
Framework

Struts 2

Table des matières

I - Struts2 (présentation).....	3
1. Struts 2 : Historique et présentation.....	3
1.1. Historique de Struts 1 et 2.....	3
1.2. Struts 2 et MVC.....	4
II - Installation et configuration de Struts2.....	5
1. Struts 2 : installation et configuration.....	5
1.1. Librairies de Struts2.....	5
1.2. Configuration de Struts2.....	7
III - Fonctionnement de Struts2.....	10
1. Struts 2 : principes de fonctionnement.....	10
1.1. Fonctionnement élémentaire de Struts2.....	10
1.2. Exemple simple (calcul tva via struts2).....	11

IV - Formulaire/actions CRUD et Session.....	15
1. CRUD avec Struts 2.....	15
2. Session utilisateur avec Struts 2.....	21
V - Struts2 : fonctionnalités diverses et avancées.....	25
VI - Annexe – Essentiel de Struts 1.....	26
1. Présentation de STRUTS 1.....	26
1.1. Valeur ajoutée de STRUTS :.....	26
1.2. Architecture de STRUTS :.....	26
1.3. Super Contrôleur ActionServlet et configuration associée.....	28
1.4. ActionForm Bean et Validation des saisies.....	28
1.5. Action Bean.....	29
1.6. Considérations sur le multi-threading.....	31
1.7. Librairie de Tags STRUTS prédéfinis pour pages JSP.....	32
1.7.a. internationalisation.....	32
1.7.b. Accès simplifié au "FormBean" lié à la page courante.....	33
1.7.c. Affichage de valeurs dans une page jsp.....	33
2. Présentation rapide des fonctionnalités avancées.....	34
2.1. Struts Validator.....	34
2.2. Modules.....	34
2.3. Tiles.....	34
2.4. JSTL et STRUTS_EL.....	35
3. Struts - configuration de base.....	35
3.1. Exemple de configuration d'un projet STRUTS sous eclipse.....	35
4. Positionnement STRUTS % frameworks concurrents.....	37
VII - Annexe – Bibliographie, Liens WEB + TP.....	38
1. Bibliographie et liens vers sites "internet".....	38
2. TP.....	38

I - Struts2 (présentation)

1. Struts 2 : Historique et présentation



1.1. Historique de Struts 1 et 2

Struts 1 fut un **projet très populaire** de la communauté "Jakarta Apache" dans les années "**2000-2004**". Créé par [Craig McClanahan](#), **Struts 1.x** était à l'époque le **framework "java/web MVC"** de référence.

Né de la fusion entre l'héritage de Struts 1.X et le framework **WebWork**, **Struts 2** est à considérer comme une **profonde refonte de l'ancien framework offrant plus de puissance, de fonctionnalités et de souplesse**. **Struts 2.x** dont la première version est sortie en **2007** est donc très différent de Struts 1.x.

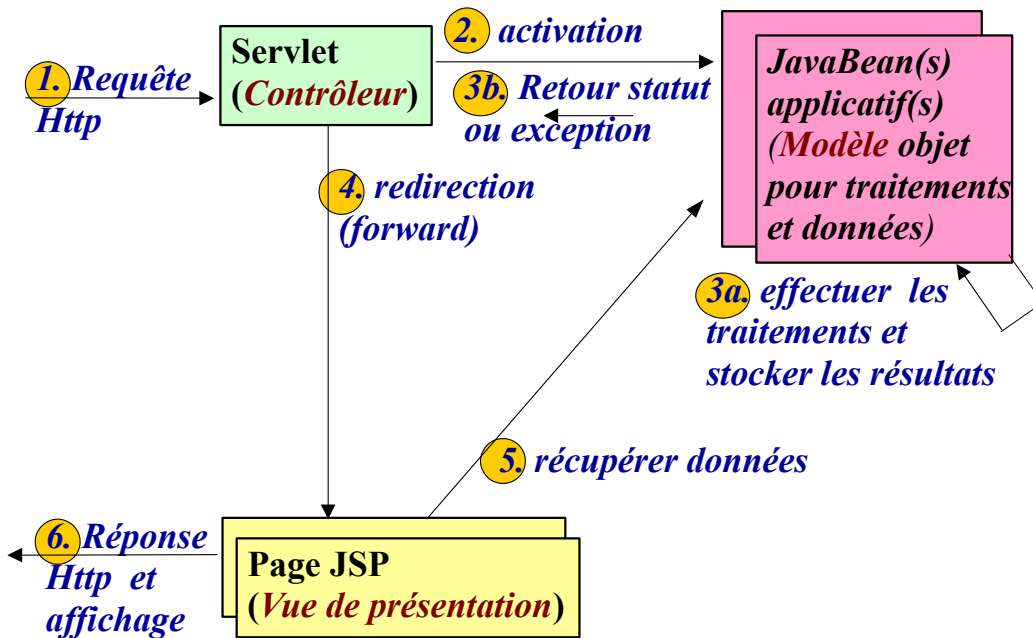
NB : Ce support de cours correspond à la version "Struts2"

Certains éléments de "Struts 1" sont placés en annexes (à titre de comparaisons)

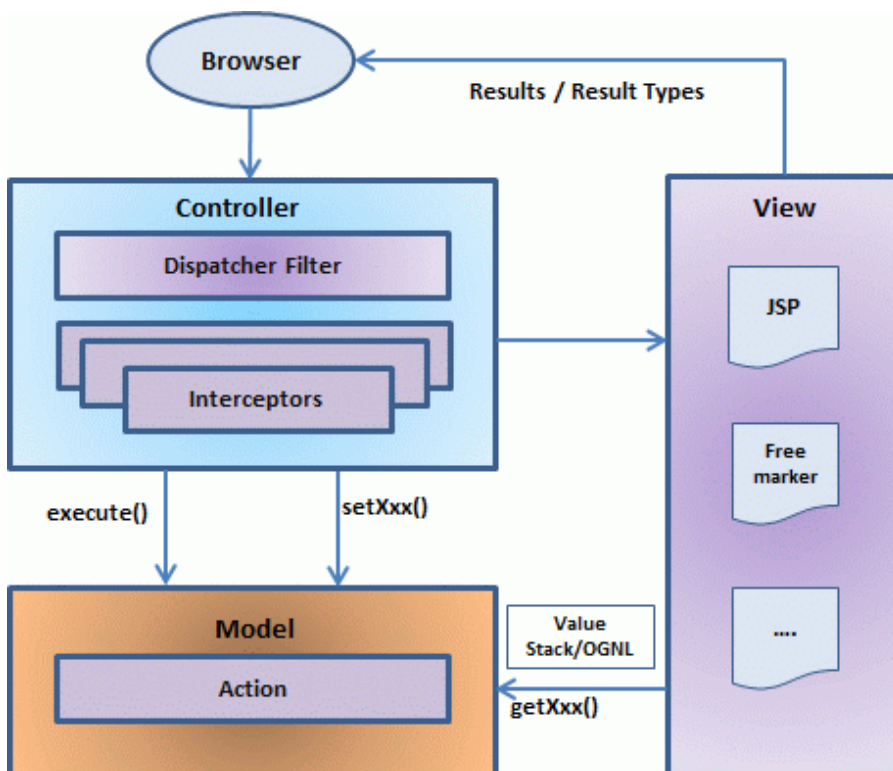
1.2. Struts 2 et MVC

MVC basique sans struts2 (avec Servlet et JSP) :

MVC(2) : "Servlet + page JSP + JavaBean"



MVC avec Struts2 :



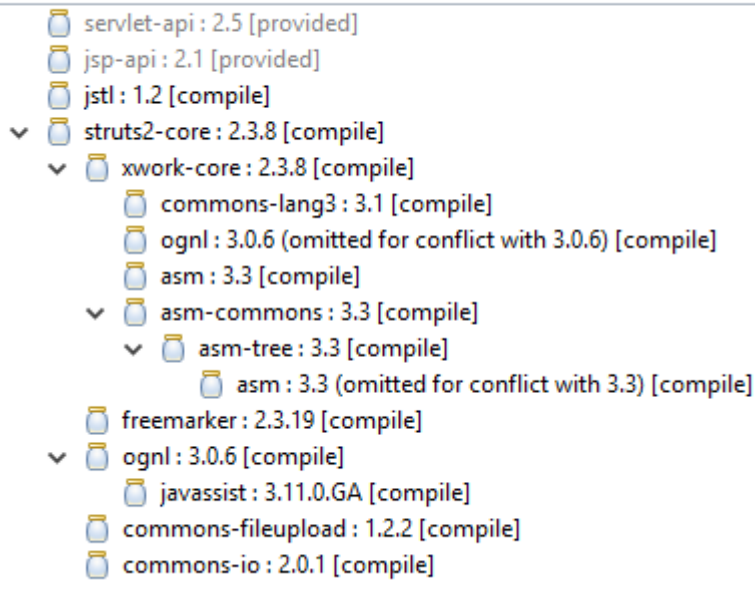
II - Installation et configuration de Struts2

1. Struts 2 : installation et configuration

1.1. Librairies de Struts2

Principale librairies de la version 2.3.8 :

Dependency Hierarchy



Configuration maven (avec Struts 2.3.8 , servlet-api 2.5 , jdk 1.6) :

pom.xml

```

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>tp</groupId>
  <artifactId>struts2WebApp</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>

  <dependencies>

    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>servlet-api</artifactId>
      <version>2.5</version>
    
```

```
        <scope>provided</scope>
    </dependency>

    <dependency>
        <groupId>javax.servlet.jsp</groupId>
        <artifactId>jsp-api</artifactId>
        <version>2.1</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jstl</artifactId>
        <version>1.2</version>
    </dependency>

    <dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts2-core</artifactId>
        <version>2.3.8</version>
    </dependency>

</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.5</version>
            <configuration>
                <source>1.6</source>
                <target>1.6</target>
            </configuration>
        </plugin>
    </plugins>
    <finalName>${project.artifactId}</finalName>
</build>
</project>
```

1.2. Configuration de Struts2

WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

  <display-name>struts2WebApp</display-name>
  <filter>
    <filter-name>struts2</filter-name>
    <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter
    </filter-class>
  </filter>

  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/s2/*</url-pattern>
  </filter-mapping>

</web-app>
```

NB1 : La valeur de url-pattern peut être /* (si namespace = "/") dans struts.xml

ou bien "/s2/*" (si namespace = "/s2") dans struts.xml

ou encore une autre

NB2 : les pages jsp associées à la partie "struts2" de l'application pourront éventuellement être rangées dans un répertoire "pages" ou "jsp" ou "s2-jsp" :

```

▼ s2-jsp
  tva_input.jsp
  tva_result.jsp
  welcome.jsp
▼ WEB-INF
  web.xml
  index.html
```

struts.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
<constant name="struts.enable.DynamicMethodInvocation" value="false" />
<constant name="struts.devMode" value="false" />

<package name="tp.web.actions" namespace="/s2" extends="struts-default">
  <default-action-ref name="welcome"/>

  <action name="welcome">
    <result>/s2-jsp/welcome.jsp</result>
  </action>

  <action name="saisir_tva">
    <result>/s2-jsp/tva_input.jsp</result>
  </action>

  <action name="calculer_tva" class="tp.web.actions.CalculTvaAction"
    method="calculer">
    <result name="success">/s2-jsp/tva_result.jsp</result>
    <result name="input">/s2-jsp/tva_input.jsp</result>
  </action>

</package>
</struts>
```

NB1 : le fichier **struts.xml** doit pouvoir être trouvé à la racine du **classpath** (ex : dans "src" ou "src/main/java" ou "src/main/resources" au sein d'un projet **eclipse** et/ou **maven** puis dans "**WEB-INF/classes**" dans le ".war" généré et à déployer dans tomcat) . Autrement dit , struts.xml n'est **pas** juste à coté de web.xml !!!

index.html (exemple)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<!-- <meta http-equiv="refresh" content="1; URL=s2/welcome.action"> -->
<title>index</title>
</head>
<body>
  <a href="s2/welcome.action">welcome (struts2)</a> <br/>
</body>
</html>
```


s2-jsp/welcome.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!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>welcome</title>
</head>
<body>
<h2>Welcome in struts2 web app</h2>
<ul>
<li>
    <s:url id="saisir_tva_url" action="saisir_tva"></s:url>
    <s:a href="%{saisir_tva_url}">tva</s:a>
</li>
<li>...</li>
</ul>
</body>
</html>
```

NB1: `<%@ taglib prefix="s" uri="/struts-tags" %>` à placer dans le haut de la page jsp permet d'utiliser des balises préfixées par "s :".

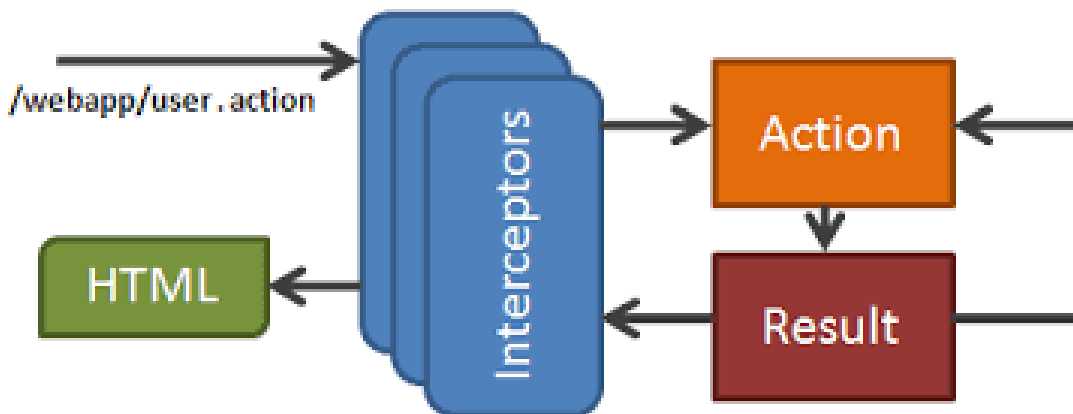
NB2: `s:url` permet de construire l'url relative `saisir_tva_url` à partir de l'action `saisir_tva` déclarée dans `struts.xml`

NB3: `<s:a href="%{saisir_tva_url}">...</s:a>` correspond à la version "struts2" d'un lien hypertexte vers une action .

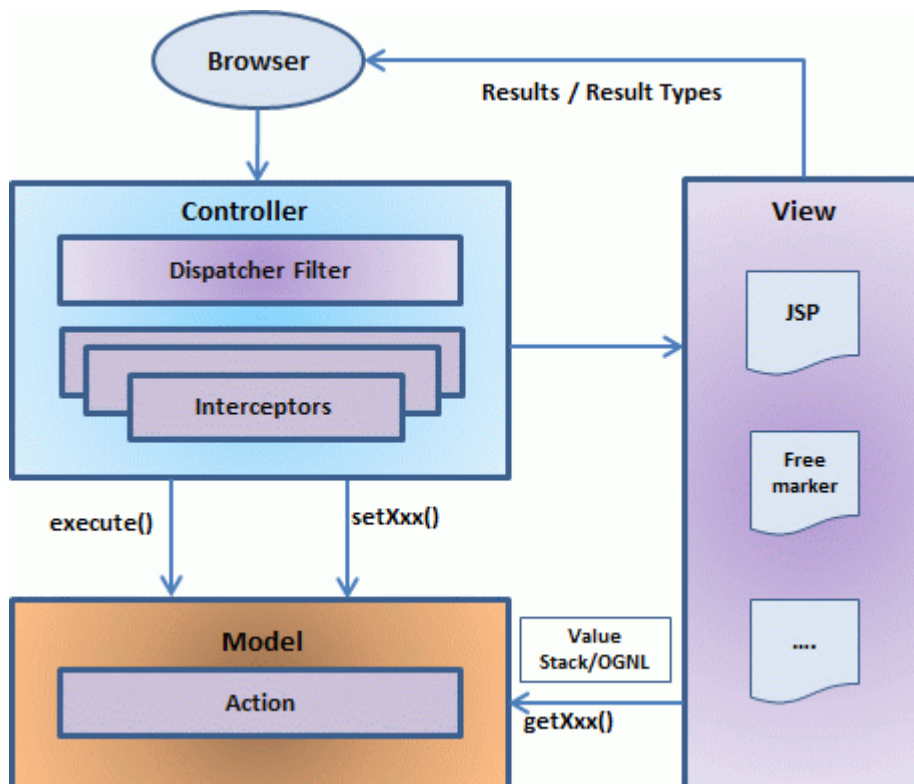
III - Fonctionnement de Struts2

1. Struts 2 : principes de fonctionnement

1.1. Fonctionnement élémentaire de Struts2



Le framework "Struts2" correspond à une implémentation du design pattern "**MVC**" (Model vue controller) au sein de laquelle, le contrôleur principal (front controller) est un filtre web "**StrutsPrepareAndExecuteFilter**" à déclarer dans *WEB-INF/web.xml* et où certains "Bean" (dits d'action) correspondent à des sous-contrôleurs avec généralement des beans de données accrochés.



1.2. Exemple simple (calcul tva via struts2)

Une **classe d'action** struts2 doit idéalement :

1. hériter de **ActionSupport**
2. être bien rangée dans un package (ex : tp.web.actions)
3. comporter des get/set public pour accéder à certaines propriétés
4. comporter au moins une méthode public retournant une String (ex : "success" ou "input" ou ...) qui sera prise en compte dans la configuration des routes dans struts.xml .

tp.web.actions.CalculTvaAction

```
package tp.web.actions;

import com.opensymphony.xwork2.ActionSupport;

public class CalculTvaAction extends ActionSupport{

    private static final long serialVersionUID = 1L;

    private String stHt;
    private String stTaux;//en %
    private Double tva;
    private Double ttc;

    public String getStHt() {
        return stHt;
    }
    public void setStHt(String stHt) {
        this.stHt = stHt;
    }
    public String getStTaux() {
        return stTaux;
    }
    public void setStTaux(String stTaux) {
        this.stTaux = stTaux;
    }
    public Double getTva() {
        return tva;
    }
    public void setTva(Double tva) {
        this.tva = tva;
    }
    public Double getTtc() {
        return ttc;
    }
    public void setTtc(Double ttc) {
        this.ttc = ttc;
    }
}
```

```

public String calculer() {
    System.out.println("Dans la méthode calculer ...");

    if(this.stHt.isEmpty()) {
        System.out.println("le champ ht ne doit pas être vide...");
        return "input";
    }
    if(this.stTaux.isEmpty()) {
        System.out.println("le champ taux ne doit pas être vide...");
        return "input";
    }

    try {
        this.tva = Double.parseDouble(this.stHt) *
            Double.parseDouble(this.stTaux) / 100;
        this.ttc = Double.parseDouble(this.stHt) + this.tva;
        System.out.println("Sucess.....");
        return "success";
    } catch (Exception e) {
        System.out.println("une erreur de calcul ..." + e.getMessage());
        return "input";
    }
}
}

```

dans struts.xml

```

...
<package name="tp.web.actions" namespace="/s2" extends="struts-default">
...
    <action name="saisir_tva">
        <result>/s2-jsp/tva_input.jsp</result>
    </action>

    <action name="calculer_tva" class="tp.web.actions.CalculTvaAction"
        method="calculer">
        <result name="success">/s2-jsp/tva_result.jsp</result>
        <result name="input">/s2-jsp/tva_input.jsp</result>
    </action>
</package>
</struts>

```

s2-jsp/tva_input.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!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>tva_input</title>
</head>
<body>
<h2>tva (input)</h2>
<div id="formulaire">
    <s:form method="post" action="calculer_tva">

        <s:textfield name="stHt" id="ht"
            label="ht" labelposition="left">
        </s:textfield>

        <s:textfield name="stTaux" id="taux"
            label="taux" labelposition="left">
        </s:textfield>

        <s:submit value="calculer"></s:submit>
    </s:form>
</div>
</body>
</html>

```

tva (input)

ht:	<input type="text" value="200"/>
taux:	<input type="text" value="20"/>
<input type="button" value="calculer"/>	

Points à bien respecter :

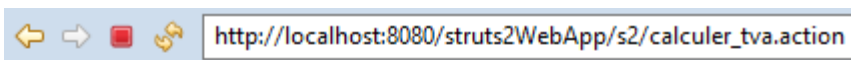
- la valeur de action="" de la balise <s:form > doit correspondre à un nom d'action de struts.xml
- les "name" des <s:textfield > doivent correspondre aux propriétés du bean d'action et ces propriétés doivent idéalement être de type "String" pour pouvoir bien gérer les éventuelles erreurs de saisies .
-

s2-jsp/tva-result.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!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>tva_input</title>
</head>
<body>
<h2>tva (result)</h2>
<div id="divTvaResult">
    ht: <s:property value="stHt" /> <br/>
    taux: <s:property value="stTaux" /> <br/>
    tva: <s:property value="tva" /> <br/>
    ttc: <s:property value="ttc" /> <br/>
</div>
<hr/>
<s:url id="saisir_tva_url" action="saisir_tva"></s:url>
<s:a href="%{saisir_tva_url}">nouveau calcul tva</s:a><br/>
</body>
</html>

```


tva (result)

ht: 200
 taux: 20
 tva: 40.0
 ttc: 240.0

[nouveau calcul tva](#)

IV - Formulaires/actions CRUD et Session

1. CRUD avec Struts 2

tp.web.data.User

```
package tp.web.data;

public class User {
    private Long id;
    private String name;
    private String gender; //"Male" or "Female"
    private String country; //"France", "USA", "UK", ...
    private String aboutYou;
    private Boolean wishToBeInMailingList;

    public User() {
        super();
    }

    //+get/set
}
```

tp.simu.dao.UserDao

```
package tp.simu.dao;

import java.util.List;

import tp.web.data.User;

public interface UserDao {

    public void saveOrUpdateUser(User user) ;
    public void removeUser(Long userId) ;
    public List<User> allUser() ;
    public User userId(Long userId) ;
}
```

tp.simu.dao. UserDaoStub

```
package tp.simu.dao;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
```

```
import tp.web.data.User;

public class UserDaoStub implements UserDao{
    private static UserDaoStub uniqueInstance =null;
    public static UserDaoStub getInstance(){
        if(uniqueInstance==null) {
            uniqueInstance=new UserDaoStub();
        }
        return uniqueInstance;
    }

    //ici simulation en mémoire (sans Database) via map(idUser,user)
    private Map<Long,User> userMap = new HashMap<Long,User>();
    private Long lastId=0L; //for auto_incr

    public void saveOrUpdateUser(User user) {
        if(user.getId()==null){
            user.setId(++lastId);
        }
        userMap.put(user.getId(), user);
    }

    public void removeUser(Long userId) {
        userMap.remove(userId);
    }

    public List<User> allUser() {
        return new ArrayList<User>(userMap.values());
    }

    public User userById(Long userId) {
        return userMap.get(userId);
    }
}
```


tp.web.action.UserCrudAction

```

package tp.web.actions;

import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.apache.struts2.ServletActionContext;

import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

import tp.simu.dao.UserDao;
import tp.simu.dao.UserDaoStub;
import tp.web.data.User;

public class UserCrudAction extends ActionSupport implements ModelDriven<User>{

    private UserDao userDao = UserDaoStub.getInstance() ;
        //singleton (dao stub without DB, memoryMap)

    private List<User> userList = new ArrayList<User>();
    private User user = new User();//model bean instance (for input)

    public String refreshList()
    {
        userList = userDao.allUser();
        return SUCCESS;
    }

    public String saveOrUpdate(){
        userDao.saveOrUpdateUser(user);
        return SUCCESS;
    }

    public String delete(){
        HttpServletRequest request = (HttpServletRequest)
            ActionContext.getContext().get( ServletActionContext.HTTP_REQUEST);
        userDao.removeUser(Long.parseLong( request.getParameter("id")));
        return SUCCESS;
    }

    public String selectToEdit(){
        HttpServletRequest request = (HttpServletRequest)
            ActionContext.getContext().get( ServletActionContext.HTTP_REQUEST);
        user = userDao.userById(Long.parseLong( request.getParameter("id")));
        return SUCCESS;
    }

    @Override

```

```

    public User getModel() {
        return user;
    }

    public List<User> getUserList() {
        return userList;
    }

    public void setUserList(List<User> userList) {
        this.userList = userList;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }
}

```

dans **struts.xml**

```

...
<action name="listUser" method="refreshList" class="tp.web.actions.UserCrudAction">
    <result name="success">/s2-jsp/user_crud.jsp</result>
</action>
<action name="selectUserToEdit" method="selectToEdit"
        class="tp.web.actions.UserCrudAction">
    <result name="success">/s2-jsp/user_crud.jsp</result>
</action>
<action name="saveOrUpdateUser" method="saveOrUpdate"
        class="tp.web.actions.UserCrudAction">
    <result name="success" type="redirect">listUser</result>
</action>
<action name="deleteUser" method="delete" class="tp.web.actions.UserCrudAction">
    <result name="success" type="redirect">listUser</result>
</action>
...

```

s2-jsp/user_crud.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!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>users (crud)</title>
</head>
<body>

```

```

<h2>users (crud)</h2>

<s:form action="saveOrUpdateUser">
  <s:push value="user">
    <s:hidden name="id" />
    <s:textfield name="name" label="User Name" />
    <s:radio name="gender" label="Gender"
      list="{ 'Male', 'Female' }" />
    <s:select name="country" list="{ 'India', 'USA', 'UK', 'France' }"
      headerKey="" headerValue="Select"
      label="Select a country" />
    <s:textarea name="aboutYou" label="About You" />
    <s:checkbox name="wishToBeInMailingList"
      label="Would you like to join our mailing list?" />
    <s:submit value="submit (save or update)" />
  </s:push>
</s:form>
<s:debug />
<s:if test="userList.size() > 0">
  <div class="content">
    <table class="userTable" border="1" cellpadding="5px">
      <tr class="even">
        <th>Name</th>
        <th>Gender</th>
        <th>Country</th>
        <th>About You</th>
        <th>wishToBeInMailingList</th>
        <th>Edit</th>
        <th>Delete</th>
      </tr>
      <s:iterator value="userList" status="userStatus">
        <tr class="<s:if test="#userStatus.odd == true">odd</s:if>
          <s:else>even</s:else>">
          <td><s:property value="name" /></td>
          <td><s:property value="gender" /></td>
          <td><s:property value="country" /></td>
          <td><s:property value="aboutYou" /></td>
          <td><s:property value="wishToBeInMailingList" /></td>
          <td>
            <s:url id="editURL" action="selectUserToEdit">
              <s:param name="id" value="%{id}"></s:param>
            </s:url>
            <s:a href="%{editURL}">Select to Edit</s:a>
          </td>
          <td>
            <s:url id="deleteURL" action="deleteUser">
              <s:param name="id" value="%{id}"></s:param>
            </s:url>
            <s:a href="%{deleteURL}">Delete</s:a>
          </td>
        </tr>
      </s:iterator>
    </table>
  </div>
</s:if>

```

```

</div>
</s:if>
<hr/>
<s:url id="welcome_url" action="welcome"></s:url>
<s:a href="%{welcome_url}">retour vers welcome</s:a><br/>
</body>
</html>

```

NB: dans la logique de struts2 , tout est **exprimé en relatif** (au sein de l'écriture de la page jsp)

users (crud)

User Name:

Gender: ☒ Male ☐ Female

Select a country:

About You:

☒ Would you like to join our mailing list?

users (crud)

User Name:

Gender: ☐ Male ☐ Female

Select a country:

About You:

☐ Would you like to join our mailing list?

[\[Debug\]](#)

Name	Gender	Country	About You	wishToBeInMailingList	Edit	Delete
jean Bon	Male	France	gentil	true	Select to Edit	Delete
axelle Air	Female	UK	de bonne humeur	false	Select to Edit	Delete

2. Session utilisateur avec Struts 2

tp.web.data.Produit

```
package tp.web.data;

public class Produit {

    private Long numero;
    private String label;
    private Double prix;
    ...
}
```

tp.web.data.Selection

```
public class Selection {
    Long numProduit;
    Integer quantite=1;
    ...
}
```

tp.web.actions.CaddyAction

```
package tp.web.actions;

import org.apache.struts2.dispatcher.SessionMap;
import org.apache.struts2.interceptor.SessionAware;

import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

...

public class CaddyAction extends ActionSupport
    implements ModelDriven<Selection> , SessionAware{

    private SessionMap<String,Object> sessionMap;

    @Override //from SessionAware (for session injection)
    public void setSession(Map<String, Object> map) {
        sessionMap=(SessionMap)map;
    }

    private List<Selection> listeSelection = null; //caddy
    private Selection selection = new Selection();//model bean instance (for input)

    private ProduitDao prodDao = ProduitDaoStub.getInstance();
    private List<Produit> produitList = new ArrayList<Produit>();

    public String ajouterDansCaddy(){
        getListeSelection().add(selection);
        return SUCCESS;
    }
}
```

```

    }

    public String loadListeProduits(){
        this.setProduitList(prodDao.allProduit()); //store in session
        return SUCCESS;
    }

    public String listerContenuCaddy(){
        this.getListeSelection(); //in session
        return SUCCESS;
    }

    @Override
    public Selection getModel() {
        return selection;
    }

    public List<Selection> getListeSelection() {
        this.listeSelection = (List<Selection>)this.sessionMap.get("caddy");
        if(this.listeSelection==null){
            this.listeSelection = new ArrayList<Selection>();
            this.sessionMap.put("caddy", listeSelection);
        }
        return listeSelection;
    }

    public void setListeSelection(List<Selection> listeSelection) {
        this.listeSelection = listeSelection;
        this.sessionMap.put("caddy", listeSelection);
    }

    public Selection getSelection() {
        return selection;
    }

    public void setSelection(Selection selection) {
        this.selection = selection;
    }

    public List<Produit> getProduitList() {
        this.produitList = (List<Produit>)this.sessionMap.get("produits");
        if(this.produitList==null){
            this.produitList = new ArrayList<Produit>();
            this.sessionMap.put("produits", produitList);
        }
        return produitList;
    }

    public void setProduitList(List<Produit> produitList) {
        this.produitList = produitList;
        this.sessionMap.put("produits", produitList);
    }
}

```

dans **struts.xml**

```
...
<action name="ListeProduitsForCaddy" method="loadListeProduits"
        class="tp.web.actions.CaddyAction">
    <result name="success"/>s2-jsp/add_in_caddy.jsp</result>
</action>

<action name="addProdSelectionInCaddy" method="ajouterDansCaddy"
        class="tp.web.actions.CaddyAction">
    <result name="success"/>s2-jsp/add_in_caddy.jsp</result>
</action>

<action name="listerContenuCaddy" method="listerContenuCaddy"
        class="tp.web.actions.CaddyAction">
    <result name="success"/>s2-jsp/display_caddy_content.jsp</result>
</action>
```

s2-jsp/add_in_caddy.jsp

```
...
<body>
    <h2>add_in_caddy (session)</h2>
    <s:form action="addProdSelectionInCaddy">
        <s:push value="selection">
            <select name="numProduit" id="numProduit">
                <s:iterator value="produitList" status="prodStatus">
                    <option value='<s:property value="numero"/>'>
                        [<s:property value="numero"/>]
                        <s:property value="label"/> -
                        <s:property value="prix"/> euro(s)
                    </option>
                </s:iterator>
            </select>
            <s:select name="quantite" list="{1,2,3,4,5,6,7,8,9,10}"
                headerKey="" headerValue="Select"
                label="Selectionner une quantité" />
            <s:submit value="submit (addInCaddy)" />
        </s:push>
    </s:form>
    <hr/>
    nombre de selection(s) dans le caddy : <s:property value="listeSelection.size()" />
    <hr/>
    <s:url id="listerContenuCaddy_url" action="listerContenuCaddy"></s:url>
    <s:a href="%{listerContenuCaddy_url}">visualiser le contenu du caddy</s:a><br/>
</body>
</html>
```

add_in_caddy (session)

[1] stylo - 1.23 euro(s) ▼

Selectionner une quantité: 3 ▼

submit (addInCaddy)

nombre de selection(s) dans le caddy : 2

[visualiser le contenu du caddy](#)

s2-jsp/display-caddy-content.jsp (version basique pour vérifier session ok)

```
...
<body>
  <h2>caddy content (in session)</h2>
  <div class="content">
    <table class="userTable" border="1" cellpadding="5px">
      <tr class="even">
        <th>numProduit</th> <th>quantite</th> <th>...</th>
      </tr>
      <s:iterator value="listeSelection" status="selectionStatus">
        <tr>
          <td><s:property value="numProduit" /></td>
          <td><s:property value="quantite" /></td> <td>...</td>
        </tr>
      </s:iterator>
    </table>
  </div>
</hr>
  <s:url id="welcome_url" action="welcome"></s:url>
  <s:a href="%{welcome_url}">retour vers welcome</s:a><br/>
</body>
</html>
```

caddy content (in session)

numProduit	quantite	...
1	1	...
2	3	...

[retour vers welcome](#)

NB : `<s:property value="#session.proprieteEnSession" />` permet d'accéder et afficher une chose stockée en session

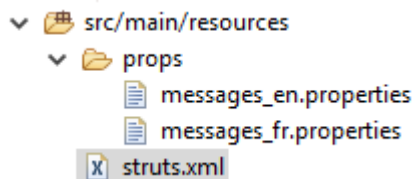
V - Struts2 : fonctionnalités diverses et avancées

1. internationalisation avec Struts 2

Dans **struts.xml**

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
<constant name="struts.custom.i18n.resources" value="props.messages" />
...
```



messages_en.properties

```
vat.input.title=VAT (input)
vat.form.eot=excluded of tax
vat.form.rate=vat rate
vat.form.submit.value=compute
```

messages_fr.properties

```
vat.input.title=TVA (formulaire)
vat.form.eot=ht
vat.form.rate=taux
vat.form.submit.value=calculer
```

s2-jsp/tva_input.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!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>tva_input</title>
</head>
<body>
<h2><s:text name="vat.input.title" />
<!-- <s:property value="%{getText('vat.input.title')}"/> -->
</h2>
```

```

<div id="formulaire">
    <s:form method="post" action="calculer_tva">

        <s:textfield name="stHt" id="ht"
            label="%{getText('vat.form.eot')}}" labelposition="left">
        </s:textfield>

        <s:textfield name="stTaux" id="taux"
            label="%{getText('vat.form.rate')}}" labelposition="left">
        </s:textfield>

        <s:submit value = "%{getText('vat.form.submit.value')}}"></s:submit>
    </s:form>
</div>
</body>
</html>

```

Soit *localité par défaut de l'utilisateur* (selon paramétrage du pc , du navigateur , ... et selon info véhiculée au sein de la requête HTTP) ou bien soit *localité forcée via paramétrage explicite d'un lien hypertexte* :

```

<s:url id="saisir_tva_url_fr" action="saisir_tva">
    <s:param name="request_locale">fr</s:param>
</s:url>
<s:url id="saisir_tva_url_en" action="saisir_tva">
    <s:param name="request_locale">en</s:param>
</s:url>
<s:a href="%{saisir_tva_url_fr}">tva (fr)</s:a> ,
<s:a href="%{saisir_tva_url_en}">vat (en)</s:a>

```

- [tva \(fr\)](#) , [vat \(en\)](#)

TVA (formulaire)	VAT (input)
ht: <input type="text" value="200"/> taux: <input type="text" value="20"/> <input type="button" value="calculer"/>	excluded of tax: <input type="text"/> vat rate: <input type="text"/> <input type="button" value="compute"/>

2. Validation de formulaire avec Struts2

Soit un formulaire d'inscription de ce genre :

inscription (avec validation)

id:	<input type="text" value="1"/>
speudo:	<input type="text" value="power user"/>
age:	<input type="text" value="30"/>
email:	<input type="text" value="user1@gmail.com"/>
numeroEtRue:	<input type="text" value="5 rue elle"/>
codePostal:	<input type="text" value="75000"/>
ville:	<input type="text" value="Paris"/>
dateInscription:	<input type="text" value="12/12/2017"/> x
<input type="button" value="inscrire"/>	

resultat inscription

id: 1
 speudo: power user
 age: 30
 email: user1@gmail.com
 numeroEtRue: 5 rue elle
 codePostal: 75000
 ville: Paris
 dateInscription: 12/12/17

[nouvelle inscription](#)

[retour vers welcome](#)

La version (basique) du code est la suivante :

tp.web.data.Utilisateur

```
public class Utilisateur {
    private Long id;
    private String pseudo;
    private Integer age;
    private String email;
    private String numeroEtRue;
    private String codePostal;
    private String ville;
    private Date dateInscription;

    //+get/set , +constructeur(s) , + toString()
}
```

tp.web.actions.InscriptionAction

```
public class InscriptionAction extends ActionSupport implements ModelDriven<Utilisateur> {

    private static final long serialVersionUID = 1L;

    private Utilisateur utilisateur = new Utilisateur(); //à inscrire

    public String inscrire() {
        System.out.println("Dans la méthode inscrire ...");
        try {
            System.out.println("Sucess.....");
            return "success";
        } catch (Exception e) {
            System.out.println("une erreur a eu lieu ..." + e.getMessage());
        }
    }
}
```

```

        return "input";
    }

    @Override
    public Utilisateur getModel() {
        return this.utilisateur;
    }
}

```

Dans **struts.xml**

```

...
<action name="saisir_inscription">
    <result>/s2-jsp/inscription.jsp</result>
</action>

<action name="inscrire_utilisateur" class="tp.web.actions.Inscription.Action"
    method="inscrire">
    <result name="success">/s2-jsp/inscription_result.jsp</result>
    <result name="input">/s2-jsp/inscription.jsp</result>
</action>
...

```

Dans **s2-jsp/inscription.jsp**

```

...
<s:form method="post" action="inscrire_utilisateur">
    ...

    <s:textfield name="pseudo" id="speudo"
        label="speudo" labelposition="left">
    </s:textfield>

    <s:textfield name="age" id="age"
        label="age" labelposition="left">
    </s:textfield>

    <s:textfield name="email" id="email"
        label="email" labelposition="left">
    </s:textfield>
    ...

```

Sans paramétrage explicite des contraintes de validation, une exception de ce genre est levée en cas de saisie erronée (age=abc par exemple) :

[java.lang.NoSuchMethodException](#):
 tp.web.data.Utilisateur.setAge([Ljava.lang.String;)

→ incompatibilité détectée entre **"abc"** et **setAge(Integer age)**

et par défaut le message d'erreur remonté est le suivant :

Invalid field value for field "age".

age:

Une bonne validation (xml) passe d'abord par la préparation d'un fichier ".properties" pour certains messages d'erreur :

props/messages_fr.properties

```
inscription.erreur.validate.mail=email pas valide
inscription.erreur.validate.codePostal=codePostal non valide (5 chiffres)
inscription.erreur.validate.dateInscription=dateInscription doit être entre 01/01/2017 et 31/12/2020

invalid.fieldvalue.age=age invalide (non numerique)
invalid.fieldvalue.id=id invalide (non numerique)
invalid.fieldvalue.date=date pas valide (jj/MM/aaaa)
```

Il faut ensuite placer juste à coté de la classe d'action un fichier dont le nom exact est

ActionClass-actionName-validation.xml

exemple :

InscriptionAction-inscrire_utilisateur-validation.xml

```
<!DOCTYPE validators PUBLIC
    "-//Apache Struts//XWork Validator 1.0.2//EN"
    "http://struts.apache.org/dtds/xwork-validator-1.0.2.dtd">
<validators>

    <field name="id">
        <field-validator type="required">
            <message><![CDATA[ id is required ]]></message>
        </field-validator>
        <field-validator type="long">
            <param name="min">0</param>
            <message><![CDATA[must be positive ]]></message>
        </field-validator>
    </field>

    <field name="age">
        <field-validator type="required">
            <message><![CDATA[ age is required ]]></message>
        </field-validator>
        <!-- field-validator type="conversion" est declenche si "abc"
            au lieu d'une valeur numerique -->
        <!-- <field-validator type="conversion">
            <message><![CDATA[not an integer !]]></message>
        </field-validator> -->
        <!-- key ="invalid.fieldvalue.age" par défaut et si pas trouvé : i
            nvalid field for age-->
        <!-- <field-validator type="conversion">
            <message key ="invalid.fieldvalue.age"/>
        </field-validator-->
```

```

    </field-validator> -->
    <field-validator type="int">
        <param name="min">0</param>
        <param name="max">150</param>
        <message><![CDATA[ must be between 0 and 150 ]]></message>
    </field-validator>
</field>

<field name="pseudo">
    <field-validator type="requiredstring">
        <message><![CDATA[ pseudo is required ]]></message>
    </field-validator>
    <field-validator type="stringlength">
        <param name="minLength">3</param>
        <message><![CDATA[ at least 3 characters ]]></message>
    </field-validator>
</field>

<field name="email">
    <field-validator type="requiredstring">
        <message><![CDATA[ email is required ]]></message>
    </field-validator>
    <field-validator type="email">
        <!-- <message><![CDATA[ email invalide ]]></message> -->
        <message key="inscription.erreur.validate.mail" />
    </field-validator>
</field>

<field name="codePostal">
    <field-validator type="requiredstring">
        <message><![CDATA[ codePostal is required ]]></message>
    </field-validator>
    <field-validator type="regex">
        <param name="expression">^\d{5}$</param>
        <!-- <message><![CDATA[ codePostal invalide (doit avoir 5
            chiffres) ]]></message> -->
        <message key="inscription.erreur.validate.codePostal" />
    </field-validator>
</field>

<field name="numeroEtRue">
    <field-validator type="requiredstring">
        <message><![CDATA[ numeroEtRue is required ]]></message>
    </field-validator>
</field>

<field name="ville">
    <field-validator type="requiredstring">
        <message><![CDATA[ ville is required ]]></message>
    </field-validator>
</field>

```

```

<field name="dateInscription">
  <field-validator type="required">
    <message><![CDATA[ dateInscription is required ]]></message>
  </field-validator>
  <field-validator type="date">
    <param name="min">01/01/2017</param>
    <param name="max">31/12/2020</param>
    <!-- <message><![CDATA[ date invalide (doit etre entre 01/01/2017 et
                                   31/12/2020) ]]></message> -->
    <message key="inscription.erreur.validate.dateInscription" />
  </field-validator>
</field>

</validators>

```

Principaux validateurs :

required	Pour spécifier une valeur obligatoire (non facultative)
requiredstring	Idem mais pour chaîne de caractères
stringlength	Pour vérifier la longueur d'une chaîne de caractère souvent complémentaire à "required"
int, long , double, ...	Pour vérifier si la valeur est bien est int ou un long ou un double souvent complémentaire à "required" param "min" et "max"
email	Pour vérifier une valeur de type email (xxx@yyy.zzz)
date	Pour vérifier une valeur de type date (jj/MM/aaaa)
regex	Pour vérifier la conformité d'une valeur saisie vis à vis d'une expression régulière

Attention :

Une moindre erreur de nom de champ/"field" peut compromettre le comportement global de la validation.

A des fins de "debug" , on pourra temporairement insérer ceci dans le haut d'une page jsp

```

<s:actionerror />

<s:if test="hasFieldErrors()">
  <div class="errorMessage">
    <s:fielderror />
  </div>
</s:if>

```

de façon à afficher la liste de tous les messages d'erreurs .

inscription (avec validation)

id:

at least 3 characters

pseudo:

must be between 0 and 150

age:

email pas valide

email:

numeroEtRue:

codePostal non valide (5 chiffres)

codePostal:

ville is required

ville:

Invalid field value for field "dateInscription".
dateInscription is required

dateInscription:

inscription (avec validation)

id:

pseudo:

age:

email:

numeroEtRue:

codePostal:

ville:

dateInscription doit être entre 01/01/2017 et 31/12/2020

dateInscription:

Pour obtenir (en plus) une validation en javascript (coté navigateur/client) , il faut placer l'option **validate="true"** au sein de la balise `<s:form ... />`

exemple :

```
<s:form method="post" action="inscrire_utilisateur" validate="true">
  ...
</s:form>
```

D'autre part, pour permettre la génération des parties "struts/...js" , il faut éventuellement élargir les `<filter-mapping>` dans **web.xml** :

```
<filter>
  <filter-name>struts2</filter-name>
  <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-
class>
</filter>
<filter-mapping>
  <filter-name>struts2</filter-name>
  <url-pattern>/s2/*</url-pattern> <!-- pour partie namespace "s2" de cette appli -->
</filter-mapping>
<filter-mapping>
  <filter-name>struts2</filter-name>
  <url-pattern>/struts/*</url-pattern> <!-- pour partie predefinie "struts" et validation (.js) -->
</filter-mapping>
```


inscription (avec validation)

id:

at least 3 characters

pseudo:

must be beetween 0 and 150

age:

email:

numeroEtRue:

codePostal:

ville:

dateInscription:

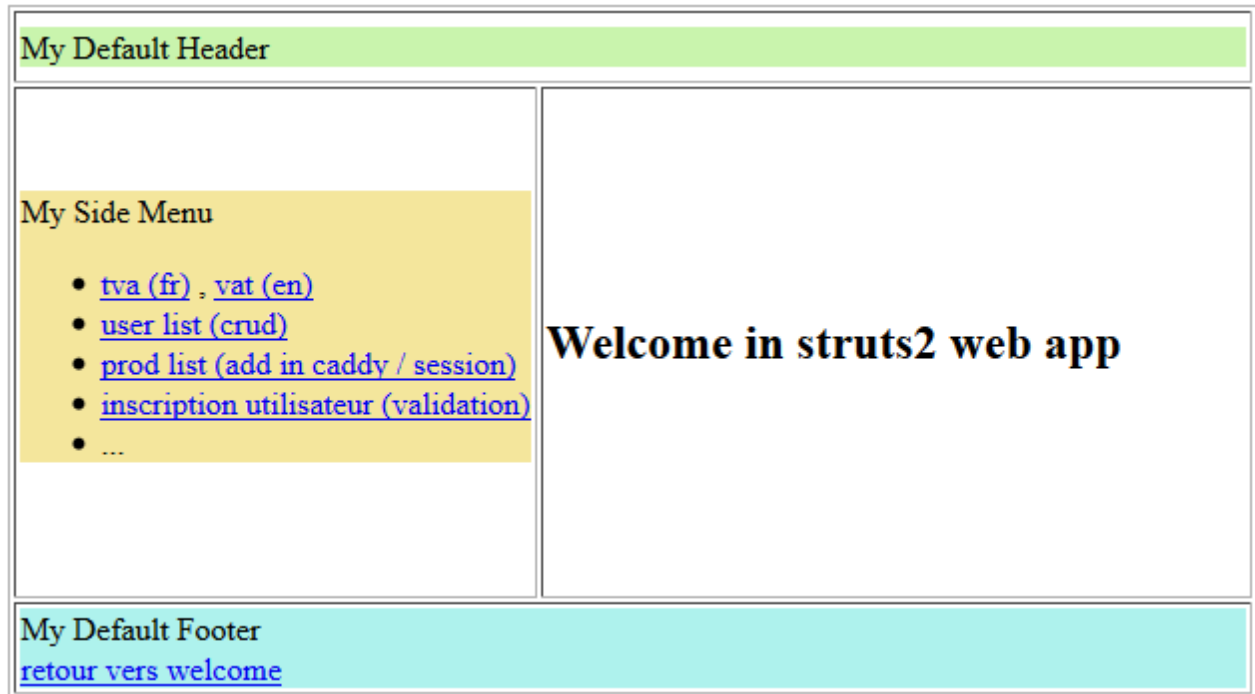
via code javascript généré automatiquement :

```
...
<script type="text/javascript" src="/struts2WebApp/struts/xhtml/validation.js"></script>
<script type="text/javascript" src="/struts2WebApp/struts/utls.js"></script>

<form id="inscrire_utilisateur" name="inscrire_utilisateur"
  onsubmit="return validateForm_inscrire_utilisateur();"
  action="/struts2WebApp/s2/inscrire_utilisateur.action" method="post"
  onreset="clearErrorMessages(this);clearErrorLabels(this);"
...
<script type="text/javascript">
  function validateForm_inscrire_utilisateur() { ... }
</script>
```

3. Template / Tiles

Les **tiles** de struts correspondent à une **extension** permettant de **réutiliser des modèles de mises en pages avec héritages et redéfinitions**.



dans **pom.xml**

```
...
<dependency>
  <groupId>org.apache.struts</groupId>
  <artifactId>struts2-tiles-plugin</artifactId>
  <version>2.3.8</version>
</dependency>
...
```

dans WEB-INF/**web.xml**

```
...
<context-param>
  <param-name>org.apache.tiles.impl.BasicTilesContainer.DEFINITIONS_CONFIG</param-name>
  <param-value>/WEB-INF/tiles.xml</param-value>
</context-param>

  <listener>
    <listener-class>org.apache.struts2.tiles.StrutsTilesListener</listener-class>
  </listener>
...
```

WEB-INF/tiles.xml

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE tiles-definitions PUBLIC
    "-//Apache Software Foundation//DTD Tiles Configuration 2.0//EN"
    "http://tiles.apache.org/dtds/tiles-config_2_0.dtd">
<tiles-definitions>

    <definition name="baseLayout" template="/s2-tiles-jsp/template/baseLayout.jsp">
        <put-attribute name="title" value="Template"/>
        <put-attribute name="header" value="/s2-tiles-jsp/default/header.jsp"/>
        <put-attribute name="menu" value="/s2-tiles-jsp/default/menu.jsp"/>
        <put-attribute name="body" value="/s2-tiles-jsp/default/body.jsp"/>
        <put-attribute name="footer" value="/s2-tiles-jsp/default/footer.jsp"/>
    </definition>

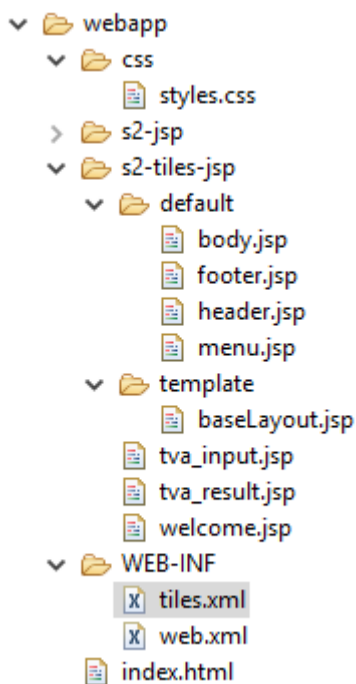
    <definition name="welcome_tile" extends="baseLayout">
        <put-attribute name="title" value="Welcome"/>
        <put-attribute name="body" value="/s2-tiles-jsp/welcome.jsp"/>
    </definition>

    <definition name="tva_input_tile" extends="baseLayout">
        <put-attribute name="title" value="tva_input"/>
        <put-attribute name="body" value="/s2-tiles-jsp/tva_input.jsp"/>
    </definition>

    <definition name="tva_result_tile" extends="baseLayout">
        <put-attribute name="title" value="tva_result"/>
        <put-attribute name="body" value="/s2-tiles-jsp/tva_result.jsp"/>
    </definition>

</tiles-definitions>

```



s2-tiles-jsp/template/baseLayout.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!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">
<%@ taglib uri="http://tiles.apache.org/tags-tiles" prefix="tiles" %>
    <title>
        <tiles:insertAttribute name="title" ignore="true" />
    </title>
</head>
<body>
    <table border="1" cellpadding="2" cellspacing="2" align="center">
        <tr>
            <td height="30" colspan="2">
                <tiles:insertAttribute name="header" />
            </td>
        </tr>
        <tr>
            <td height="250">
                <tiles:insertAttribute name="menu" />
            </td>
            <td width="350">
                <tiles:insertAttribute name="body" />
            </td>
        </tr>
        <tr>
            <td height="30" colspan="2">
                <tiles:insertAttribute name="footer" />
            </td>
        </tr>
    </table>
</body>
</html>

```

s2-tiles-jsp/default/menu.jsp

```

<%@ taglib prefix="s" uri="/struts-tags" %>
<div style="background-color: #f4e69c">My Side Menu<br/>
    <ul>
        <li>
            <s:url id="saisir_tva_url" action="saisir_tva"></s:url>
            <s:a href="%{saisir_tva_url}">tva</s:a>
        </li>
        <li>...</li>
    </ul>
</div>

```

s2-tiles-jsp/default/footer.jsp

```
<%@ taglib prefix = "s" uri = "/struts-tags" %>
<div style="background-color: #aef2ed">My Default Footer <br/>
  <s:url id="welcome_url" action="welcome"></s:url>
  <s:a href="%{welcome_url}">retour vers welcome</s:a><br/>
</div>
```

s2-tiles-jsp/welcome.jsp

```
<%@ taglib prefix = "s" uri = "/struts-tags" %>
<h2>Welcome in struts2 web app</h2>
```

Dans struts.xml

```
...
<package name="tp.web.actions" namespace="/s2" extends="struts-default">
  <result-types>
    <result-type name="tiles" class="org.apache.struts2.views.tiles.TilesResult" />
  </result-types>

  <default-action-ref name="welcome"/>

  <action name="welcome">
    <!-- <result>/s2-jsp/welcome.jsp</result>-->
    <result type="tiles">welcome_tile</result>
  </action>

  <action name="saisir_tva">
    <!-- <result>/s2-jsp/tva_input.jsp</result> -->
    <result type="tiles">tva_input_tile</result>
  </action>

  <action name="calculer_tva" class="tp.web.actions.CalculTvaAction"
    method="calculer">
    <!-- <result name="success">/s2-jsp/tva_result.jsp</result>
    <result name="input">/s2-jsp/tva_input.jsp</result> -->
    <result name="success" type="tiles">tva_result_tile</result>
    <result name="input" type="tiles">tva_input_tile</result>
  </action>
...
```

4. lien entre spring et struts2

Dans **pom.xml**

```
...
<dependency>
  <groupId>org.apache.struts</groupId>
  <artifactId>struts2-spring-plugin</artifactId>
  <version>2.3.8</version>
</dependency>
...
```

WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

  <display-name>struts2WebApp</display-name>
  <filter>
    <filter-name>struts2</filter-name>
    <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter
  </filter-class>
  </filter>
  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/s2/*</url-pattern>
  </filter-mapping>

  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:/springContext.xml</param-value>
  </context-param>
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener
  </listener-class>
  </listener>

</web-app>
```

src/main/resources/springContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd" >

    <!-- <import resource="dataSourceSpringConf.xml" /> -->
    <!-- <import resource="jpaSpringConf.xml" /> -->
    <context:annotation-config/>

    <context:component-scan base-package="tp.web.actions"/>
    <!-- <context:component-scan base-package="tp.service.spring"/> -->
    <context:component-scan base-package="tp.dao.spring"/>

</beans>
```

tp.dao.spring.UserDaoSpring

```
....
@Repository //ou @Component (spring) ou @Service sur un service métier
@Scope("singleton")
public class UserDaoSpring implements UserDao{
    ...
}
```

tp.web.actions.SpringUserCrudAction

```
package tp.web.actions;
...
@Component //spring --> bean d'action d'abord géré par Spring puis utilisé par struts2
//default id/name of this spring bean : springUserCrudAction
@Scope("prototype") //scope spring compatible avec logique / cycle de vie "struts2"
public class SpringUserCrudAction extends ActionSupport implements ModelDriven<User> {
    ...
    @Autowired
    protected UserDao userDao = null;
    ...
}
```

dans struts.xml

```
<!-- class="tp.web.actions.UserCrudAction" sans spring
ou bien class="springUserCrudAction" avec spring -->

    <action name="listUser" method="refreshList" class="springUserCrudAction">
        <result name="success"/>s2-jsp/user_crud.jsp</result>
    </action>
```

5. Intercepteur Struts2

Un intercepteur "struts2" est surtout utile pour forcer à naviguer vers une page de login avant de pouvoir déclencher certaines actions. D'autres types d'intercepteurs sont envisageables.

Exemple de classe d'intercepteur :

tp.web.interceptors **MyLoginInterceptor**

```
package tp.web.interceptors;
```

```
import java.util.Map;
```

```
import com.opensymphony.xwork2.ActionInvocation;
```

```
import com.opensymphony.xwork2.interceptor.AbstractInterceptor;
```

```
public class MyLoginInterceptor extends AbstractInterceptor {
```

```
    @Override
```

```
    public String intercept(ActionInvocation invocation) throws Exception {
```

```
        System.out.println("dans MyLoginInterceptor.intercept() ...");
```

```
        Map<String, Object> session = invocation.getInvocationContext().getSession();
```

```
        if(session.get("userId")==null)
```

```
        {
```

```
            //si pas de "userId" en session http renvoyer une chaine "result" pour
```

```
            //demander à rediriger vers une action définie dans struts.xml pour
```

```
            //forcer à passer préalablement par "saisir_inscription" ou "login"
```

```
            return "redirect_to_saisir_inscription"; //ou "redirect_login" ou ...;
```

```
        } else {
```

```
            return invocation.invoke();
```

```
        }
```

```
    }
```

```
    @Override
```

```
    public void destroy() {
```

```
        super.destroy();
```

```
        System.out.println("fin de MyLoginInterceptor");
```

```
    }
```

```
    @Override
```

```
    public void init() {
```

```
        super.init();
```

```
        System.out.println("initialisation de MyLoginInterceptor");
```

```
    }
```

```
}
```


Exemple d'utilisation dans struts.xml :

```
...
<package name="tp.web.actions" namespace="/s2" extends="struts-default">

    <interceptors>
        <interceptor name="myLoginInterceptor"
                     class="tp.web.interceptors.MyLoginInterceptor">
        </interceptor>
    </interceptors>

    ....

    <action name="ListeProduitsForCaddy" method="loadListeProduits"
           class="tp.web.actions.CaddyAction">
        <interceptor-ref name="defaultStack"/> <!-- intercepteurs par défaut -->
        <interceptor-ref name="myLoginInterceptor"/>
        <result name="success">/s2-jsp/add_in_caddy.jsp</result>
        <result name="redirect_to_saisir_inscription"
                type="redirect">saisir_inscription</result>
    </action>

    ...

    <action name="saisir_inscription" >
        <result>/s2-jsp/inscription.jsp</result>
    </action>

    ...
```

ANNEXES

VI - Annexe – Essentiel de Struts 1

1. Présentation de STRUTS 1

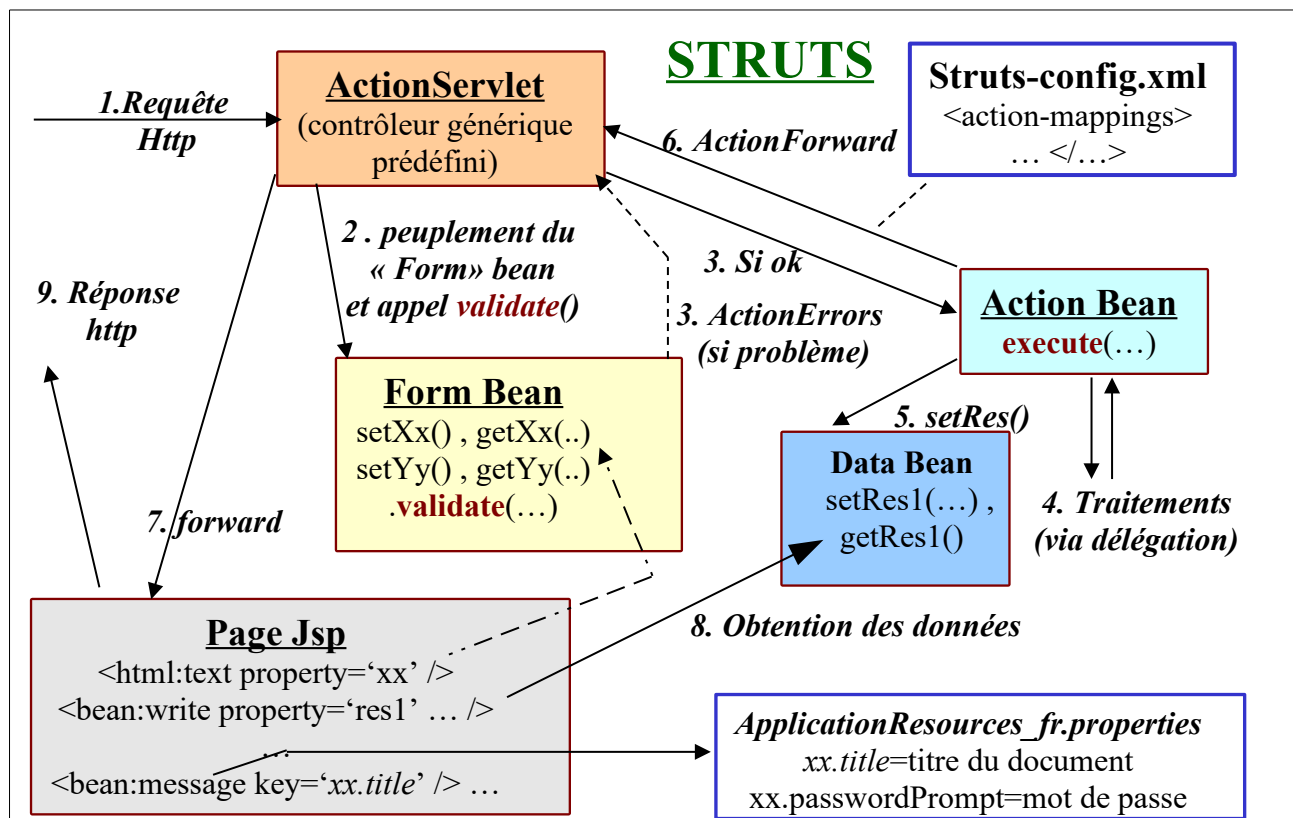
1.1. Valeur ajoutée de STRUTS :

- **Framework structurant** (allant dans le sens d'une bonne conception).
- **Modèle « MVC2 » partiellement « pré-programmé »**.
- **Très grande simplification de la gestion des formulaires en cas d'erreur(s) de saisie.**
- **Bonne bibliothèque de « Tag. » prédéfinis simplifiant la syntaxe des pages JSP.**
- **Support de l'internationalisation** (Les textes des messages sont stockés dans des fichiers de ressources).
- **Gestion simplifiée de l' « upload file »**.

1.2. Architecture de STRUTS :

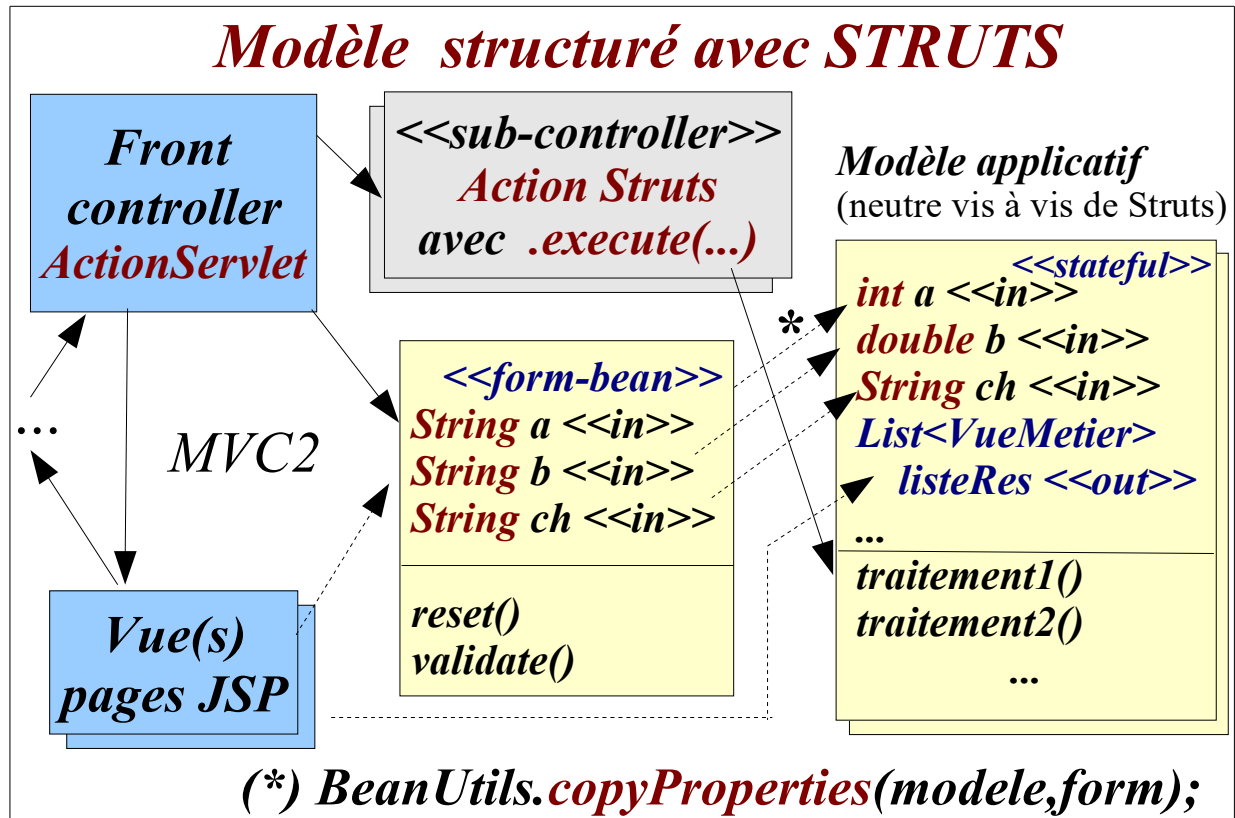
L'architecture de STRUTS repose sur les éléments suivants :

- Un **servlet prédéfini « ActionServlet »** jouant le rôle de « **Super Contrôleur** » générique.
- Des « **java bean** » de type « **ActionForm Bean** » (à programmer) *faisant office de « firewall » entre http et les actions internes* : un objet « **ActionForm Bean** » sert à gérer les paramètres http (stockage, relecture, vérification de saisies).
Nb : un objet « **ActionForm Bean** » est généralement utilisé pour **re-peupler un formulaire html** avec les **champs non erronés** (L'utilisateur n'a plus qu'à re-saisir les champs à problème).
- Des « **java bean** » de type **Action** (à programmer) et jouant le rôle de « **sous servlet** » : un objet « **action** » déclenche des traitements, peuple éventuellement un « **DataBean** » avec les résultats et **initialise un renvoi vers une vue** (page JSP) en utilisant les « **ActionMapping** » du fichier de configuration **struts_config.xml**.
- Des **pages JSP** dont la partie dynamique est en quasi totalité gérée via des balises prédéfinies (TagLib **struts-html** et **struts-bean**).



Dans le schéma précédent il vaut mieux distinguer :

- les **éléments spécifiques à STRUTS** et donc très peu portables : Page Jsp avec tags spécifiques, ActionFormBean & ActionBean avec héritages imposés + fonctions imposées validate() et perform() avec aspects "Struts & web" .
- les **éléments du modèle métier** : Bean de données ("vue des entités"), Bean de traitement non orienté web ("Proxy vers EJB , Traitement applicatif , Règle du métier ,).



1.3. Super Contrôleur ActionServlet et configuration associée

Le servlet **ActionServlet** constitue le cœur du framework « STRUTS ».

Voici un extrait du fichier de configuration **web.xml** :

```
...
<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>
  <init-param>
    <param-name>validate</param-name>
    <param-value>true</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>
...
<!-- Action Servlet Mapping -->
  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>

  <!-- The Welcome File List -->
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
...
```

1.4. ActionForm Bean et Validation des saisies

Un « Form Bean » se déclare dans le fichier **struts-config.xml** de la façon suivante :

```
<form-beans>
  <!-- Logon form bean -->
  <form-bean name="logonForm" type="org.apache.struts.webapp.example.LogonForm"/>
  ...
</form-beans>
```

Ainsi associé à un nom logique, ce bean peut être référencé au sein d'un « Action mapping » :

```
<action-mappings>
  ...
  <action path="/logon" type="org.apache.struts.webapp.example.LogonAction" name="logonForm"
scope="request" input="/logon.jsp"/>
  ...
</action-mappings>
```

En conséquence de tous ces paramétrages, une URL en `.../monAppWeb/logon.do` déclenchera le servlet **action** qui avant de déclencher l'action (de `path= "/logon"`) va :

- peupler le formBean de nom (`name="logonForm"`) en y appelant automatiquement les méthodes en `setXxx()`.
- invoquer la méthode **validate()** pour vérifier si les valeurs des paramètres Http ont bien été saisis.

Le code Java du **FormBean** est le suivant:

```
package org.apache.struts.webapp.example;
import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.*;

public final class LogonForm extends ActionForm {
    private String password = null;
    private String username = null;
    public String getPassword() { return (this.password); }
    public void setPassword(String password) { this.password = password; }
    public String getUsername() { return (this.username); }
    public void setUsername(String username) { this.username = username; }
    public void reset(ActionMapping mapping, HttpServletRequest request)
        { this.password = null; this.username = null; }

    /* Validate the properties that have been set from this HTTP request,
     * and return an <code>ActionErrors</code> object that encapsulates any
     * validation errors that have been found. If no errors are found, return
     * <code>null</code> or an <code>ActionErrors</code> object with no
     * recorded error messages.    */

    public ActionErrors validate(ActionMapping mapping,
                                HttpServletRequest request) {
        ActionErrors errors = new ActionErrors();
        if ((username == null) || (username.length() < 1))
            errors.add("username", new ActionMessage("error.username.required"));
        if ((password == null) || (password.length() < 1))
            errors.add("password", new ActionMessage("error.password.required"));
        return errors;
    }
}
```

Remarque importante: En cas d'erreur, la méthode **execute()** du bean d'action n'est même pas invoquée et le contrôleur de struts renvoie la page (`input="/logon.jsp"`) dont la balise `<html:errors/>` sera automatiquement remplacée par la collection d'erreurs retournées.

1.5. Action Bean

Via l'entrée suivante du fichier de configuration `struts-config.xml`

```

<action-mappings>
...
<action path="/logon" type="org.apache.struts.webapp.example.LogonAction" name="logonForm"
scope="request" input="/logon.jsp"/>
...
</action-mappings>

```

une URL en `.../monAppWeb/logon.do` déclenchera le servlet **action** qui (sauf erreur de validation au niveau du FormBean) , va déclencher la méthode **execute()** de l'objet **action** (de path= `"logon"` et dont la **classe java** correspond à la valeur de l'attribut **type**).

```

...
public final class LogonAction extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException
    { /* NB: le nouveau nom de perform() est maintenant execute() */

// Récupérer les paramètres d'entrées
String username = ((LogonForm) form).getUsername();
String password = ((LogonForm) form).getPassword();

/* Traitements divers et variés */

ActionErrors errors = new ActionErrors();
if (/* problème */)
    errors.add(ActionErrors.GLOBAL_ERROR,
        new ActionMessage("error.xxx.yyy"));

// Remonter les erreurs et effectuer une redirection vers le formulaire qui est à l'origine de
// la requête courante:
if (!errors.empty()) {
    saveErrors(request, errors);
    return (new ActionForward(mapping.getInput()));
}

else
{
    ...
    // Effectuer la redirection prévue en cas de succès
    return (mapping.findForward("success"));
}

}
}

```

Dans le cas présent , la redirection déclenchée par notre action fait référence à une configuration globale :

```

<global-forwards>
    <forward name="logoff" path="/logoff.do"/>
    <forward name="logon" path="/logon.jsp"/>
    <forward name="success" path="/mainMenu.jsp"/>

```

```
<global-forwards>
```

Dans d'autres cas, l'action est associée à ses propres noms logiques d'éléments de redirection (via **<action-mappings>**):

```
<action path="/editSubscription" type="org.apache.struts.webapp.example.EditSubscriptionAction"
name="subscriptionForm" scope="request" validate="true">
    <forward name="failure" path="/mainMenu.jsp"/>
    <forward name="success" path="/subscription.jsp"/>
</action>
```

1.6. Considérations sur le multi-threading

Un "bean" de type "Action" que l'on peut considérer comme un *"bean" de traitement* ou comme un *"sous servlet"* **est toujours instancié qu'une seule fois** par les mécanismes internes de STRUTS . La méthode **perform(...)** de la version 1.0.2 ou **execute(..)** de la version 1.1 peut donc être appelée plusieurs fois en même temps par différents threads.

Conséquences: des blocs en **synchronized(this) { ... }** s'imposent parfois s'il faut manipuler des attributs déclarés dans le haut de la classe d' action.

1.7. Librairie de Tags STRUTS prédéfinis pour pages JSP

Le fichier **web.xml** fait référence a ces librairies de Tag :

```
<!-- Struts Tag Library Descriptors -->
  <taglib>
    <taglib-uri>/tags/struts-bean</taglib-uri>
    <taglib-location>/WEB-INF/struts-bean.tld</taglib-location>
  </taglib>
  <taglib>
    <taglib-uri>/tags/struts-html</taglib-uri>
    <taglib-location>/WEB-INF/struts-html.tld</taglib-location>
  </taglib>
  <taglib>
    <taglib-uri>/tags/struts-logic</taglib-uri>
    <taglib-location>/WEB-INF/struts-logic.tld</taglib-location>
  </taglib>
```

Une page JSP commençant de la manière suivante pourra alors bénéficier d'un grand nombre de facilités.

```
<%@ page language="java" %>
<%@ taglib uri="/tags/struts-bean" prefix="bean" %>
<%@ taglib uri="/tags/struts-html" prefix="html" %>
....
```

1.7.a. internationalisation

```
<html:html locale="true">
<head>
<title><bean:message key="logon.title"/></title>
<html:base/>
</head>
<body bgcolor="white">...
</body>
</html:html>
```

Dans cet exemple le tag **<bean:message key="logon.title"/>** dicte aux mécanismes de STRUTS de récupérer le texte associé au point d'entrée "logon.title" du fichier de ressources lié à l'application web courante .

Ce fichier de ressources était référencé dans web.xml de la façon suivante (anciennes versions).

```
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
    <param-name>application</param-name>
    <param-value>.....ApplicationResources</param-value>
  </init-param>
... </servlet>
```

et est maintenant référencé dans **struts-config.xml** :

```
<message-resources parameter="java.resources.application" />
```

En fonction de la langue utilisée par l'utilisateur , le fichier pris en compte sera **ApplicationResources.properties** (pour la langue par défaut) ou bien **ApplicationResources_xx.properties** où **xx** est le code du pays (ex: **fr** pour France).

```
button.cancel=Cancel
button.confirm=Confirm
database.load=Cannot load database from {0}
error.password.required=<li>Password is required</li>
logon.title=MailReader Demonstration Application - Logon
...
prompt.password=Password:
```

Nb: les parties en {0} , {1} sont prévues pour être automatiquement remplacés par les paramètres de **FormatMessage** (*java.text*).

1.7.b. Accès simplifié au "FormBean" lié à la page courante

A la place d'écrire:

```
<input type="text" name="username"
value="<%= loginBean.getUsername() %>" />
```

on peut se permettre d'écrire seulement

```
<html:text property="username" />
```

Dans ce dernier le **java bean** automatiquement utilisé en interne pour récupérer les valeurs (via `getXxx()`) correspond au **"Form Bean"** (de *scope="session" ou "request"*) dont le nom (dans `struts-config.xml` [`<action path="/logon" ... name='logonForm' ...>`]) est **"logonForm"** si l'attribut **action** de la balise `<html:form>` englobante vaut **"/logon"**.

1.7.c. Affichage de valeurs dans une page jsp

La balise `<bean:write>` de Struts est une version améliorée de `<jsp:getProperty>` qui:

- reconnaît une syntaxe évoluée des noms de propriétés (ex: **property="subpartx.propy"** déclenchant `getSubpartx().getPropy()`).
- en version 1.1, sait effectuer un formatage des données (ex: **format="#.##"** pour n'afficher que 2 chiffres après la virgule)
- ...

```
<bean:write name="empruntForm" property="mensualite" format="#.##" />
```

2. Présentation rapide des fonctionnalités avancées

2.1. Struts Validator

Struts Validator est une **extension** (maintenant intégrée par défaut) qui **permet d'encoder au sein d'un fichier "validation.xml" des règles de validation** qui seront automatiquement appliquées à certains "Form Beans".

D'autre part , On peut éventuellement demander à ce que la **validation encodée en XML** soit **déclenchée coté client** (navigateur) via une interprétation d'instructions **JavaScript** .

Avantages :

- Paramétrage XML souvent plus rapide que code java
- Règles prédéfinies (à paramétrer) souvent suffisantes pour des validations de premier niveau

Limitations:

- Si demandées coté client en version javascript , les validations ne sont bien effectuées que si l'on utilise un navigateur récent (la validation coté serveur doit alors prendre le relais).
- Des validations "inter-champs" (de second niveau) doivent généralement être encodées en Java

2.2. Modules

Une **grande application WEB** ("*xxx.war*") peut éventuellement être **décomposée en plusieurs "sous-modules"** .

Chaque sous module restera **relativement indépendant des autres** car il disposera de son propre fichier central "**struts-config.xml**" .

2.3. Tiles

Découpage sophistiqué d'une page en diverses régions:

- Modèle de découpage ("entête" , "pied de page" , "contenu" ,)
- Définitions de base (avec attribution de sous pages "jsp" pour chaque région)
- Définitions spécifiques (héritant d'une définition de base) et redéfinissant des sous pages spécifiques (cible directe d'un forward) .

==> Les **Tiles** permettent d'obtenir rapidement une **mise en page homogène , flexible et évolutive** sur l'ensemble d'un site.

2.4. JSTL et STRUTS EL

JSTL (Jsp Standard Tag Library) est un ensemble standard de "Tag" pour les pages JSP.

Ce standard est historiquement apparu après STRUTS.

Dès les premières versions de STRUTS , des ensembles de "Tag" spécifiques à STRUTS ont été mis en place et sont aujourd'hui encore employés dans un très grand nombre d'applications Web.

D'autre part, beaucoup de programmeurs "STRUTS" sont habitués à utiliser ces balises spécifiques (pas véritablement "standard").

Pour des raisons d'homogénéité (lisibilité , maintenance , ...) , il est fortement conseillé de s'en tenir qu'aux seules balises "STRUTS" sur des anciens projets.

Sur de nouveaux projets , on peut éventuellement utiliser les nouvelles balises "JSTL" à la place des anciennes balises "spécifiques STRUTS" au sein des pages JSP.

Etant donné que les balises "JSTL" ne couvrent pas l'ensemble des fonctionnalités de celles de STRUTS, on peut être amené à mixer les deux (STRUTS + JSTL) au sein d'une même page JSP. Cette collaboration fonctionne bien mais pose quelques problèmes de lisibilités (syntaxe moyennement homogène).

3. Struts - configuration de base

3.1. Exemple de configuration d'un projet STRUTS sous eclipse

Créer un **nouveau projet JAVA** sous eclipse (ne pas séparer les sources des binaires au départ) (exemple de nom de projet : *tp_struts*)

Créer ensuite les répertoires ("folder") suivants:

- **WebContent**
- **ConfigTomcat5**
- **WebContent/WEB-INF**
- **WebContent/WEB-INF/src**
- **WebContent/WEB-INF/classes**

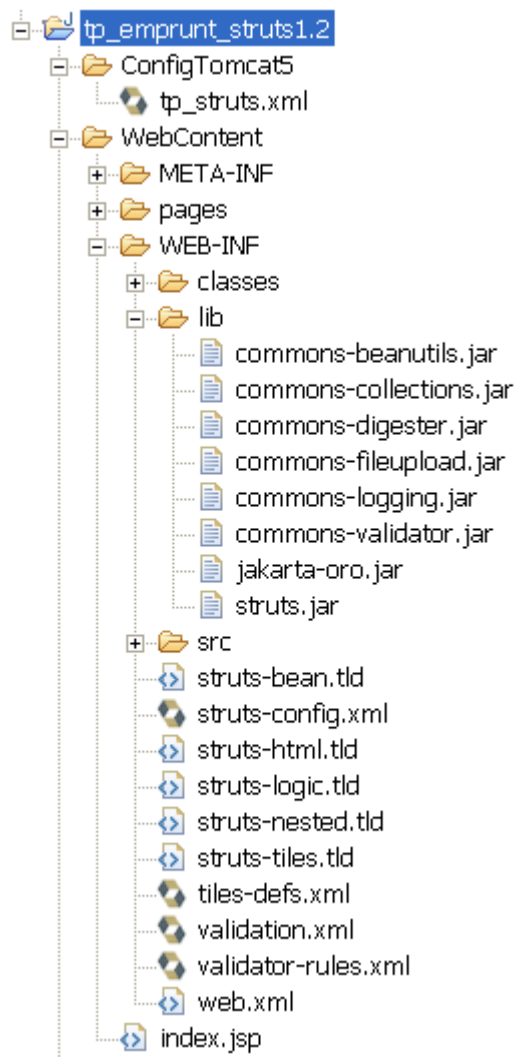
Sélectionner le projet et activer le menu contextuel "*properties/java build path/sources*"

Sélectionner alors "**WebContent/WEB-INF/src**" comme répertoire source et "**WebContent/WEB-INF/classes**" comme répertoire de sortie ("ouput") pour le résultat des compilations.

Extraire ensuite le contenu de l'archive "**struts-blank.war**" sous le répertoire "**WebContent**" du

projet (soit `c:\...\eclipse\workspace\tp_struts\WebContent`)

Effectuer un "**Refresh**" au sein de **eclipse** pour visualiser le résultat :



Petite erreur (*dans ce point de départ*) à corriger au sein du fichier **struts-config.xml**:

```
<!-- <message-resources parameter="MessageResources" /> -->
      <message-resources parameter="java.MessageResources" />
```

Activer ensuite le menu contextuel "**properties/java build path/libraries**" du projet et :

- via le bouton "**Add Jar ...**", intégrer au projet tous les fichiers ".jar" du répertoire WebContent/WEB-INF/lib du projet courant .
- via le bouton "**Add External Jar ...**", intégrer le fichiers "servlet-api.jar" du répertoire `.../Tomcat5\common\lib` .

Ecrire ou récupérer un petit fichier xml qui permettra à Tomcat d'utiliser directement le code de

l'application Web développé au niveau du projet eclipse:

ConfigTomcat5/tp_struts.xml

```
<Context className="org.apache.catalina.core.StandardContext"
  docBase="C:\eclipse\workspace\tp_emprunt_struts1.2\WebContent"
  path="/tp_struts" privileged="false"
  reloadable="true" >
</Context>
```

Recopier ensuite ce fichier xml au sein du répertoire **Tomcat5\conf\Catalina\localhost**

Démarrer ensuite tomcat via Tomcat5\bin\startup.bat puis effectuer un premier test via l'url suivante: http://localhost:8080/tp_struts

4. Positionnement STRUTS % frameworks concurrents

<i>Framework Java/WEB</i>	<i>caractéristiques</i>
STRUTS	Framework "assez ancien" qui a vraiment fait ses preuves . Assez simple à mettre en oeuvre, ce framework est assez populaire au sein de la communauté des développeurs Java. Très lié à l'api des servlets et au protocole HTTP , ce framework n'offre pas de vision abstraite. C'est du concret bien perceptible.
JSF (Java Server Faces)	Ce framework beaucoup plus récent (proposé par SUN et inspiré du framework .NET) offre une vision plus abstraite masquant les détails de HTTP. Logique événementielle .
Extension spring	Modèle objet basé sur de D.P. "IOC"
....	
....	

VII - Annexe – Bibliographie, Liens WEB + TP

1. Bibliographie et liens vers sites "internet"

http://struts.apache.org/	Site officiel struts 2 (apache)
https://www.roseindia.net/struts/struts2/index.shtml	
http://kmdkaci.developpez.com/tutoriels/java/bien-debuter-avec-struts2	Tutoriel (developpez.com)
...	Livre en anglais (Struts2 in action, ...)

2. TP

a) Effectuer via Struts 2 un calcul de mensualite constante pour rembourser un emprunt.

$\text{tauxMens} = (\text{tauxAnnuel} / 100) / 12 ;$

$\text{nbMois} = \text{nbAnnees} * 12 ;$

$\text{mens} = \text{MontantEmprunte} * \text{tauxMensuel} / (1 - \text{Math.pow}(1 + \text{tauxMensuel}, -\text{nbMois})) ;$

b) Développer de A à Z avec Struts 2 la partie "frontEnd" d'une application web de type "consultation de comptes bancaires" avec "login client", "virement interne" et "affichage listes des dernières opérations".