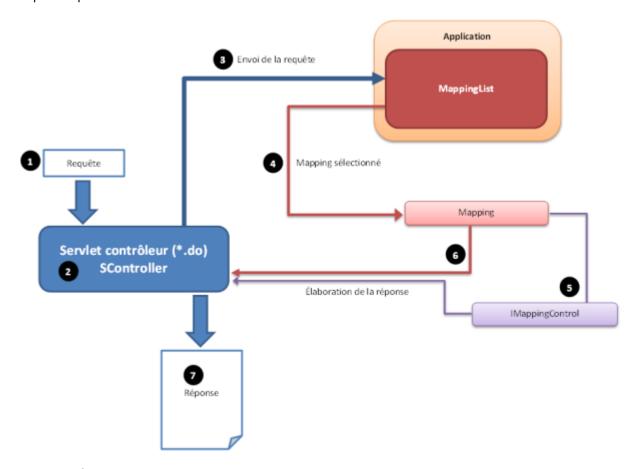
Il s'agit du même contexte que dans le TP précédent.

Le développement à effectuer le sera dans le cadre d'une architecture applicative de type client serveur Web...

Il s'agit de mettre en place un contrôleur unique dans le cadre de MVC2, d'en comprendre la logique, et les intérêts.

Contraintes techniques

Le contrôleur unique doit permettre de gérer la logique applicative du site, et de lancer les éventuels contrôles à effectuer sur l'accès aux pages. On entend par logique applicative l'association entre une requête, et la réponse qui sera fournie.



Principe à mettre en place :

1. Une requête en *.do est envoyée vers le serveur

- 2. Elle est prise en charge par le contrôleur unique (servlet **SController**)
- 3. La requête est comparée à la liste des mappings présents dans l'application
- 4. Le premier mapping (s'il existe) correspondant à la requête est sélectionné
- 5. Si le mapping comporte un contrôle, celui-ci est effectué
- 6. La réponse est préparée
- 7. Elle est ensuite renvoyée vers le client

Missions

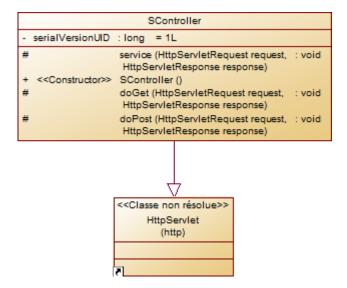
- Implémenter les classes décrites ci dessous
- Concevoir et réaliser les tests nécessaires pour éprouver le fonctionnement dans les différentes situations possibles
- Sécuriser le site en empêchant les requêtes vers les autres URL que *.do
- Créer une classe de gestion d'un fichier de mappings (mappingList). Intégrer cette gestion sur l'application web du tp4MVC.

Annexes

Package web.controller

Servlet SController

Servlet centralisant toutes les requêtes effectuées vers le serveur



Ih SController.java

```
package web.controller;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
```

2025/10/20 06:51 3/9 Mise en place de MVC

```
* Servlet implementation class SController
@WebServlet("*.do")
public class SController extends HttpServlet {
    private static final long serialVersionUID = 1L;
   /* (non-Javadoc)
    * @see
javax.servlet.http.HttpServlet#service(javax.servlet.http.HttpServletRequest,
javax.servlet.http.HttpServletResponse)
     */
    /**
     * Appelle la méthode process sur la liste des mappings récupérée dans
l'application
     */
    @Override
    protected void service(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        // TODO à implémenter
        super.service(request, response);
    }
    /**
     * @see HttpServlet#HttpServlet()
    */
    public SController() {
        super();
    * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    }
    /**
    * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
    */
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    }
```

Classe Mapping et interface IMappingControl

La classe Mapping représente une redirection possible, à partir d'une requête, vers une réponse, en utilisant

Last update: 2019/08/31 14:38

éventuellement une instance de lMappingControl pour vérifier la validité de la requête.

```
Mapping
# requestURL
                 : String
# responseURL : String
# controllerClass : Class<IMappingControl>
+ <<Constructor>> Mapping (String requestURL,
                   String responseURL)
+ <<Constructor>> Mapping (String requestURL,
                   String responseURL,
                   Class<IMappingControl> controllerClass)
                   getMappingControl ()
                                                              : IMappingControl
#
                   getResponseURL (HttpServletRequest reque : String
                   process (HttpServletRequest request,
                                                             : boolean
                   HttpServletResponse response)
                   matches (HttpServletRequest request)
                                                            : boolean
                   execute (HttpServletRequest request,
                                                             : boolean
                    HttpServletResponse response)
```

0-	IMappingControl	
+	isValid (HttpServletRequest request, HttpServletResponse response)	: boolean
+	onInvalidControl (HttpServletRequest req uest, HttpServletResponse response)	: void
+	beforeProcessAction (HttpServletRequest request, HttpServletResponse response)	: boolean

|h IMappingControl

```
package web.controller;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 * @author jc
 */
public interface IMappingControl {
     * Contrôle la validité d'une requête
     * @param request requête
     * @param response réponse
     * @return vrai si la requête est valide
    public boolean isValid(HttpServletRequest request,HttpServletResponse
response);
     * Méthode appelée si la requête n'est pas valide
     * @param request requête
     * @param response réponse
     */
    public void onInvalidControl(HttpServletRequest
request, HttpServletResponse response);
     * Contrôle la requête avant toute action
     * @param request requête
     * @param response réponse
     * @return vrai si l'exécution peut continuer
    public boolean beforeProcessAction(HttpServletRequest
request,HttpServletResponse response);
```

Ih Mapping

2025/10/20 06:51 5/9 Mise en place de MVC

```
package web.controller;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
* Représente un mapping d'URL, permettant d'interpréter une requête, et de
fournir une réponse
* @author jc
*/
public class Mapping {
    protected String requestURL;
    protected String responseURL;
    protected Class<IMappingControl> controllerClass;
    public Mapping(String requestURL, String responseURL) {
        //TODO à implémenter
    public Mapping(String requestURL, String responseURL,
Class<IMappingControl> controllerClass) {
       //TODO à implémenter
    }
    /**
     * @return l'instance de la classe de contrôle
    public IMappingControl getMappingControl(){
       //TODO à implémenter
        return null;
    * Retourne l'url de réponse associée à la requête
    * @param request requête
    * @return url de réponse
    */
    protected String getResponseURL(HttpServletRequest request){
        //TODO à implémenter
        return "";
    }
    /**
     * Effectue la redirection vers la requête appropriée
    * @param request
    * @param response
     * @return
    protected boolean process(HttpServletRequest request,HttpServletResponse
response){
       //TODO à implémenter
        return false;
    }
    /**
```

```
* Vérifie que la requête correspond à requestURL du mapping courant
     * @param request requête
     * @return vrai si la requête correspond au mapping courant
     */
    public boolean matches(HttpServletRequest request){
        //TODO à implémenter
        return false;
    /**
     * Appelle le process en tenant compte de l'appel des méthode sur
l'instance IMappingControl
     * Dans l'ordre : si beforeProcessAction alors (si isValid, process sinon
onInvalidControl)
    * @param request requête
     * @param response réponse
     * @return vrai si le mapping a été effectué sans erreur
     */
    public boolean execute(HttpServletRequest request,HttpServletResponse
response){
        //TODO à implémenter
        return false;
```

Classe MappingList

Représente une collection de mappings intégrée dans une application.

```
MappingList
items : ArrayList<Mapping>
<<Constructor>> MappingList ()
                 add (String requestURL,
                                                          : void
                  String responseURL,
                  Class<IMappingControl> controllerClass)
                 add (String requestURL,
                                                          : void
                 String responseURL)
                                                          : void
                 add (Mapping mapping)
                 remove (int index)
                                                          : Mapping
                 remove (Mapping mapping)
                                                          boolean
                 getFirstMatches (HttpServletRequest requ : Mapping
                 process (HttpServletRequest request,
                  HttpServletResponse response)
```

|h MappingList

```
package web.controller;
import java.util.ArrayList;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
    * Classe contenant la liste des mappings de l'application
    * @author jc
```

2025/10/20 06:51 7/9 Mise en place de MVC

```
*
 */
public class MappingList {
    private ArrayList<Mapping> items;
    public MappingList(){
       //TODO à implémenter
    /**
    * Crée et ajoute un mapping dont les paramètres sont passés à la méthode
    * @param requestURL URL de la requête
    * @param responseURL URL de la réponse
     * @param controllerClass Classe de contrôle
    */
    public void add(String requestURL, String responseURL,
Class<IMappingControl> controllerClass) {
       //TODO à implémenter
    * Crée et ajoute un mapping dont les paramètres sont passés à la méthode
    * @param requestURL URL de la requête
    * @param responseURL URL de la réponse
    */
    public void add(String requestURL, String responseURL) {
       //TODO à implémenter
     * Ajoute le mapping passé en paramètre à la liste des mappings
    * @param mapping
    public void add(Mapping mapping) {
       //TODO à implémenter
     * Supprime le mapping dont l'index est passé en paramètre
    * @param index index du mapping à supprimer
     * @return vrai si le mapping a été supprimé
    */
    public Mapping remove(int index){
       //TODO à implémenter
       return null:
    }
    /**
     * Supprime le mapping passé en paramètre
    * @param mapping mapping à supprimer
    * @return vrai si le mapping a été supprimé
     */
    public boolean remove(Mapping mapping){
```

```
//TODO à implémenter
        return false;
    /**
     * Retourne le premier mapping correspondant à la requête
     * @param request requête
     * @return le mapping correspondant à la requête
    public Mapping getFirstMatches(HttpServletRequest request) {
        //TODO à implémenter
        return null;
     * Recherche le mapping correspondant à la requête dans la liste des
mappings
    * puis l'exécute. Ne fait rien si aucun mapping ne correspond
     * @param request
     * @param response
    public void process(HttpServletRequest request,HttpServletResponse
response) {
        //TODO à implémenter
```

Sécurisation

Le contrôleur redirige les requêtes en *.do. Si nécessaire, il peut être judicieux de restreindre les autres requêtes, pour des raisons de sécurité par exemple.

Le fichier **web.xml** de configuration d'une application web associée à Tomcat permet de poser des restrictions d'accès (comme les htaccess avec Apache).

Dans l'exemple suivant, les requêtes vers les jsp sont interdites, ainsi que l'accès au dossier config du serveur, et à tout son contenu.

WEB-INF/web.xml

From:

http://slamwiki2.kobject.net/ - SlamWiki 2.1

Permanent link:

http://slamwiki2.kobject.net/slam4/tp4?rev=1353763871

Last update: 2019/08/31 14:38

