TP 4 : Struts :

Exercice 1 :

2°)

**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;

/\*\*

\* Servlet implementation class Hello

\*/

**public** **class** Hello **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

/\*\*

\* **@see** HttpServlet#HttpServlet()

\*/

**public** Hello() {

**super**();

// **TODO** Auto-generated constructor stub

}

/\*\*

\* **@see** HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

// **TODO** Auto-generated method stub

response.getWriter().append("Hello world ").append(request.getContextPath());

}

/\*\*

\* **@see** HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

// **TODO** Auto-generated method stub

doGet(request, response);

}

}

3°) l’annotation présente : @WebServlet("/Hello")

4°) Il est nécessaire d’ajouter :

<servlet>

<servlet-name>action</servlet-name>

<servlet-class>fr.esigelec.Hello</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>action</servlet-name>

<url-pattern>\*.php</url-pattern>

</servlet-mapping>

Résultat :



5°) Il est nécessaire d’ajouter :

<servlet>

<servlet-name>action</servlet-name>

<servlet-class>fr.esigelec.Hello</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>action</servlet-name>

<url-pattern>\*.do</url-pattern>

</servlet-mapping>

Résultat :



6°)



7°)Cet exercice nous permet de voir l’utilisation de mapping des servlets.

On redirige tous les urls terminant par .do, vers un unique servlet ici Hello.

Il s’agit de « l’initiation » pour comprendre le principe de struts.

Exercice 2 :

4°)

* Controller :
  + Le package Controller contient tous les « Controller » du site qui vont permettre d’exécuter les fichiers. Toute les classes qui seront présentes dans ce package hériterons de la classe Action de Struts.
* Dao :
  + Il s’agit du package qui va contenir les interfaces qui vont définir les différentes méthodes pour le classe DAO en fonction du type de connexion (Hibernate, JDBC)
* Dao.JDBC :
  + Contient toutes les implémentations et toutes les différentes classes nécessaires au fonctionnement du projet via l’utilisation de JDBC.
* DAO.Hibernate :
  + Contient toutes les implémentations des précédentes interfaces définit dans le package DAO, ainsi que toutes les classes nécessaires au fonctionnement du projet via l’utilisation d’Hibernate.
* DTO :
  + Contient les classes métiers correspondant aux interfaces.
* Forms :
  + Contient toutes les classes qui vont être reliées au form dans le fichier JSP. Les différents liens seront faits dans le fichier de configuration.
* Business :
  + Ce package est pour le moment vide. IL contient toute la logique comme les services.

Les fichiers suivants servent :

* Web.xml :
  + Permet de définir les différents paramètres pour l’utilisation du site. Il va également permettre la déclaration des différents servelts, form, ainsi que les patterns d’URL.
* Struts-config.xml
  + Contient les différents cas et type de retour possible lors de la validation ou non d’un formulaire, ainsi que la déclaration des différentes classes pour les formes.
* Bdd.properties :
  + Définit les différents éléments pour permettre la connexion à la base de données (URL de connexion, Login, Driver nécessaire, password, … etc).
* mesMessages.properties :
  + Permet de définir les messages d’erreur ou le texte qui s’affichera dans la page en permettant d’avoir une personnalisation en fonction de la langue par exemple.

5°)

A°) Une Connexion Pool est une classe permettant la gestion de la connexion vers la base de données de plusieurs utilisateur en même temps en utilisant un pool de connexion.

B°) Un pool de connexion va générer et gérer un certain nombre de connexion ouverte et prête à l’emploi par les futur utilisateurs. Il ne sera donc pas nécessaire de faire toute l’étape d’ouverture et de connexion pour l’utilisateur. La Connexion sera déjà prête.

C°) La méthode close de la connexion va fermer la connexion courante de l’utilisateur / contexte.

6°)

1. Certaines pages se trouvent dans le dossier WEB-INF car cela permet d’empêcher l’accès à ces différentes pages via un accès URL. Ces pages ne sont appelable uniquement via l’utilisation d’un controller.
2. Index.jsp n’est pas dans le dossier car cela nécessiterait de crée une identification de la route avec l’addresse « / » et donc appeler un controller. Il est plus simple d’uniquement afficher la page.

7°) Le bean avec la portée session n’est valable que pour la session. IL n’est pas uniquement valable pour la page en cours.

8°)

Message dans mesMessages\_fr

erreur.nom.vide=le nom ne doit pas être vide (message francais)

erreur.prenom.vide=le prénom ne doit pas être vide (message francais)

erreur.nom.too\_long=le nom doit contenir moins de 12 caractères. (message francais)

erreur.prenom.too\_long=le prénom doit contenir moins de caractères.(message francais)

Messages dans mesMessages\_en

erreur.nom.vide=name should not be blank (message anglais)

erreur.prenom.vide=firstname should not be blank (message anglais)

erreur.nom.too\_long=name should be contain less thant 12 characters (message anglais)

erreur.prenom.too\_long=firstname should be contain less thant 12 characters (message anglais)

erreur.age.vide=age can not be equal to 0 !!!!!

erreur.age.limit=age must be between 12 and 90 !!!!!!!

Message dans mesMessage

erreur.nom.vide=le nom est vide

erreur.prenom.vide=le prenom est vide

erreur.nom.too\_long=name should be contain less thant 12 characters (message anglais)

erreur.prenom.too\_long=firstname should be contain less thant 12 characters (message anglais)

erreur.age.vide=age can not be equal to 0 !!!!!

erreur.age.limit=age must be between 12 and 90 !!!!!!!

Code Java :

**if**("".equals(nom))

errors.add("erreur.nom",**new** ActionMessage("erreur.nom.vide"));

**else** **if** (nom.length() > 12)

errors.add("erreur.nom.too\_long", **new** ActionMessage("erreur.nom.too\_long"));

**if**("".equals(prenom))

errors.add("erreur.prenom",**new** ActionMessage("erreur.prenom.vide"));

**else** **if**(prenom.length() > 12)

errors.add("erreur.prenom.too\_long", **new** ActionMessage("erreur.prenom.too\_long"));

9°) La gestion de l’internationalisation est géré par Struts par les différents fichiers de langue disponible.

Struts va donc récupérer la langue dans le header de la page transmise