Tutorial Struts

L'objectif de ce Tutorial est de comprendre le fonctionnement de Struts 1.x. à travers la réalisation d'une petite application de gestion de contacts. Les étapes sont plus au moins les mêmes qu'on pourra retrouver dans les livres références sur le sujet ou bien sur les sites/forums pour développeurs. Plus particulièrement, il aborde les points suivants :

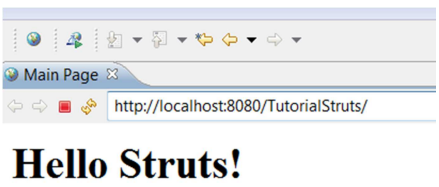
- La mise en place de l'environnement Struts 1 .x autour d'Eclipse avec un accès à une base de données (MySQL ici)
- L'application du pattern architectural MVC avec Struts 1.x
- Les briques de bases du fichier struts-config
- Comment écrire une Action Form qui valide un formulaire
- Comment écrire des servlets Action
- Comment utiliser l'i18n (du moins le principe)
- Comment déclarer une DataSource (resource) et l'invoquer/récupérer avec jndi
- Optionnellement, traiter les exceptions

Etape 1 : Mise en place de l'environnement

- **Important : Commencer par faire les étapes décrites dans la partie Requirements de ce document.**
- Créez ensuite dans Eclipse : File ->New->Other..>Web->**Dynamic Web Project**
    - Nommez votre projet **TutorialStruts**
    - Choisissez la **version 2.5** pour l'option « Dynamic web module version » puis validez
    - Créez sous le WebContent de votre projet, un répertoire et nommez-le **pages**. On y mettra toutes nos pages jsp
    - Créez une page jsp sous **pages**, et nommez-la **main.jsp**. Cette page sera notre point d'entrée à l'application. C'est à partir de là qu'on invoquera les servlets (actions).
    - Indiquez au **web.xml** que cette jsp est votre Welcome-file

```
<display-name>TutorialStruts</display-name>
  <welcome-file-list>
    <welcome-file>pages/main.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

    - Faites que votre jsp ait un titre (exp. Main page) et qu'elle affiche un message de bienvenue.
    - Bouton droit sur votre projet->Run on server
    - Configurer Tomcat s'il le faut, vous devez avoir au niveau de votre browser :



Auteur : Reda Bendraou©

- Jusqu'à présent rien de spécial ! Passons à l'installation de Struts
- Rien de plus simple dézipez le contenu du zip de Struts que vous avez téléchargé sur le site d'apache (voir partie Requirements à la fin du document) et copiez tous les jars dans le répertoire **WEB-INF/lib** de votre projet web

- Pour l'affichage des messages (info, erreur, etc.), Struts privilégie l'utilisation du même principe que celui utilisé pour l'i18n (internationalisation). Le principe est d'éviter d'écrire les messages en dur dans vos jsp, en utilisant les instructions out.print ("messages à afficher") mais plutôt d'écrire ses messages dans des fichiers de propriétés puis d'y faire référence dans vos jsp en utilisant des clés (key) et la taglib <bean> de struts. Exemple, au lieu d'avoir dans votre **main.jsp**

```
<body>
	<h1><% out.print("Hello Struts" ) ;%></h1>
</body>
```

Il faudrait avoir plutôt ça :

```
<body>
	<h1><bean:message key="label.hello"/></h1>

</body>
```

- Pour ce faire, bouton droit sur votre répertoire src (source)-> new file et vous le nommez **Resources.properties**. Rajoutez la ligne suivante :
  - `label.hello=Hello Struts!`
- Ici **label.hello** c'est la clé (key) et l'évaluation de la clé affichera le Hello Struts !
- Voici le code de votre **main.jsp** :

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="bean" uri="http://struts.apache.org/tags-bean" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title><bean:message key="main.page.title"/></title>
</head>
<body>
<h1><bean:message key="label.hello"/></h1>
</body>
</html>
```

- Et le contenu de **Resources.properties**

```
label.hello=Hello Struts!
main.page.title=Main Page
```

- Pour l'instant ne lancez pas l'exécution vu que nous n'avons pas encore mis en place struts.
- Pour cela, créez un fichier **struts-config.xml** sous le répertoire WEB-INF de votre projet. Il doit contenir le code suivant. Pour l'instant il ne contient rien sauf une indication sur le fichier de propriétés que vous allez utiliser pour vos messages.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_3.dtd">

<struts-config>

  <!-- ========== message file ============================ -->
  <message-resources parameter="Resources" />

</struts-config>
```

- Finalement, le **web.xml** doit rediriger les requêtes (passer la main) à struts. Pour cela modifiez le fichier **web. xml** avec le code suivant. Il indique tout simplement que toutes les requêtes ***.do** seront redirigées vers la servlet principale de Struts, i.e., **ActionServlet**

```
  <!-- Configuration de l'action servlet -->
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
    <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <!-- Action Servlet Mapping -->
<servlet-mapping>
              <servlet-name>action</servlet-name>
<!-- indique que toutes les url .do atterrirons chez l'Action Servlet de Struts -->
              <url-pattern>*.do</url-pattern>
  </servlet-mapping>
```

- Testez en déployant votre application (bouton droit sur le projet ->run on server)
- Créez deux autres fichiers **Resources_fr_FR.properties** et **Resources_fr.properties** toujours dans le même répertoire src (répertoire de source java) qui eux contiendrons les mêmes clés mais les messages en français
- Testez, les messages doivent être en français maintenant (bien sûr seulement si les paramètres locaux de votre système sont fr, FR.)
- Recule sur cette partie :
    - Installer Struts revient à mettre les jars dans **WEB-INF\lib**
    - Le **web.xml** ne sert plus qu'à forwarder les requêtes vers Struts
    - Struts se base sur **struts-config.xml** pour aiguiller les requêtes (on le verra plus dans prochaine section)
    - Les messages ne sont plus codés en dur mais plutôt dans des fichiers .properties

Etape 2 : Pratiquer Struts

Cette partie suppose que vous avez une base de données (nommée **jee** par exemple) qui tourne (sous MySQL) et que la base contient une table **Contact**. (Reportez-vous à la partie Requirements, section **Déclarer une source de données JDBC**, si vous voulez définir une Data Source et la charger avec Tomcat et en utilisant JNDI (Java Naming & Directory Interface, API de connexion à des annuaires, équivalent à LDAP) au lieu de passer par le traditionnel code JDBC).

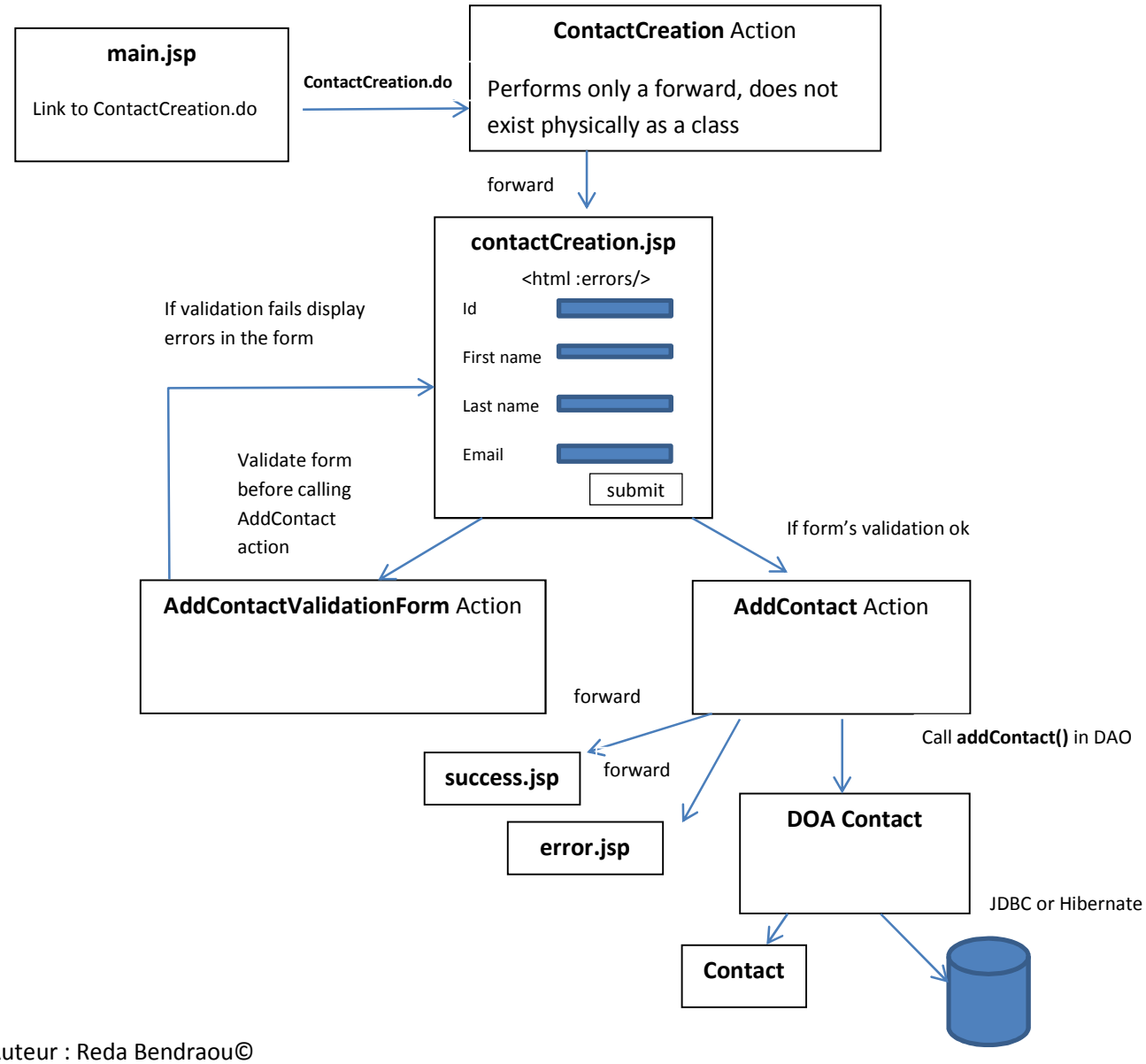
Le DDL pour la création de votre table. Le code de ce tuto suppose que vous avez une base MySQL préalablement créée qui se nomme « **jee** »

Auteur : Reda Bendraou©

```
CREATE TABLE contact (
        ID_CONTACT BIGINT NOT NULL,
        FIRSTNAME VARCHAR(255),
        LASTNAME VARCHAR(255),
        EMAIL VARCHAR(255),
        PRIMARY KEY (ID_CONTACT)
) ENGINE=InnoDB;
```

On commence par appliquer le **forward**. Votre page **main.jsp** doit maintenant avoir un lien vers une servlet qui ne fait que nous forwarder vers une page jsp de rajout d'un nouveau contact. L'idée ici est que toutes vos requêtes doivent être sous forme de **nomRequete.do** même si au final vous ne demandez que l'affichage d'une page jsp. L'objectif derrière est de toujours respecter le MVC imposé par Struts (i.e, servlet pour le contrôle, jsp pour les vues).

Voici le schéma que nous aimerions réaliser pour la suite :



Pour réaliser ce schéma, voici les différentes étapes :

- Création de trois packages sous **src** (répertoire de sources d'Ecplise) :
  - **org.lip6.struts.domain** qui contiendra vos classes métier (ex. Contact) et vos DAO
  - **org.lip6.struts.actionForm** qui contiendra vos Action Forms
  - **org.lip6.struts.servletAction** qui contiendra vos Servlet Actions
- Modifiez votre **main.jsp** pour qu'elle fournisse un lien (href) vers l'action **ContactCreation**

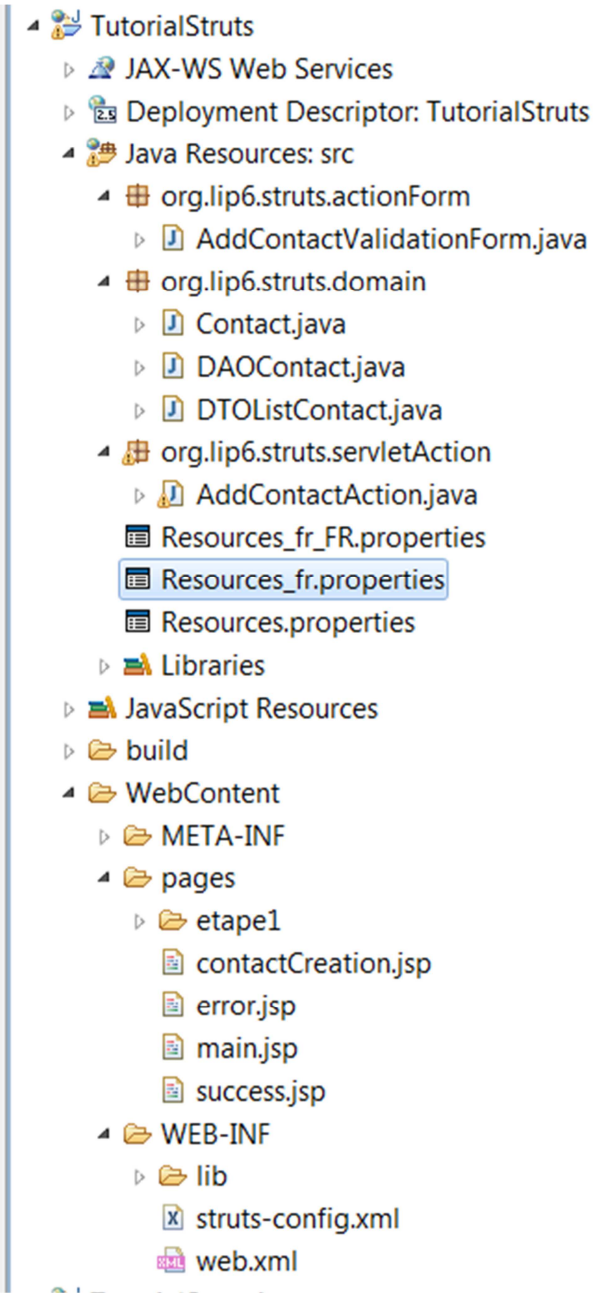
- Modification du **struts-config.xml** pour y inclure l'action (**ContactCreation**) qui ne fait que vous forwarder vers un formulaire d'ajout de contact (**creationContact.jsp**). **Attention**, cette action n'existe pas physiquement, i.e., il n'y pas de classe Java de créée pour cette action.
- Ecriture de la page **creationContact.jsp** pour le formulaire de création.
-

- Une fois que vous cliquez sur submit du formulaire de création, ce dernier devra être validé par une action form (ici, **AddContactValidationForm**)
- Si la validation se passe bien, l'action **AddContact** traitera alors votre requête. Sinon, un message d'erreur est affiché directement sur la page du formulaire (i.e., **creationContact.jsp**, ici utilisation du tag **<html:erros/>**).

- Le rôle **d'AddContact** action est d'instancier le **DAOContact** en lui demandant de créer une instance de **Contact** et de la rendre persistante dans la base.
- Si **AddContact** ne génère pas d'erreurs, vous êtes rediriger vers une jsp (ici **success.jsp**), autrement vers la page d'erreur (ici **error.jsp**)
- **Important :**
  - Une fois que toutes vos pages /classes/ actions créées, il faudra penser à mettre à jour votre fichier **struts-config.xml**, car c'est dans ce fichier que vous décrivez toute cette cinématique. Le mieux c'est de le faire au fur et à mesure. **Attention**, vous ne pouvez pas tester si toute la cinématique n'est pas complétement décrite dans le **struts-config.xml** ! Vous aurez des messages d'erreurs autrement !

- Pensez à mettre à jour vos fichiers **Resources** à chaque ajout d'un nouveau message. C'est source d'erreurs !
- Une fois que vous pensez avoir tout définit, commencez les tests.
- On suppose que votre partie DOA, bd, fonctionne correctement. Sinon, simulez avec un tableau en mémoire pour les contacts et avec des prints.
- Vous devez avoir l'arborescence ci-dessous dans votre projet à la fin de ces étapes.

- Si vous arrivez, à l'aide du cours, à réaliser ces étapes tout seul, c'est très bien ! Essayez au moins à le faire sans regarder la suite. Si vous avez du mal, voici le code des fichiers utilisés pour réaliser cette fonctionnalité :



Pour réaliser cette fonctionnalité, voici le code des différents fichiers, dans l'ordre des étapes à réaliser, décrites juste avant :

- Commencez par modifiez votre **main.jsp** (n'oubliez de mettre à jour le fichier **Resources.properties** , fr et fr_FR, si besoin pour les messages) :

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="html" uri="http://struts.apache.org/tags-html" %>
<%@ taglib prefix="bean" uri="http://struts.apache.org/tags-bean" %>
<%@ taglib prefix="logic" uri="http://struts.apache.org/tags-logic" %>
<%@ taglib prefix="nested" uri="http://struts.apache.org/tags-nested" %>
<html:html>
   <head>
         <title><bean:message key="main.page.title"/></title>

   </head>

   <body>
         <h1><bean:message key="main.page.menu"/></h1>

         <h4><a href="ContactCreation.do"><bean:message
key="main.addcontact.link"/></a></h4><br>

   </body>
</html:html>
```

- Ensuite dans votre répertoire pages, rajoutez la page jsp **creationContact.jsp** qui contiendra le formulaire de création de contact.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="html" uri="http://struts.apache.org/tags-html" %>
<%@ taglib prefix="bean" uri="http://struts.apache.org/tags-bean" %>
<%@ taglib prefix="nested" uri="http://struts.apache.org/tags-nested" %>

   <html:html>

   <head>

   <title><bean:message key="add.contact"/></title>

   <html:base/>

   </head>

   <body bgcolor="white">

   <html:form action="/AddContact">

   <html:errors/>

   <table>

        <tr>

          <td align="center" colspan="2">
```

```
            <font size="4">Please Enter the Following Details</font>
      </tr>
            <tr>
          <td align="right">
            Contact Id
          </td>
          <td align="left">
            <html:text property="id" size="30" maxlength="30"/>
          </td>
        </tr>
            <tr>
          <td align="right">
            First Name
          </td>
          <td align="left">
            <html:text property="firstName" size="30" maxlength="30"/>
          </td>
        </tr>
        <tr>
          <td align="right">
            Last Name
          </td>
          <td align="left">
            <html:text property="lastName" size="30" maxlength="30"/>
          </td>
        </tr>

        <tr>
          <td align="right">
            E-mail address
          </td>
          <td align="left">
            <html:text property="email" size="30" maxlength="30"/>
          </td>
        </tr>
      <tr>
            <td align="right">
              <html:submit>Save</html:submit>
            </td>
    </tr>
  </table>
   </html:form>
   </body>
   </html:html>
```

L'action de validation du formulaire qui se trouve dans le package **org.lip6.struts.actionForm** :

```
package org.lip6.struts.actionForm;

import javax.servlet.http.HttpServletRequest;

import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionMessage;
public class AddContactValidationForm extends ActionForm {
        /**
       *
       */
      private static final long serialVersionUID = 1L;
```

```
private long id=0;
  private String firstName=null;
  private String lastName=null;
  private String email=null;


/**
   * @return Email
   */
  public String getEmail() {
    return email;
  }

  /**
   * @return First Name
   */
  public String getFirstName() {
    return firstName;
  }

  /**
   * @return Last name
   */
  public String getLastName() {
    return lastName;
  }

  /**
   * @param string Sets the Email
   */
  public void setEmail(String string) {
    email = string;
  }

  /**
   * @param string Sets the First Name
   */
  public void setFirstName(String string) {
    firstName = string;
  }

  /**
   * @param string sets the Last Name
   */
  public void setLastName(String string) {
    lastName = string;
  }

  /**
   * @return ID Returns ID
   */
  public long getId() {
    return id;
  }

  /**
   * @param l Sets the ID
   */
  public void setId(long l) {
    id = l;
  }
```

```
        public void reset(ActionMapping mapping, HttpServletRequest request) {
                  this.id=0;
                this.firstName=null;
                this.lastName=null;
                this.email=null;
        }

        public ActionErrors validate(
                  ActionMapping mapping, HttpServletRequest request ) {
                  ActionErrors errors = new ActionErrors();

                  if( getFirstName()== null || getFirstName().length() < 1 ) {
                    errors.add("first name",new
ActionMessage("creation.fn.error.required"));
                  }
                  if( getLastName()== null || getLastName().length() < 1 ) {
                    errors.add("last name",new
ActionMessage("creation.ln.error.required"));
                  }
                  if( getEmail() == null || getEmail().length() < 1 ) {
                    errors.add("email", new
ActionMessage("creation.email.error.required"));
                  }
                  return errors;
              }
}
```

La servlet action **AddContact** qui se trouve dans le package **org.lip6.struts.servletAction** :

```
package org.lip6.struts.servletAction;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.Globals;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionMessage;
import org.apache.struts.action.ActionMessages;
import org.lip6.struts.actionForm.AddContactValidationForm;
import org.lip6.struts.domain.DAOContact;

public class AddContactAction extends Action {


      public ActionForward execute(final ActionMapping pMapping,
                  ActionForm pForm, final HttpServletRequest pRequest,
                  final HttpServletResponse pResponse) {

            final AddContactValidationForm
lForm=(AddContactValidationForm)pForm;

            final long id = lForm.getId();

            final String firstName = lForm.getFirstName();
            final String lastName = lForm.getLastName();
```

```
            final String email = lForm.getEmail();

            // create a new Contact
            final DAOContact lDAOContact = new DAOContact();
            final String lError = lDAOContact.addContact(id, firstName,
lastName, email);



            if(lError == null) {
                  // if no exception is raised,  forward "success"
                  return pMapping.findForward("success");
            }
            else {
                  // If any exception, return the "error" forward
                  return pMapping.findForward("error");
            }
      }
}
```

Les classes Métier + DAO. Attention **DAOContact** utilise une source de données localisée à l'aide de JNDI (voir la partie Requirements). Vous devez réécrire cette partie avec du code JDBC i.e., l'étape de chargement du pilote, connexion, etc., si vous n'utilisez pas JNDI.

```
package org.lip6.struts.domain;
package org.lip6.struts.domain;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.LinkedList;
import java.util.List;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;

public class DAOContact {
private final static String RESOURCE_JDBC = "java:comp/env/jdbc/dsMyDB";



      public String addContact(final long id, final String firstName, final
String lastName, final String email) {
            try {
                  final Context lContext = new InitialContext();
                  final DataSource lDataSource = (DataSource)
lContext.lookup(RESOURCE_JDBC);
                  final Connection lConnection  = lDataSource.getConnection();

                  // adding a new contact
                  final PreparedStatement lPreparedStatementCreation =
```

```
                    lConnection.prepareStatement("INSERT INTO CONTACT(ID_CONTACT,
FIRSTNAME, LASTNAME, EMAIL) VALUES(?, ?, ?, ?)");

                    lPreparedStatementCreation.setLong(1, id);
                    lPreparedStatementCreation.setString(2, firstName);
                    lPreparedStatementCreation.setString(3, lastName);
                    lPreparedStatementCreation.setString(4, email);
                    lPreparedStatementCreation.executeUpdate();

                    return null;
              } catch (NamingException e) {

                    return "NamingException : " + e.getMessage();

              } catch (SQLException e) {

                    return "SQLException : " + e.getMessage();

              }
        }


}
```

La classe **Contact** :

```
package org.lip6.struts.domain;


public class Contact {
        private long id;
        private String firstName;
        private String lastName;
        private String email;


       /**
          * @return Email
          */
         public String getEmail() {
           return email;
         }

         /**
          * @return First Name
          */
         public String getFirstName() {
           return firstName;
         }

         /**
          * @return Last name
          */
         public String getLastName() {
           return lastName;
         }

         /**
          * @param string Sets the Email
          */
```

```
    public void setEmail(String string) {
      email = string;
    }

    /**
     * @param string Sets the First Name
     */
    public void setFirstName(String string) {
      firstName = string;
    }

    /**
     * @param string sets the Last Name
     */
    public void setLastName(String string) {
      lastName = string;
    }

    /**
     * @return ID Returns ID
     */
    public long getId() {
      return id;
    }

    /**
     * @param l Sets the ID
     */
    public void setId(long l) {
      id = l;
    }

  }
```

Les pages **success.jsp** et **error.jsp**

success.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="html" uri="http://struts.apache.org/tags-html" %>
<%@ taglib prefix="bean" uri="http://struts.apache.org/tags-bean" %>
<html:html>
      <head>

            <title><bean:message key="success"/></title>
      </head>
      <body>

            <bean:message key="contact.add"/>
      </body>
</html:html>
```

error.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="html" uri="http://struts.apache.org/tags-html" %>
<%@ taglib prefix="bean" uri="http://struts.apache.org/tags-bean" %>
```

```
<html:html>
	<head>
		<title><bean:message key="title.error"/></title>
	</head>
	<body>
		<html:errors/><br/>
	</body>
</html:html>
```

Resources.properties (même chose pour **Resources_fr_FR.properties, Resources_fr.properties** où les messages sont en français)

```
label.hello=Hello Struts!
main.page.title=Main Page
main.page.menu=Main Menu
add.contact=Add a new Contact
title.error=Error!!!
contact.add=Contact is properly added into the database!
success=success!
creation.id.error.required=Contact's id is required
creation.fn.error.required=Contact's fist name is required
creation.ln.error.required=Contact's last name is required
creation.email.error.required=Contact's email is required
main.addcontact.link=Add Contact with Action Form Validation Class
```

Le plus important, le fichier **struts-config.xml**

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_3.dtd">

<struts-config>
	<form-beans>
		<form-bean name="AddContactValidationForm"
type="org.lip6.struts.actionForm.AddContactValidationForm"/>
	</form-beans>
	<action-mappings>

	<action path="/ContactCreation" forward="/pages/contactCreation.jsp" />

	<action path="/AddContact"
				type="org.lip6.struts.servletAction.AddContactAction"
				name="AddContactValidationForm" scope="request"

				input="/pages/contactCreation.jsp">
			<forward name="success" path="/pages/success.jsp" />
			<forward name="error" path="/pages/error.jsp" />
	</action>
	</action-mappings>

<!-- ========== Ressources de definitions de messages
============================ -->
	<message-resources parameter="Resources" />

</struts-config>
```

Finalement le **web.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">
  <display-name>TutorialStruts</display-name>
  <welcome-file-list>
    <welcome-file>pages/main.jsp</welcome-file>
  </welcome-file-list>
    <!-- Configuration de l'action servlet -->
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
    <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
    <!-- The below example tells the container that the servlet will be loaded
at first.
    The <load-on-startup> sub-element indicates the order in which each servlet
should be loaded.
    Lower positive values are loaded first. If the value is negative or
unspecified,
    then the container can load the servlet at anytime during startup -->
    <load-on-startup>1</load-on-startup>
  </servlet>
  <!-- Action Servlet Mapping -->
      <servlet-mapping>
            <servlet-name>action</servlet-name>
            <url-pattern>*.do</url-pattern>
      </servlet-mapping>
            <!-- declaring a JDBC data source -->
      <resource-ref>
            <description>JDBC resource to access my DB i.e.
dfj_hibernate</description>
            <res-ref-name>jdbc/dsMyDB</res-ref-name>
            <res-type>javax.sql.DataSource</res-type>
            <res-auth>Container</res-auth>
      </resource-ref>
</web-app>
```

Etape 3 : Maitrisez Struts

A vous de jouer ! Rajoutez de nouvelles fonctionnalités si vous voulez vraiment apprendre Struts :

- Afficher la liste des Contacts
- Supprimer un Contact
- Modifiez un Contact

Requirements :

Auteur : Reda Bendraou©

Ce Tp a été réalisé avec les outils suivants :

Eclipse Helios

Tomcat 6

Struts 1.3

Télécharger la dernière release d'Eclipse version JEE développeur à l'adresse suivante : http://www.eclipse.org -> downloads->Eclipse for JEE dev. Cette version doit contenir WTP (Web Tool Platform)

Télécharger la dernière version du JDK (si ce n'est pas déjà installé sur votre machine !)
Sur : http://developers.sun.com/downloads/

Après installation (de préférence directement sur C: ou bien C:\MonRep\JDK1.X), assurez-vous d'avoir une variable d'environnement JAVA_HOME qui pointe vers le répertoire d'installation du JDK.

Rajoutez à la variable PATH de votre environnement l'entrée suivante : %JAVA_HOME%\bin

Télécharger la dernière version de Tomcat et déziper là quelque part sur votre disque. Avoir un c:/tomcatNumVersion/ serait pas mal.

Télécharger la dernière version de MySQL database server Sur : http://dev.mysql.com/downloads/
Pour pouvoir y accéder partout depuis une fenêtre Dos, rajouter une entrée dans votre variable d'environnement PATH : c:\MonRepInstall\...\MySQL Server.NumVersion\bin;

Lors de l'installation faire attention au login/mot de pass à définir pour la base. Choisir par exemple root (login) et root (password) ou bien root(login) et (rien du tout) comme password

Télécharger la dernière version du connector (driver) MySQL pour Java sur :
http://dev.mysql.com/downloads/connector/j/

Décompressez le .zip du connector MySQL et placez-le mysql-connector-java-numVersion-bin.jar du connector JDBC dans le repertoire **WebContent/WEB-INF/lib** de votre projet web

Finalement, bouton droit sur votre projet web-> properties-> Java Build Path -> Libraries -> add External Jars -> puis pointer vers les servlet*.jar et servlet*jsp*.jar qui se trouvent dans le répertoire plugin de votre éclipse. Autrement, vous pouvez les copier directement dans le **WebContent/WEB-INF/lib** de votre projet web.

Téléchargez la dernière version stable de Struts 1.x (un fichier zip, prenez que la version lib, pas besoin de prendre tout struts) sur le site d'apache struts

http://struts.apache.org/downloads.html

Déclarer la source de données JDBC pour MySql

Auteur : Reda Bendraou©

Modifiez **server.xml** de Tomcat (dans Eclipse) (important, modifiez le nom de la base, user, password selon votre config.)

```
……
<!-- Global JNDI resources
       Documentation at /docs/jndi-resources-howto.html
  -->
  <GlobalNamingResources>
    <!-- Editable user database that can also be used by
         UserDatabaseRealm to authenticate users
    -->
    <Resource auth="Container" driverClassName="com.mysql.jdbc.Driver"
maxActive="100" maxIdle="30" maxWait="10000" name="jdbc/dsMyDB" password=""
type="javax.sql.DataSource" url="jdbc:mysql://localhost:3306/jee"
username="root"/>

    <Resource auth="Container" description="User database that can be updated
and saved" factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
name="UserDatabase" pathname="conf/tomcat-users.xml"
type="org.apache.catalina.UserDatabase"/>
  </GlobalNamingResources>…….
```

Modifiez le fichier **context.xml** de Tomcat dans Eclipse

```
<!-- Default set of monitored resources -->
    <WatchedResource>WEB-INF/web.xml</WatchedResource>
    <!-- added for the Struts TP -->
    <ResourceLink name="jdbc/dsMyDB" global="jdbc/dsMyDB"
type="javax.sql.DataSource"/>
    <!-- Uncomment this to disable session persistence across Tomcat restarts --
>
```

Copier le connecteur mysql dans le répertoire **tomcat-numVersion-Repertoire/lib** de Tomcat

Très IMPORTANT: Redémarrez Eclipse

Modifiez le **web.xml** de votre projet afin d'indiquer à Tomcat la datasource à charger :

```
…..
      <!-- declaring a JDBC data source -->
      <resource-ref>
            <description>JDBC resource to access my DB i.e.
            dfj_hibernate</description>
            <res-ref-name>jdbc/dsMyDB</res-ref-name>
            <res-type>javax.sql.DataSource</res-type>
            <res-auth>Container</res-auth>
      </resource-ref>
</web-app>
```

Si vous aviez du code JDBC à l'ancienne dans votre **DAOContact**, vous pouvez le modifier comme suit :

```
package org.lip6.struts.domain;
package org.lip6.struts.domain;
```

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.LinkedList;
import java.util.List;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;

public class DAOContact {
private final static String RESOURCE_JDBC = "java:comp/env/jdbc/dsMyDB";


      public String addContact(final long id, final String firstName, final
String lastName, final String email) {
            try {
                  final Context lContext = new InitialContext();
                  final DataSource lDataSource = (DataSource)
lContext.lookup(RESOURCE_JDBC);
                  final Connection lConnection  = lDataSource.getConnection();

                  // adding a new contact
                  final PreparedStatement lPreparedStatementCreation =

                  lConnection.prepareStatement("INSERT INTO CONTACT(ID_CONTACT,
FIRSTNAME, LASTNAME, EMAIL) VALUES(?, ?, ?, ?)");

                  lPreparedStatementCreation.setLong(1, id);
                  lPreparedStatementCreation.setString(2, firstName);
                  lPreparedStatementCreation.setString(3, lastName);
                  lPreparedStatementCreation.setString(4, email);
                  lPreparedStatementCreation.executeUpdate();

                  return null;
            } catch (NamingException e) {

                  return "NamingException : " + e.getMessage();

            } catch (SQLException e) {

                  return "SQLException : " + e.getMessage();

            }
      }


}
```