Développer une application Struts à l'aide de NetBeans

par GeertJan Wielenga (Son blog) Vincent Brabant (Traducteur) (Java, NetBeans, et Co)

Date de publication : 28/11/2005

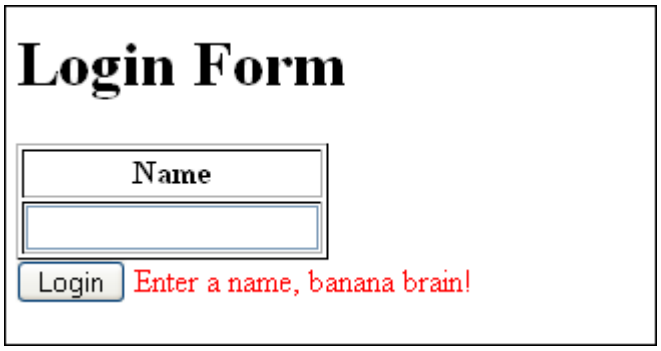
Dernière mise à jour : 05/12/2005

Ce tutoriel va vous faire découvrir l'ABC de Struts, au travers de l'EDI NetBeans 5.0 qui nous fournit un support Struts "out of the box".
Un grand merci à **GeertJan** pour avoir donné son accord pour traduire cet article qui fut publié sur son blog. (cfr Remerciements).

1 - Première partie

L'EDI NetBeans 5.0 simplifie Struts. En ne cochant qu'une seule case dans l'assistant New Web Application, toutes les bibliothèques Struts sont rajoutées à votre projet. De plus, vous aurez également le fichier magique struts-config.xml, où vous enregistrez toutes les fonctionnalités de Struts sur lesquelles vous travaillez. Des assistants sont également fournis, pour la création de deux catégories de classes Struts -- actionform beans et les actions. Des assistants sont également accessibles depuis le fichier struts-config.xml, pour la génération du code utilisé pour enregister les fonctionnalités de Struts.

Voici une procédure toute simple permettant la création d'une application Struts dans l'EDI NetBeans 5.0. Vous utiliserez le framework Struts pour créer une très simple page d'identification:



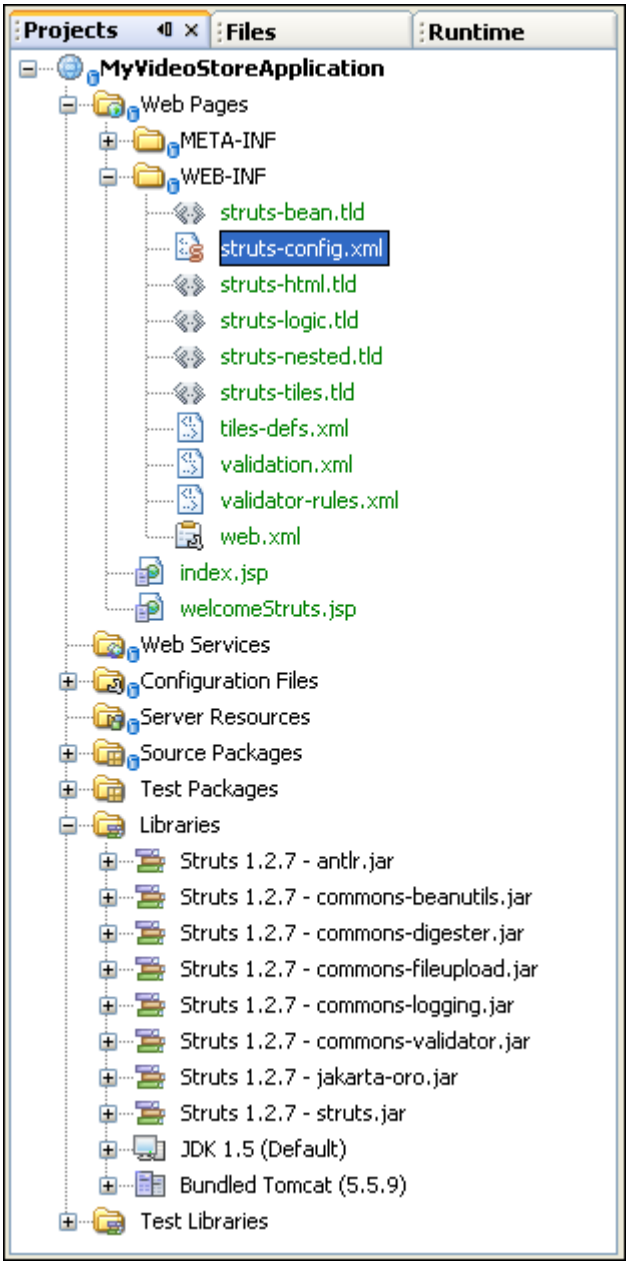
écran d'identification

Tout le traitement sera fait par Struts (Vous découvrirez ce en quoi consiste le traitement dans les étapes décrites ci-dessous). Vous n'allez pas apprendre beaucoup conceptuellement, mais à travers les étapes qui suivent ci-dessous, vous allez vous familiariser avec le framework Struts et la façon dont le support pour le framework a été implémenté dans l'EDI NetBeans 5.0.

1.1 - Configuration du projet

Cliquez sur Maj-Ctrl-N. Dans la catégorie Web, choisissez Web Application, et cliquez sur Next. Appelez-la MyVideoStoreApplication. Cliquez sur Next. Sélectionnez Struts 1.2.7 et cliquez sur Add Struts TLD. Cliquez sur Finish.

Voici ce que vous devriez voir, avec le fichier le plus important (le fichier struts-config.xml) sélectionné:



Structure du Projet

1.2 - Création des Pages JSP

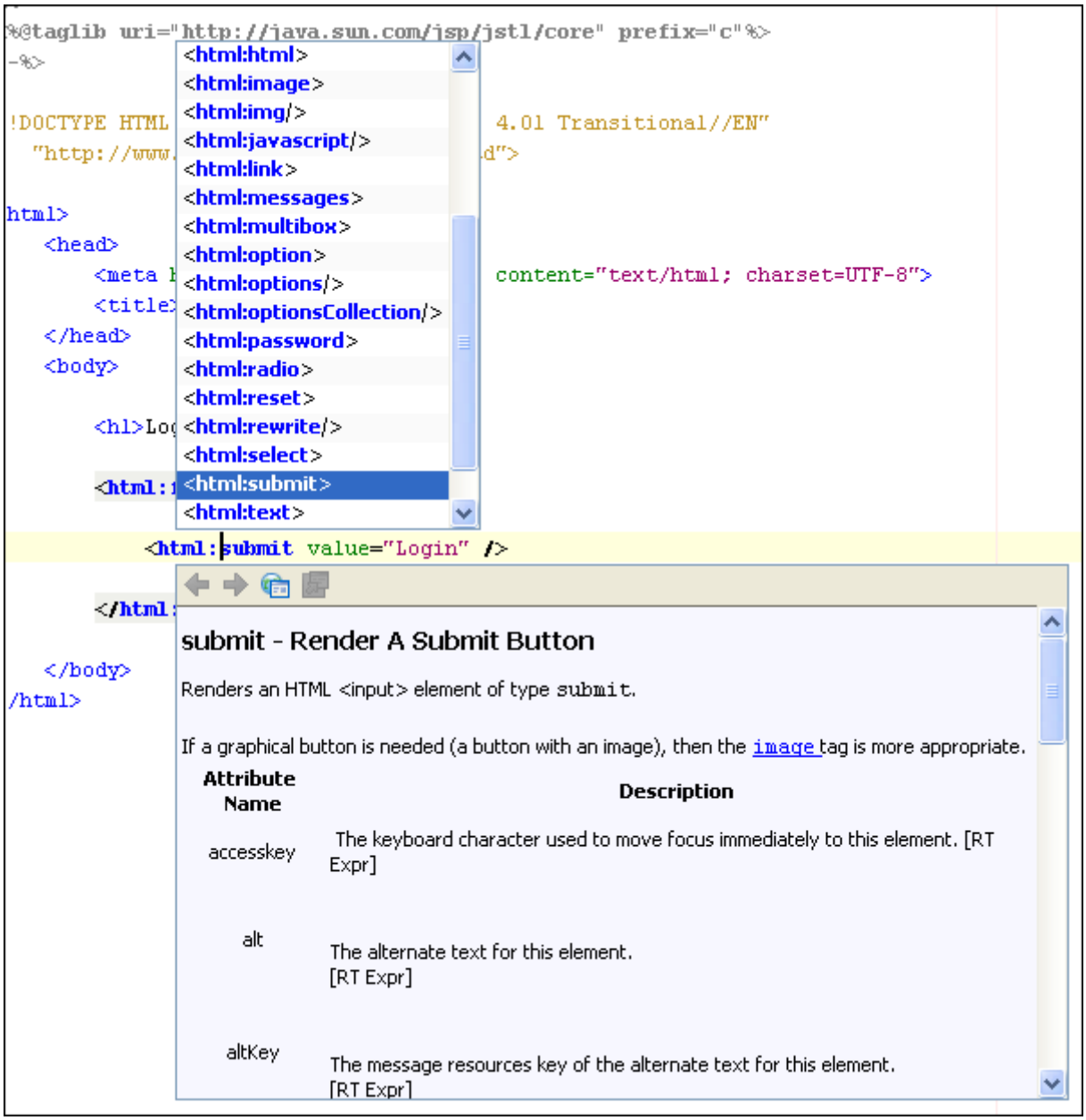
Cliquez-droit sur le noeud du projet MyVideoStoreApplication, choisissez New > JSP et appelez la loginForm. Cliquez sur Finish. Créez une autre page JSP et appelez-la loginSuccessful. Dans l'Éditeur de Source, remplacez le contenu des tags H1 dans les deux nouveaux fichiers JSP par quelque chose de plus parlant. Par exemple, modifier le texte JSP Page en Login Form et Login Successful! respectivement. Faites de même pour le texte dans les tags TITLE.

Copiez maintenant les deux premières directives du fichier welcomeStruts.jsp en haut de votre nouvelle page loginForm.jsp. Dans loginForm.jsp, en dessous des tags H1, ajoutez ce qui suit:

```
<html:form action="login">
```

```
    <html:submit value="Login" />
</html:form>
```

Il est facile de deviner que tout ce qui se trouve entre les tags html:form est géré par Struts. Notez que tout en tapant, l'EDI vous aide en suggérant différentes façon de completer le code que vous tapez. Vous avez également accès à la Javadoc Struts.



Code Completion & Javadoc des tags Struts

Dans la Palette de Composant, à droite de l'Éditeur de Source, étendez la section HTML et déposer l'élément Table juste au-dessus de la ligne

```
<html:submit value="Login" />
```

La boite de dialogue Insert Table apparait. Assurez-vous que les valeurs de Rows et Columns soient bien sur 1, pour qu'il crée un tableau d'une ligne et une colonne. Cliquez sur OK. Entre les tags TH, introduisez ce qui suit:

```
<bean:message key="login.name" />
```

Entres les tags TD, introduisez ce qui suit:

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

Ajouter login.name comme cléf au fichier ApplicationResource.properties et ajouter un message explication. Par exemple:

```
login.name=Name
```

Le corps de LoginForm.jsp contient maintenant:

```
<body>

    <h1>Login Form</h1>

    <html:form action="login">
        <table border="1">
            <thead>
                <tr>
                    <th><bean:message key="login.name" /></th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td><html:text property="name" /></td>
                </tr>
            </tbody>
        </table>
        <html:submit value="Login" />
    </html:form>

</body>
```

Corps de LoginForm.jsp

1.3 - Création du Bean Struts ActionForm

Cliquez-droit sur le noeud du projet MyVideostoreApplication et choisissez New > File/Folder. Dans la catégorie Web, choisissez le Bean Struts ActionForm et cliquez sur Next. Sélectionnez com.myapp.struts dans la liste déroulante des package et pressez Finish. Le bean actionform s'ouvre dans l'Éditeur de Source. Ouvrez struts-config.xml dans l'Éditeur de Source et notez qu'il contient, parmis d'autres choses, ce qui suit

```
<form-beans>
    <form-bean name="NewStrutsActionForm" type="com.myapp.struts.NewStrutsActionForm"/>
```

```
</form-beans>
```

Naviguez maintenant jusque dans le bean actionform dans l'Éditeur de Source et jetez un oeil à la méthode validate. Notez qu'un champ nommé name est validé par défaut. Si la validation échoue, ce qui se produit lorsqu'aucun nom n'est introduit dans la page JPS, un message qui est identifié par error.name.required est retournée. Ajoutez error.name.required comme clefs dans le fichier ApplicationResource.properties, et ajouter un message significatif. Par exemple:

```
error.name.required=Enter a name, banana brain!
```

Ajoutez maintenant ce qui suit dans le fichier loginForm.jsp, juste au dessus du tag de cloture </html:form>:

```
<html:errors />
```

1.4 - Création de l'Action Struts

Cliquez-droit sur le noeud du projet MyVideoStoreApplication et choisissez New > File/Folder. Dans la catégorie Web, choisissez l'Action Struts et cliquez sur Next. Sélectionnez com.myapp.struts dans la liste déroulante des packages. Introduisez login dans l'Action Path (le contenu de l'Action Path est donc maintenant /login).Cliquez sur Next. Remarquez que l'EDI suggère que vous associez l'action avec le bean actionform créé dans l'étape précédente. Dans Input Resource, sélectionnez votre page loginForm.jsp. Cliquez sur Finish.

Notez que maintenant, struts-config.xml contient, parmis d'autres choses, ce qui suit:

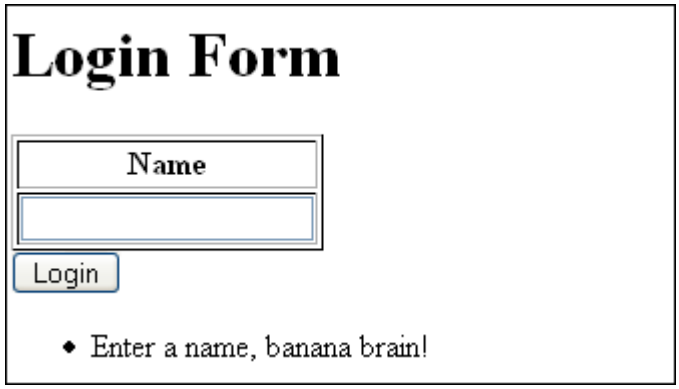
```
<action-mappings>
   <action input="/loginForm.jsp"
            name="NewStrutsActionForm"
            path="/login"
           scope="session"
            type="com.myapp.struts.NewStrutsAction"/>
   <action path="/Welcome"
        forward="/welcomeStruts.jsp"/>
</action-mappings>
```

Naviguez jusqu'à l'action et jetez un oeil à la méthode execute. Notez que cette action redirige vers la vue résultante appelée success. Vous avez besoin de définir la page loginsuccessfil.jsp comme étant la vue résultante. Ouvrez struts-config.xml dans l'Éditeur de Source, cliquez-droit n'importe où et choisissez Struts > Add forward. Introduisez success dans Forward Name. Naviguez jusqu'à la page loginSuccessful.jsp dans Resource File. Cliquez sur Action. Choisissez /login depuis la liste déroulante. Cliquez sur Add. Notez que le fichier struts-config.xml devrait maintenant être semblable à ceci:

```
<action-mappings>
  <action input="/loginForm.jsp"
           name="NewStrutsActionForm"
           path="/login"
          scope="session"
           type="com.myapp.struts.NewStrutsAction">
      <forward name="success"
               path="/loginSuccessful.jsp"/>
  </action>
  <action path="/Welcome"
       forward="/welcomeStruts.jsp"/>
</action-mappings>
```

1.5 - Exécution du Projet

Cliquez-droit sur le noeud du projet MyVideoStoreApplication et choisissez Run Project. Le projet déploye et affiche la page loginForm.jsp. Vous devriez voir quelque chose de semblable à ceci:

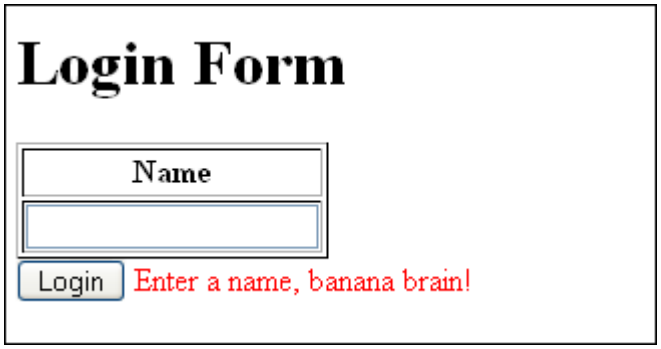


LoginForm

Lorsque vous introduisez un nom (ou n'importe quoi d'autre, pour autant que ce soit quelque chose), loginSuccessful.jsp est affiché. Une chose intéressante à faire à ce moment est de revenir dans le fichier ApplicationResource.properties. Modifier les quatres premières clefs par ceci:

```
errors.header=
errors.prefix=<span style="color: red">
errors.suffix=
errors.footer=
```

Déployez à nouveau l'application. Si la page loginSuccessful.jsp s'affiche, et que vous ne le désirez pas, modifier l'attribut scope du chemin /login dans struts-config.xml en request au lieu de session. Déployez à nouveau et notez que le message d'erreur a été reformaté.



LoginForm avec message d'erreur

Dans un futur tutoriel, je vais étendre ce scénario pour qu'il fournisse plus de fonctionnalités et qu'il devienne une page de login plus utile.

2 - Deuxième partie

2.1 - Introduction

Partant de la **première partie** de ce tutoriel, j'ai rajouté un champ password à la class Bean ActionForm et je l'ai rajoutée au tableau de la page loginForm.jsp. Ensuite, j'ai utilisé l'EDI pour améliorer l'application avec tout un tas de fonctionnalités Struts. Les sections qui suivent montrent comment améliorer l'application en rajoutant les choses suivantes:

2.1.1 - Fonctionnalité de validation

2.1.1.1 - Syntax-Level

J'ai modifié la classe Bean ActionForm pour vérifier si les champs name et password de la page loginForm.jsp sont remplis S'ils sont vides, on produit des messages d'erreur que la classe Bean ActionFrom obtient depuis le fichier ApplicationResource.properties.

2.1.1.2 - Business-Level

Ajout d'une classe SecurityManager et utilisation de celle-ci dans la classe Action pour vérifier si le nom et le mot de passe entré dans loginForm.jsp sont corrects. Si l'un est incorrect, produit un message d'erreur que la classe Action obtient depuis le fichier ApplicationResource.properties.

2.1.2 - Fonctionnalité d'Annulation

Ajout d'un bouton d'annulation à loginForm.jsp. Ajout de la méthode org.apache.struts.action.Action.isCancelled pour que, lorsque le bouton Cancel est enfoncé, une nouvelle page JSP est appelée via struts-config.xml.

2.1.3 - Fonctionnalité de Déconnection

Ajout d'un lien 'Déconnection' à loginSuccessful.jsp que, lorsqu'on clique dessus, appelle une nouvelle page JSP via struts-config.xml.

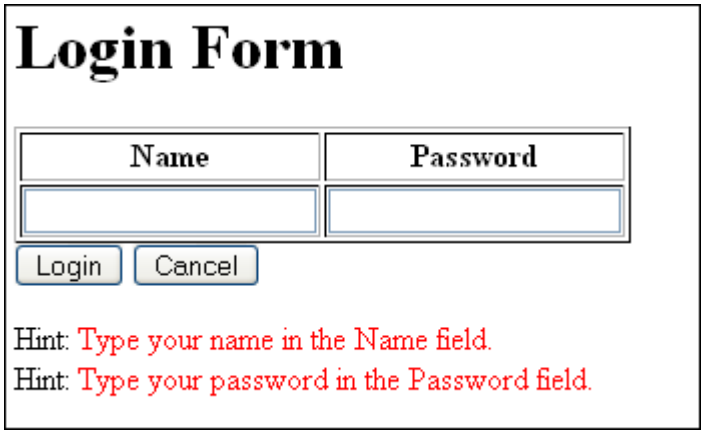
2.1.4 - Fonctionnalité de Gestion d'erreur

Réécriture du code pour le bouton d'Annulation pour que, lorsqu'on clique dessus, une erreur se produit et une nouvelle page JSP soit appelée via struts-config.xml. La nouvelle page JSP affiche un message d'erreur qu'il obtient du fichier ApplicationResource.properties.

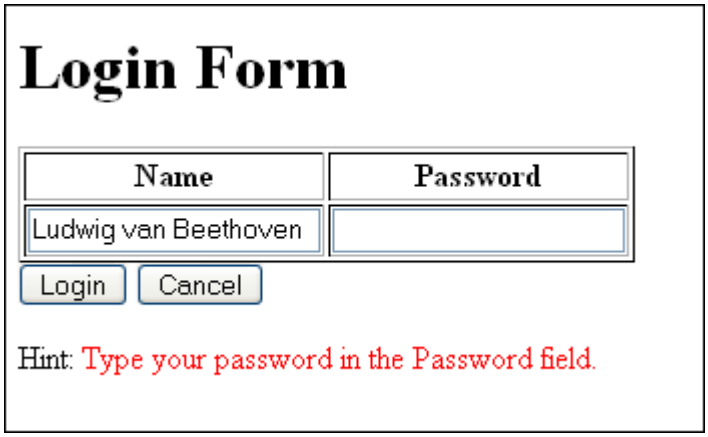
2.2 - Ce à quoi ont veut arriver

Visuellement, c'est ce à quoi l'application ressemble après que les fonctionnalités reprises ci-dessus ont été rajoutées. La première chose que vous voyez est loginForm.jsp, avec les messages d'erreur généré par la classe Bean ActionForm. En fait, je préfèrais que la méthode validate de la classe Bean ActionForm soit appelée seulement après que j'ai cliqué sur le bouton de Login. Aussi, si quelqu'un sait comment configurer cela, qu'il me le fasse savoir. J'ai

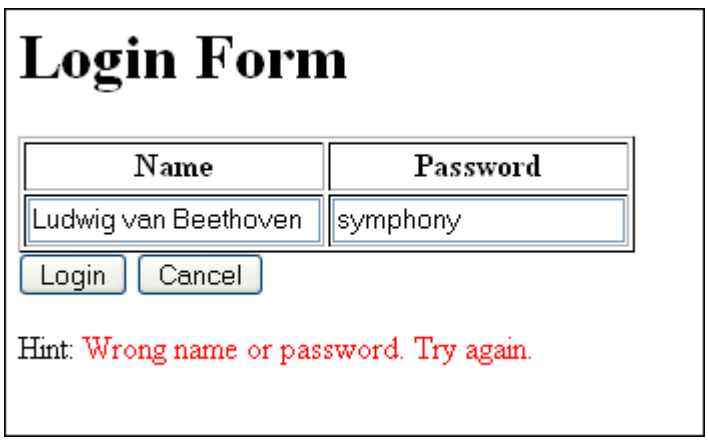
contourné le problème en préfixant 'Hint' avant le message d'erreur, pour que l'utilisateur ne finisse pas par voir le mot "erreur" avant même d'avoir fait quelque chose dans l'application. Voici donc à quoi ressemble la première page:



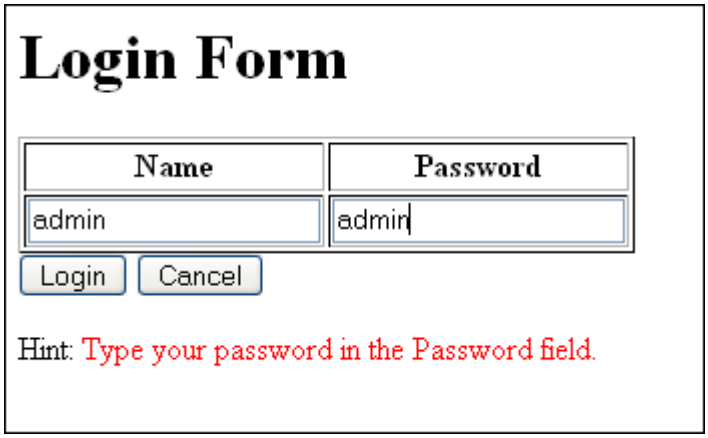
Du fait que les deux champs sont vides, la classe Bean ActionForm reçoit deux messages d'erreur via le fichier ApplicationResource.properties. Ensuite, si vous n'introduisez qu'un des deux champs (comme illustré ci-dessous), vous aurez toujours un message d'erreur pour le champs vide:



Ensuite, même si les deux champs sont remplis, vous aurez un nouveau message d'erreur si le nom et le mot de passe ne sont pas remplis correctement (comme illustré ci-dessous):

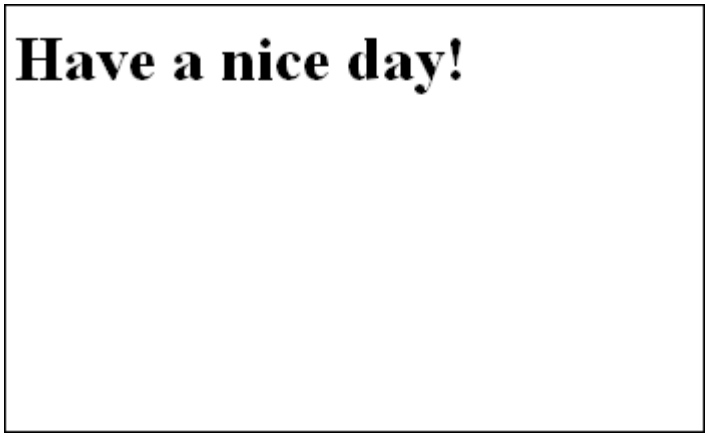


Ce n'est que lorsque le nom et le mot de passe voudront admin/admin (remarquez que le mot de passe est encrypté), comme illustré ci-dessous ...

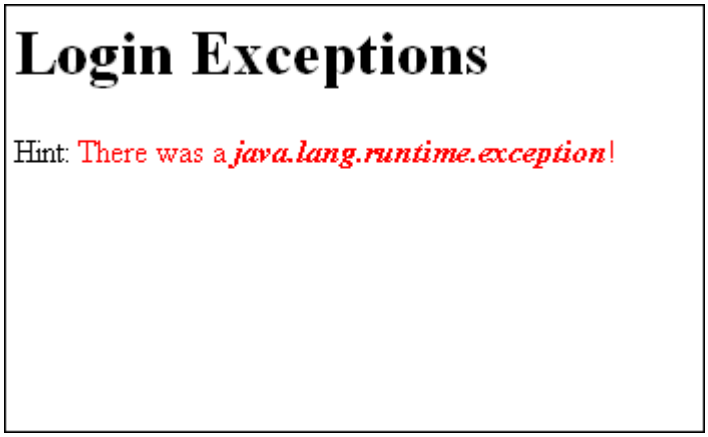


... que vous arriverez sur la page loginSuccessful.jsp:

Ensuite, lorsque vous cliquez sur le lien Logout, loginOut.jsp est affiché:



Finallement, nous modifions le bouton Annuler pour qu'une java.lang.RuntimeException soit jetée lorsque le bouton Annuler soit cliqué. On réimplémente la méthode isCanceled dans la classe Action pour que cela provoque l'appel d'une nouvelle page JSP qui affiche un message d'erreur obtenu depuis le fichier ApplicationResource.properties:



2.3 - Tutoriel

Pour implémenter toutes les fonctionnalités reprises ci-dessus, suivez les étapes décrites ci-dessous:

2.3.1 - Ajout de la Fonctionnalité de Validation au niveau de la syntaxe

Dans Struts, le Bean ActionForm agit comme un lien entre la page JSP et une action Struts. Il capture les entrées de l'utilisateur sur les pages JSP et les passe à l'action. Un Bean ActionForm peut également valider l'entrée avant de le passer à l'action. Aussi, nous voyons ici que le Bean ActionForm vérifie si les champs sont remplis ou pas (notez que le premier champ, name, est créé et validé par défaut lorsque vous utilisez l'assistant New ActionForm Bean dans l'EDI):

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

Le fichier ApplicationResource.properties (que l'EDI a créé lorsque vous avez rajouté Struts au projet) contient tous les messages qui doivent être affichés. Deux messages sont créés ci-dessus. Leur texte sont trouvés par Struts dans le fichier ApplicationResource.properties, et affichés dans LoginForm.jsp. Ainsi donc, ajoutez ce qui suit au fichier ApplicationResource.properties:

```
error.name.required=Type your name in the Name field.
error.password.required=Type your password in the Password field.
```

Et voici comment je les affiche dans loginForm.jsp:

```
<html:errors property="nameEmpty" />
<html:errors property="passwordEmpty" />
```

Pour utilisez les tags <html:errors> ci-dessus, vous devez avoir cette directive taglib au début de votre loginForm.jsp:

```
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html" %>
```

2.3.2 - Ajout de la Fonctionnalité de Validation au niveau Business

Pour le première fois, j'ai vraiment compris la différence entre la validation au niveau syntaxe, et la validation au niveau business. Posez-vous cette question: Est-ce que le business pour lequel cette application est créée se soucie du fait que les champs du formulaire d'identification sont vides ou pas ? Je doute que oui ! Mais est-ce qu'ils vont se soucier de savoir si l'utilisateur est capable d'accéder à l'application sans y être autorisée ? Oui, évidemment qu'ils vont s'en soucier. Et bien Struts fait la distinction entre une validation au niveau de la syntaxe et une validation au niveau business: La première est faite par la classe Bean ActionForm, tandis que la seconde est faite par la classe Action. Du fait que la classe Bean ActionForm agit comme une passerelle entre la page JSP et l'action, vous désiserez que la validation de la syntaxe se fasse dans la classe Bean ActionForm, parce que cette classe négocie directement avec la page JSP, alors que l'action non. L'action est plus concernée par les problèmes au niveau Business.

J'ai donc ici une classe très simple, appelée SecurityManager:

```
package com.myapp.struts;

public class SecurityManager {

    /** Creates a new instance of SecurityManager */
    public SecurityManager() {
    }

    public static boolean AuthenticateUser(String name, String password) {
        // Business level authentication - simple check of user name/password = admin/admin
        // TODO: Implement authentication (e.g. use database, etc.)
        return name.equals("admin") && password.equals("admin");
    }
}
```

Et dans la méthode execute de la classe Action, j'ai ceci:

```
if
 (SecurityManager.AuthenticateUser(newStrutsActionForm.getName(),newStrutsActionForm.getPassword())){
   return mapping.findForward(SUCCESS);
} else {
   return mapping.getInputForward();
}
```

Notez que le nom et le mot de passe sont récupérés de la classe Bean ActionForm et authentifiée à l'aide de la classe SecurityManager. Pour être capable d'utiliser la classe ActionForm, vous devez la caster vers le type correcte. Faites-ceci au début de la méthode execute de la classe Action, comme ceci:

```
NewStrutsActionForm newStrutsActionForm = (NewStrutsActionForm)form;
```

Aussi, si le nom et le mot de passe récupéré de la classe Bean ActionForm ne sont pas corrects, la page loginForm.jsp est à nouveau affichée (c'est ce que fait getInputForward()). Cependant, l'utilisateur ne sait pas ce qui ne va pas, parce que vous n'avez pas afficher un message d'erreur approprié.

Ajouter ce qui suit juste au dessus de la ligne return mapping.getInputForward():

```
ActionMessages errors = new ActionMessages();
ActionMessage error = new ActionMessage("errors.login.invalid");
errors.add("loginWrong",error);
saveErrors(request.getSession(),errors);
```

Vous voyez ici qu'un nouveau message d'erreur est créé. Il vous faut également aller dans le fichier ApplicationResource.properties et définir le texte à afficher pour l'erreur errors.login.invalid :

```
errors.login.invalid=Wrong name or password. Try again.
```

Et l'afficher ensuite dans la page loginForm.jsp, juste comme les messages d'erreurs précédents:

```
<html:errors property="loginWrong" />
```

2.3.3 - Ajouter la Fonctionnalité d'Annulation

Ajouter un bouton d'annulation est quelque chose de très simple dans Struts. Ici, vous voyez le mien, juste en dessous du bouton de soumission (il est mis en évidence dans la copie d'écran):

```
<body>

    <h1>Login Form</h1>

    <html:form action="login">

        <table border="1">
            <thead>
                <tr>
                    <th><bean:message key="login.name" /></th>
                    <th><bean:message key="login.password" /></th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td><html:text property="name" /></td>
                    <td><html:text property="password" /></td>
                </tr>
            </tbody>
        </table>
        <html:submit value="Login" />
        <html:cancel />
        <p>

        <html:errors property="nameEmpty" />
        <html:errors property="passwordEmpty" />
        <html:errors property="loginWrong" />

    </html:form>

</body>
```

Mais comment spécifier ce qui doit se passer lorsque qu'on clique sur le bouton ? Ajouter ceci à la méthode execute de la classe Action:

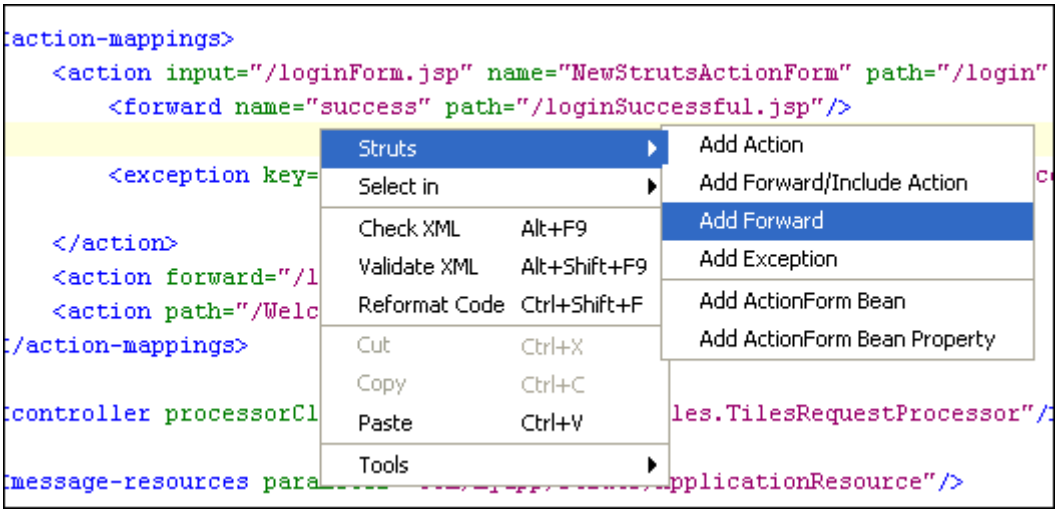
```
if (isCancelled(request)){
  return mapping.findForward(CANCEL);
}
```

Heureusement, la Javadoc de Struts, disponible dans l'EDI vous dit ce que fait le code ci-dessus:

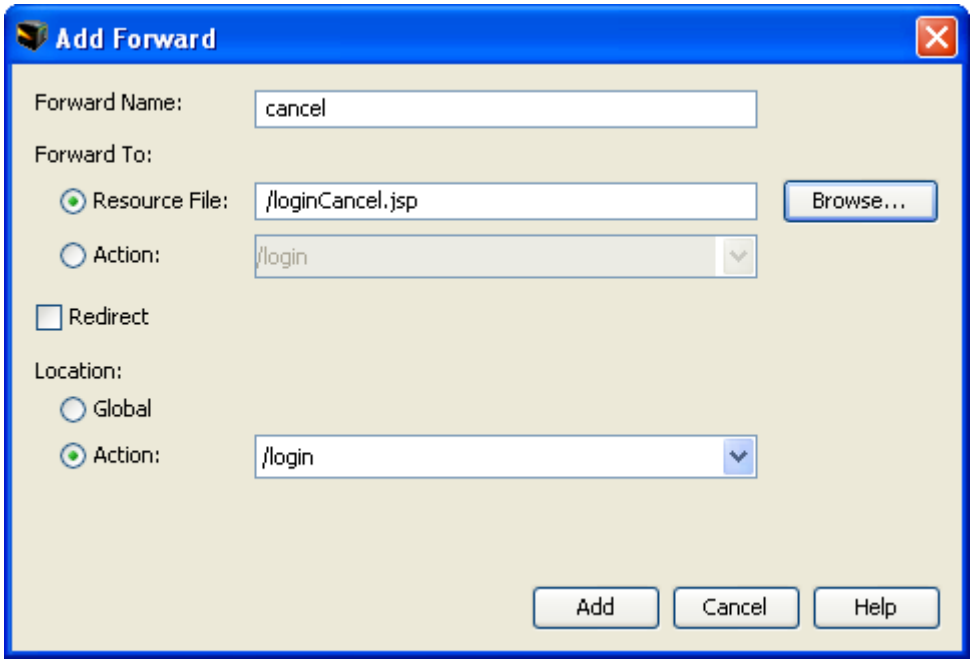
Maintenant, vous avez besoin de faire correspondre cette variable CANCEL à quelque chose. Au début de la classe Action, rajoutez la ligne suivante juste en dessous de la ligne SUCCESS:

```
private final static String CANCEL = "cancel";
```

Que devrait-il donc se produire lorsqu'on clique sur le bouton Cancel ? Ajoutez une page JSP appelée loginCancel.jsp et remplacez le texte par défaut entre les tags H1 par 'Login Cancelled!'. Maintenant, allez dans struts-config.xml, cliquez-droit n'importe où et choisissez Struts et ensuite Add Forward:



Dans la boite de dialoge Add Forward, introduisez ceci:



Lorsque vous cliquez sur Add, vous verrez cette nouvelle ligne dans l'Éditeur de Source:

```
<forward name="cancel" path="/loginCancel.jsp"/>
```

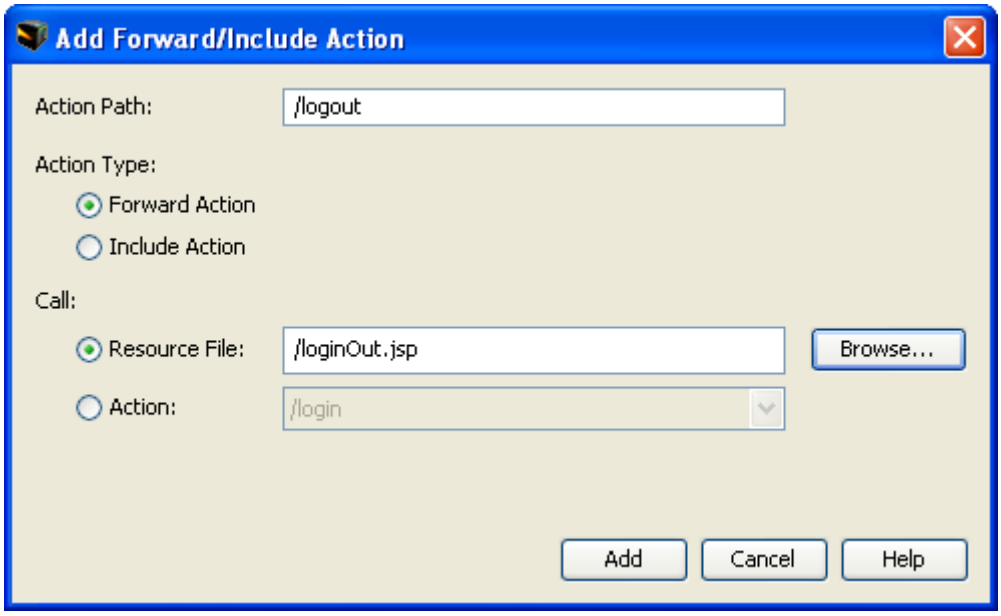
Maintenant, lorsque l'utilisateur cliquera sur 'Cancel', la méthode isCancelled sera appelée, et le struts-config.xml va faire suivre à (c'est-à-dire afficher) loginCancel.jsp

2.3.4 - Ajouter la Fonctionnalité de Logout

Dans loginSuccessful.jsp, assurez-vous que la directive taglib HTML de Struts soit présente au début de la page (comme pour loginForm.jsp). Ajoutez ensuite ce tag Struts en dessous des tags H1:

```
<html:link action="/logout" linkName="Log me out">Logout</html:link>
```

Mais que devrait-il se passer lorsqu'on clique sur ce lien ? Tout d'abord, notez que le lien ci-dessus référence une action appelée logout. Ensuite, créez une page JSP appelée loginOut.jsp et modifier les tags H1 par quelque chose comme Have a nice day!. Maintenant, lorsqu'on clique sur le lien, vous désirez que votre nouvelle page soit ouverte. Aussi, dans struts-config.xml, cliquez-droit n'importe où et choisissez Struts > Add Forward/Include Action. Spécifiez ensuite l'action que vous avez référencé dans le lien ainsi que la page JSP que vous aimeriez afficher lorsqu'on clique sur le lien.



Cliquez sur Add. Notez que vous avez maintenant défini une nouvelle action forward:

```
<action forward="/loginOut.jsp" path="/logout"/>
```

Le tac ci-dessus lie la page JSP que vous venez de créer avec le lien sur lequel l'utilisateur cliquera dans loginSuccessful.jsp pour se déconnecter de l'application.

2.3.5 - Ajouter la Fonctionnalité de Gestion d'erreur

Peu de chose sont aussi déconcertante pour un utilisateur que de voir un stack trace lorsqu'une java.lang.RuntimeException (ou toute autre exception) se produit. Vous aimeriez réduire le traumatisme causé par l'erreur. Aussi, il serait bien de créer une belle page JSP qui affichera votre erreur et pourquoi pas même quelques pistes pour une solution. Ma page de gestion d'erreur est appelée loginException.jsp. Elle contient la directive taglib pour la bibliothèque Struts HTML, et n'a rien de plus que ce qui suit entre les tags BODY:

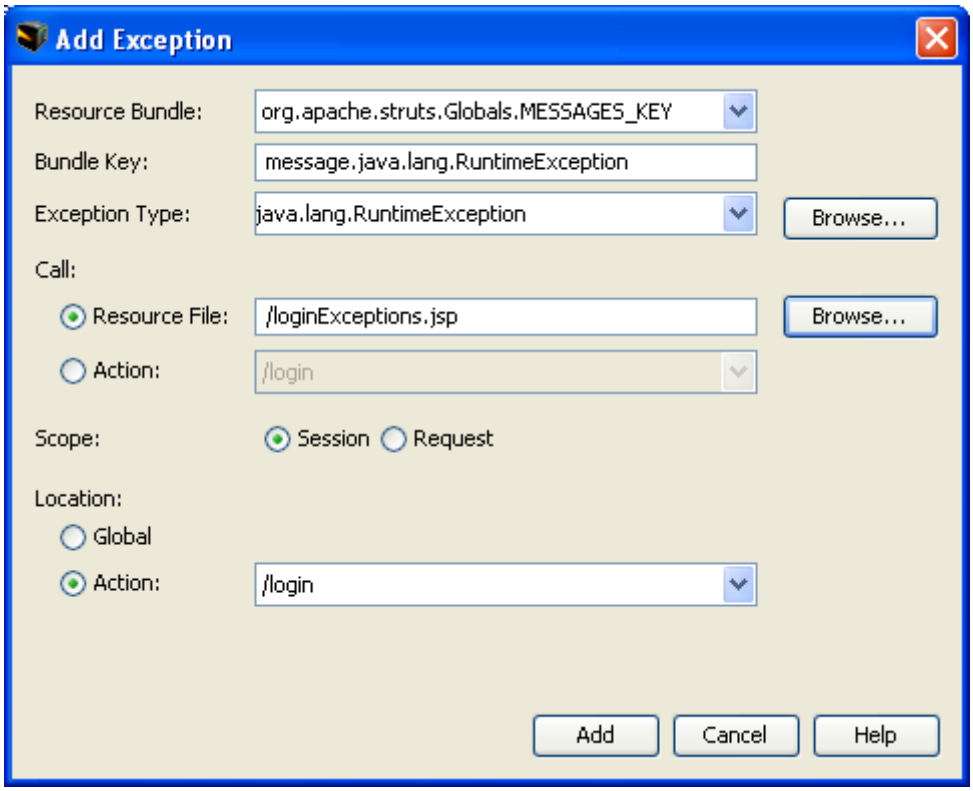
```
<body>

<hl>Login Exceptions</hl>

<html:errors />

</body>
```

Ainsi, Struts va afficher toutes les erreurs sur cette page. Mais quelle erreur ? Et comment Struts va les afficher ici ? Premièrement, forçons l'application à créer une erreur (juste pour illustrer notre exemple. Car dans la vie réelle, vous ne devrez pas forcer votre application à créer des erreurs). Retournez à la méthode isCancelled dans la classe Action. Mettez en commentaire le code qui appelle loginCancel.jsp, et remplacez le par ceci:

```
if (isCancelled(request)){
  throw new java.lang.RuntimeException();
```

```
    //return mapping.findForward(CANCEL);
  }
```

Vous pouvez voir que lorsqu'on cliquera sur le bouton Cancel une RuntimeException sera jetée, générant un horrible stack trace à la figure de l'utilisateur. Mais retournons au fichier struts-config.xml, cliquez-droit n'importe où et choisissez Struts > Add Exception. Ajoutez les valeurs suivantes dans la boite de dialogue Add Exception:



Cliquez sur Add. Notez que strusts-config.xml inclut maintenant le tag suivant:

```
<exception key="message.java.lang.RuntimeException"
               path="/loginExceptions.jsp"
               type="java.lang.RuntimeException"/>
```

Ajoutons maintenant ce qui suit au fichier ApplicationResource.properties:

```
message.java.lang.RuntimeException=There was a <b><i>java.lang.runtime.exception</i></b>!
```

Voilà. C'est fait. Lorsque l'utilisateur clique sur le bouton 'Cancel', la java.lang.RuntimeException est jetée, le struts-config.xml affiche la page loginException.jsp, et reprend le texte du message d'erreur du fichier ApplicationResource.properties.

2.4 - Conclusion

Voilà. Nous avons tout parcouru -- validation au niveau de la syntaxe, validation au niveau business, fonctionnalité d'annulation, fonctionnalité de déconnection, et fonctionnalité de gestion d'erreur. Toutes ces fonctionnalités sont très basiques et très peu sophistiquée, mais vous pouvez maintenant imaginer comment le machinerie de Struts

fonctionne et comment l'EDI NetBeans 5.0 s'intègre dedans. Un grand merci à Karel Zikmund (qui travaille actuellement pour Microsoft à Seattle) qui -- peu de temps avant de quitter l'équipe QE de NetBeans -- a créé l'application et les instructions sur lequel ce blog est basé. Karle, tu nous manques!

Laissez-moi un commentaire si vous désirez que l'on continue à traduire les parties suivantes de ce tutoriel.

Remerciements

Developpez.com a obtenu la permission de traduire en français le blog de GeertJan. GeertJan est une personne travaillant actuellement pour NetBeans et qui est chargée d'écrire des tutoriaux concernant l'EDI NetBeans 5.0. Ce tutoriel n'est que le premier tutoriel d'une longue liste. Si vous ne pouvez attendre, vous pouvez toujours vous rendre sur le blog de GeertJan.

Un merci tout particulier à GeertJan Wielenga d'avoir suggéré et accepté qu'on puisse traduire **son blog** en français.

Un merci à vedaer et autres membres de la rédaction pour la relecture du tutoriel.

Un merci particulier à vous, lecteurs, qui avez l'air d'appréciez ce genre de tutoriel.

N'oubliez pas de me faire parvenir vos réactions.

Autres Liens

Si l'Anglais ne vous fait pas peur, rendez vous sur **Le blog de GeertJan Wielenga**

Ce tutoriel est également disponible au **format PDF** (**mirroir HTTP**)