Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

M2 MIAGE FC / FA

MIAGE LILLE

Nouvelles Technologies et Systèmes d'Informations

Struts

Loïc Nogues

Octobre 2012

People matter, results count.

Présentation

- Struts sous son véritable nom, “**Apache Struts**“, est un framework open source utilisé pour faciliter le développement des applications web J2EE.

- Son but premier est de permettre la mise en place d’une **architecture MVC** (modéle-vue-controlleur) plus aisément. Il utilise pour cela l’API des servlets en les étendant et en donnant accès à des objets améliorant l’approche de ces dernières.

- Les points importants sont les suivants :
  - Structure de l’application web décrite dans un fichier *struts-config.xml*
  - Utilisation transparente des servlets via des classes adaptées
  - Exploitation des balises spécifiques dans les JSP (Java Server Page)

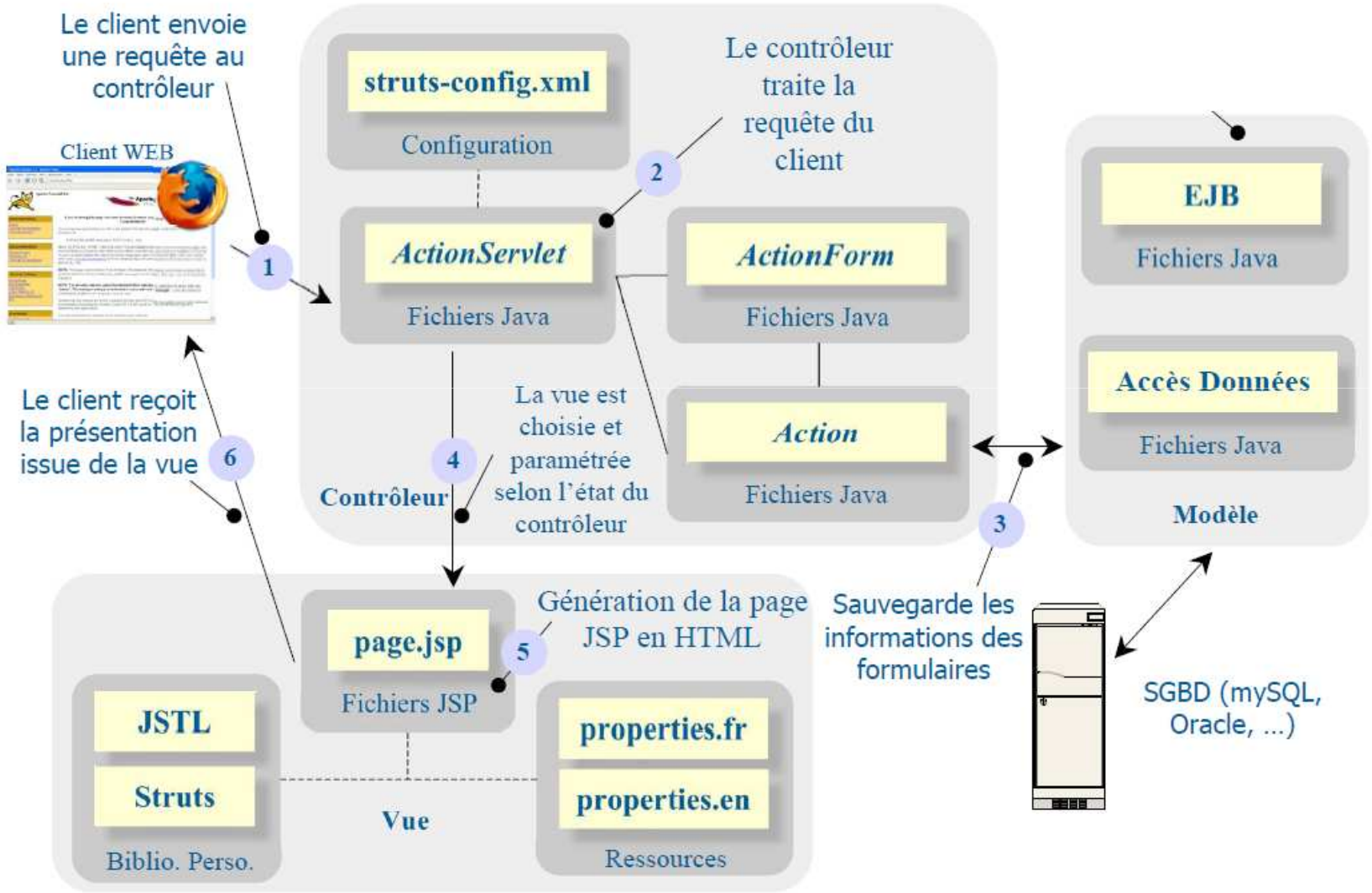
Présentation

- L'utilisation du framework Struts est assez lourd pour une application simple car il introduit un niveau de complexité non négligeable et l'effet de ses apports ne se ressent que lorsque l'application atteint une certaine taille.

- Les concurrents de Struts sont principalement Spring MVC et JSF (Java Server Face). La tendance pour le moment montre que ce framework se fait peu à peu remplacé par ces deux derniers frameworks plus puissant.

- Site officiel de Struts :
  - http://struts.apache.org

MVC et struts



Struts : le controleur

- Le contrôleur est le coeur de l'application web. Toutes les demandes du client transitent par lui

- Il est défini par un Servlet générique de type *ActionServlet* fournie par l'API de Struts

- Le contrôleur prend les informations dont il a besoin dans le fichier *struts-config.xml*

- La requête du client est traitée de deux manières
  - par une Action qui réalise des traitements, vérifications, ...
  - par un résultat affiché en retour au client (peut être dépendant de l'issue de l'action)

Intégration dans l'application web

- De manière à intégrer le framework Struts dans une application Web, il est nécessaire d'enrichir le fichier web.xml

- Par principe le contrôleur Struts est atteint par toutes les URL's se terminant par le suffixe « .do »

```
...
    <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>
    </servlet>
    <servlet-mapping>
        <servlet-name>action</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>
...
```

Le contrôleur est défini par la Servlet générique *ActionServlet*

En paramètre de la Servlet le fichier *struts-config.xml*

Toute URL terminant par « .do » est traitée par le contrôleur

Possibilité de définir plusieurs *ActionServlet* pour une même application Web

Le fichier configuration struts-config.xml

- Le fichier gérant la logique de l'application Web s'appelle par défaut struts-config.xml

- Il est placé dans le répertoire WEB-INF au même niveau que web.xml

- Il décrit essentiellement trois éléments
  - les objets Bean associés aux formulaires JSP (ActionForm)
  - les actions à réaliser suite aux résultats des objets ActionForm (Action)
  - les ressources éventuelles suites à des messages

- Le fichier de configuration est un fichier XML décrit par une DTD. La balise de départ est <struts-config>

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
          "http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
    ...
</struts-config>
```

Description de fonctionnement de l'architecture de l'application Web

Action

- Une action est un traitement obtenu suite au passage de la requête au contrôleur

- Nous distinguons deux sortes de requête client
  - requête sans paramètre issue par exemple d'une re-direction
  - requête avec paramètres issue par exemple d'un formulaire

- Les actions sont décrites dans la balise <action-mappings> au moyen de la balise <action>

- Selon le type de requête (avec ou sans paramètre) différents attributs de la balise <action> sont à renseigner

Action

- Dans le cas d'une requête sans paramètre le rôle du contrôleur est de relayer la demande du client à une URL ntrôleur

- La balise <action> dispose alors des attributs suivants
  - String path : définit le nom de l'URL (suffixe « .do » implicite)
  - String type : définit le nom de la classe Action qui doit traiter la demande. Utilisez la classe org.apache.struts.actions.ForwardAction dans ce cas précis de re-direction
  - String parameter : le nom de l'URL à qui doit être relayée la demande

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
          "http://struts.apache.org/dtds/struts-config_1_2.dtd">
<struts-config>
    <action-mappings>
        <action
            path="/monnom"
            parameter="/vues/mapage.jsp"
            type="org.apache.struts.actions.ForwardAction" />
    </action-mappings>
</struts-config>
```

Quand le client transmet l'URL « .../monnom.do » au contrôleur celui-ci redirige vers « /vues/mapage.jsp »

Il s'agit d'une re-direction

Action

- Appel du formulaire de saisie du nom et de l'age

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
          "http://struts.apache.org/dtds/struts-config_1_2.dtd">
<struts-config>
    <action-mappings>
        <action
            path="/formulaire"
            parameter="/vues/formulaire.jsp"
            type="org.apache.struts.actions.ForwardAction" />
    </action-mappings>
</struts-config>
```

Le formulaire est défini dans la page « formulaire.jsp »

Le client envoie la requête suivante

C'est la page *formulaire.jsp* qui est retournée au client

Action

- Dans le cas d'une requête avec paramètres le rôle du contrôleur est double
  - transmettre les informations dans un objet Bean de type ActionForm
  - réaliser une action spécifique (autre qu'une simple redirection)

- La balise <action> dispose, en plus des attributs déjà étudiés, des attributs suivants
  - String scope : les valeurs du formulaire sont stockées en session
  - String name : référence le nom d'une section <form-bean> (voir ci-après)
  - String validate : indique si la méthode validate de l'objet ActionForm doit être appelée ou non (voir ci-après)
  - String input : indique la vue qui sera appelée s'il y a erreur dans l'objet ActionForm

Action

- Les formulaires sont déclarés dans la balise <form-beans> au moyen de la balise <form-bean>

- La balise <form-bean> possède les attributs suivants
  - String name : nom du formulaire de la page JSP
  - String type : classe ActionForm qui stocke les paramètres du Bean

```
<?xml version="1.0" encoding="ISO-8859-1"?>
...
<struts-config>
    <form-beans>
        <form-bean name="monformulaire" type="monpackage.ClassActionForm" />
    </form-beans>
    <action-mappings>
        <action
            path="/monnom"
            name="monformulaire"
            scope="session"
            validate="true"
            input="/pageerreurs.do"
            parameter="/vues/mapage.jsp"
            type="org.apache.struts.actions.ForwardAction" />
    </action-mappings>
</struts-config>
```

Quand le client transmet l'URL « .../monnom.do » au contrôleur celui-ci redirige vers « /vues/mapage.jsp » si aucun problème

Dans le cas où les paramètres sont mauvais le contrôleur redirige vers «/pageerreurs.do »

Action

- Envoie d'une requête de type POST du formulaire

Trois actions et un formulaire sont actuellement définis

Le formulaire est défini par la valeur « formPersonne »

```
...
<struts-config>
    <form-beans>
        <form-bean name="formPersonne" type="monpackage.FormulaireBean" />
    </form-bean>

    <action-mappings>
        <action
            path="/main" name="formPersonne" scope="session" validate="true"
            input="/erreurs.do" parameter="/vues/main.jsp"
            type="org.apache.struts.actions.ForwardAction " />
        <action
            path="/formulaire"
            parameter="/vues/formulaire.jsp"
            type="org.apache.struts.actions.ForwardAction" />
        <action
            path="/erreurs"
            parameter="/vues/erreurs.jsp"
            type="org.apache.struts.actions.ForwardAction" />
    </action-map
</struts-config>
```

Les valeurs sont stockées dans *monpackage.FormulaireBean*

Si les données sont correctes direction « /vues/main.jsp » sinon direction « /erreurs.do »

Action

- Envoie d'une requête issue du formulaire

```
<%@ taglib uri="htmlstruts" prefix="html" %>
...
<body>
<center>
<h2>Personne - Formulaire</h2><hr>
<html:form action="/main" >
<table>
    <tr>
        <td>Nom</td>
        <td><html:text property="nom" size="20" /></td>
    </tr>
    <tr>
        <td>Age</td>
        <td><html:text property="age" size="3"/></td>
    </tr>
</table>
<table>
    <tr>
        <td><html:submit value="Envoyer" /></td>
        <td><html:reset value="Retablir" /></td>
        <td><html:button property="btnEffacer" value="Effacer" onclick="effacer()"/></td>
    </tr>
</table>
</html:form>
...
```

Bibliothèque de balises personnalisées Struts:HTML

L'action du formulaire est d'appeler la ressource « /main » associée

Les deux paramètres transmis en paramètre de la requête

ActionForm

- L'objectif d'un objet de type ActionForm est de stocker les informations issues d'un formulaire

- La classe Bean devra donc hériter de la classe ActionForm du package org.apache.struts.action

- C'est le contrôleur via la Servlet qui se charge de créer les instances des objets de type ActionForm

- Pour chaque propriété, le Bean doit définir un attribut et deux méthodes
  - un modifieur pour affecter une valeur à l'attribut
  - un accesseur pour obtenir la valeur de l'attribut correspondant te issue du formulaire

ActionForm

- Hormis le but de stocker les propriétés des formulaires, les objets de type ActionForm s'occupent aussi de l'aspect sémantique des données

- La méthode validate s'occupe de vérifier la validité des attributs de l'objet Bean

- ActionErrors validate(ActionMapping, HttpServletRequest)
  - le paramètre ActionMapping est un objet « image » de la configuration de l'action en cours stockée dans struts-config.xml
  - le paramètre HttpServletRequest est la requête du client transmise par la Servlet de contrôle
  - le retour ActionErrors permet de retourner des messages erreurs au client

ActionForm

- Un objet de type ActionMapping permet d'extraire les informations contenues dans le fichier struts-config.xml

- Il possède des méthodes associées
  - String getType() : pour accéder au contenu de l'attribut type
  - String getInput() : pour accéder au contenu de l'attribut input

```
...
<action
    path="/main" name="formPersonne" scope="session" validate="true"
    input="/erreurs.do" parameter="/vues/main.jsp"
    type="org.apache.struts.actions.ForwardAction " />
...
```

- Un objet ActionErrors permet d'ajouter des erreurs et l'ajout se fait par la méthode
  - add(String, ActionMessage) : où le premier paramètre correspond à la clé et le second au message d'erreur

ActionForm

- Stocker les informations du formulaire

```
public class FormulaireBean extends ActionForm {
    private String nom = null;
    private String age = null;

    public String getNom() {
        return nom;
    }

    public void setNom(String nom){
        this.nom = nom;
    }

    public void setAge(String age) {
        this.age = age;
    }

    public String getAge() {
        return age;
    }
    ...
}
```

La classe du framework Struts qui gère les Beans associés aux formulaires

Les deux attributs modélisant les propriétés du Bean

Les modifieurs et accesseurs pour traiter et modifier les propriétés

ActionForm

- Stocker et valider les informations du formulaire

```
public class FormulaireBean extends ActionForm {
    ... // Lié à la modélisation des propriétés
    public ActionErrors validate(ActionMapping arg0, HttpServletRequest arg1) {
        ActionErrors erreurs = new ActionErrors();
        if (nom == null || nom.trim().equals("")) {
            erreurs.add("nomvide", new ActionMessage("formulaire.nom.vide"));
        }
        if (age == null || age.trim().equals("")) {
            erreurs.add("agevide", new ActionMessage("formulaire.age.vide"));
        } else {
            try {
                int mon_age_int = Integer.parseInt(age);
                    if (mon_age_int < 0) {
                    erreurs.add("ageincorrect",
                    new  ActionMessage("formulaire.age.incorrect"));

                }
            } catch (Exception e) {
                    erreurs.add("ageincorrect",
                    new ActionMessage("formulaire.age.incorrect",age));
                }
        }
        return erreurs;
    }
```

Au début *erreurs* est vide donc pas d'erreur

Ajout des erreurs selon « l'arrivage »

Depuis la nouvelle version 1.2, il faut utiliser *ActionMessage* et non *ActionError* (désapprouvée)

ActionForm

- Les messages d'erreurs stockés dans un objet ActionErrors et retournés par la méthode validate sont transmis au contrôleur

- Si validate vaut « true » et que l'objet ActionErrors n'est pas null le contrôleur redirige vers la vue de l'attribut input

```
...
<action
    path="/main" name="formPersonne" scope="session" validate="true"
    input="/erreurs.do" parameter="/vues/main.jsp"
    type="org.apache.struts.actions.ForwardAction " />
...
```

- Les erreurs sont affichées dans la vue JSP au moyen de la balise personnalisée <errors> de la bibliothèque Struts-HTML

```
<%@ taglib uri="htmlstruts" prefix="html" %>
<html:errors/>
```

- La balise <errors> n'affiche pas les messages mais des identifiants présents dans un fichier ressource qui doit être référencé dans struts-config.xml

ActionForm

- Pour déclarer un fichier ressource dans le fichier configuration struts-config.xml utiliser la balise <message-resources>
  - String parameter : nom du fichier ressource
  - String key : à utiliser quand il y a plusieurs fichiers ressources

- Le fichier ressource doit porter comme extension .properties
  - Exemple de fichier : toto.properties

- Le fichier ressource doit être placer obligatoirement dans un sous-répertoire de /WEB-INF/classes. Exemples :
  - /WEB-INF/classes/toto.properties
  - /WEB-INF/classes/

- Pour choisir le fichier ressource, utilisez l'attribut bundle dans la balise <errors> en indiquant le nom de la clé

ActionForm

- Gérer les erreurs sémantiques du formulaire

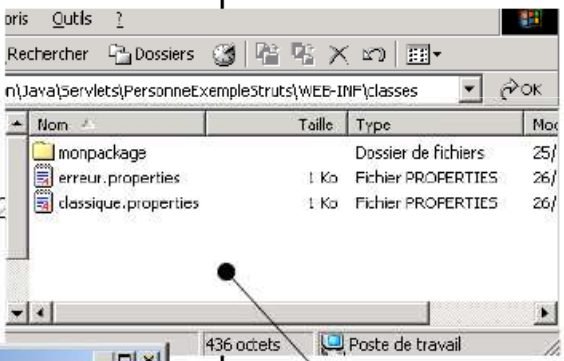
```
    ...
    <message-resources parameter="erreur" null="false" key="erreur" />
    <message-resources parameter="classique" null="false" key="classique" />
</struts-config>
```

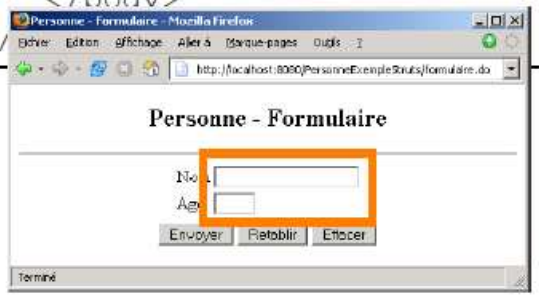
Fichier « erreur.properties »

```
formulaire.nom.vide=<li>Vous devez indiquer un nom</li>
formulaire.age.vide=<li>Vous devez indiquer un age</li>
formulaire.age.incorrect=<li>L'age [{0}] est incorrect</li>
errors.header=<ul>
errors.footer=</ul>
```

```
<%@ taglib uri="/WEB-INF/tlds/struts-

<html>
    <head>
        <title>Personne</title>
    </head>
    <body>
        <h2>Les erreurs suivantes se sont produites</h2
        <html:errors bundle="erreur" />
        <html:link page="/formulaire.do">
            Retour au formulaire
        </html:link>
    </body>
</
```





Emplacement des fichiers properties

Action

- Nous avons pour l'instant utilisé simplement la classe ForwardAction qui ne permet que de traiter des re-directions sans de réels traitements métiers

```
...
<action
    path="/main" name="formPersonne" scope="session" validate="true"
    input="/erreurs.do" parameter="/vues/main.jsp"
    type="org.apache.struts.actions.ForwardAction " />
...
```

- De manière à pouvoir réaliser des actions plus complexes (modification du modèle, création de nouveaux Bean, …) nous dérivons explicitement la classe Action

- Cette classe possède la méthode execute appelée par le constructeur de l'application Web si aucune erreur ne s'est produite

Action

- ActionForward execute(ActionMapping, ActionForm, HttpServletRequest, HttpServletResponse)
  - le paramètre ActionMapping est un objet image de la configuration de l'action en cours stockée dans struts-config.xml
  - le paramètre ActionForm correspond au Bean qui stocke l'information du formulaire
  - le paramètre HttpServletRequest est la référence de la requête
  - le paramètre HttpServletResponse est la référence de la réponse
  - le retour ActionForward est un objet pour identifier la destination prochaine que le contrôleur choisira

- Il faut modifier également struts-config.xml en ajoutant au corps de la balise <action> la balise <forward>
  - String name : étiquette pour la re-direction
  - String path : chemin de re-direction

Action

- Améliorer le traitement des actions du contrôleur

```
public class FormulaireAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
            HttpServletRequest req, HttpServletResponse res) throws Exception {
        FormulaireBean formulaire = (FormulaireBean)form;

        req.setAttribute("nom", formulaire.getNom());
        req.setAttribute("age", formulaire.getAge());

        return mapping.findForward("response");
    }
}
```

Grâce au paramètre *ActionForm* on a accès au contenu du Bean

L'objet requête de la Servlet est modifié en ajoutant deux attributs issus du Bean

On indique ici que la prochaine re-direction se fera dans « response »

```
...
<action
    path="/main" name="formPersonne" scope="session" validate="true"
    input="/erreurs.do" parameter="/vues/main.jsp"
    type="monpackage.FormulaireAction">
        <forward name="response" path="/reponse.do" />
</action>
...
```

Ajout dans le corps de cette action de la balise *<forward>*

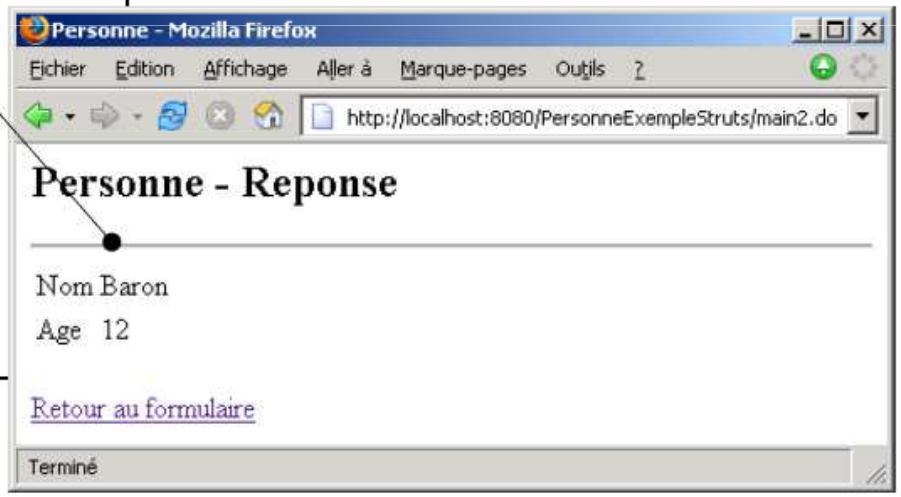
L'étiquette « response » indique une nouvelle page cible

Action

Réponse positive

```
<%@ taglib uri="htmlstruts" prefix="html" %>
<html>
    <head>
        <title>Personne</title>
    </head>
    <body>
        <h2>Personne - Reponse</h2><hr>
    <table>
        <tr>
            <td>Nom</td><td>${nom}</td>
        </tr>
        <tr>
            <td>Age</td><td>${age}</td>
        </tr>
    </table>
    <br>
    <html:link page="/formulaire.do">
        Retour au formulaire
    </html:link>
    </body>
</html>
```

Utilisation des EL dans la page JSP puisque deux attributs ont été définis dans la classe *FormulaireAction* (scope = request)



La balise *<link>* permet de retourner facilement un lien hypertexte

Taglib

http://struts.apache.org/1.x/struts-taglib/index.html

- Bean
  <%@taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
- Html
  <%@taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
- Logic
  <%@taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
- Nested
  <%@taglib uri="http://struts.apache.org/tags-nested" prefix="nested"%>

Questions

?