ERRATA - STRUTS

1.1 Avril 2005 - balise <html:form>

Certaines erreurs apparaissent lorsqu'on utilise la version 1.2.4 de Struts, celle-ci amenant des changements incompatibles avec la version 1.1 utilisée dans le tutoriel dans l'utilisation de certaines balises.

Ainsi, la page JSP [formulaire.personne.jsp] (page 33, paragraphe III.4.5) ne passe plus avec la version 1.2.4. La balise

```
<html:form action="/main" name="frmPersonne" type="istia.st.struts.personne.FormulaireBean">
```

doit-elle être remplacée par

Modification 1

```
<html:form action="/main">
```

les attributs [name] et [type] n'étant plus acceptés dans la balise <html:form>. Ces informations qui représentent ici le bean associé à l'action [/main] doivent maintenant être définies dans le fichier [struts-config.xml]. Celui défini paragraphe III.4.8, page 38 ne donne pas ces informations. Il y est écrit :

```
<struts-config>
  <action-mappings>
    <action
      path="/main"
      parameter="/vues/main.html"
      type="org.apache.struts.actions.ForwardAction"
    />
....
```

On voit qu'aucun bean n'est associé à l'action [/main]. On peut prendre exemple sur le fichier [struts-config.xml] de la page 40, pour remplacer le code ci-dessus par le suivant :

Modification 2

```
<struts-config>
  <form-beans>
    <form-bean
      name="frmPersonne"
      type="istia.st.struts.personne.FormulaireBean"
    />
  </form-beans>
  <action-mappings>
    <action
      path="/main"
      name="frmPersonne"
      scope="session"
      validate="false"
      parameter="/vues/main.html"
      type="org.apache.struts.actions.ForwardAction"
    />
....
```

Ici, l'action [/main] est associée au bean [frmPersonne], lui-même associé à la classe [istia.st.struts.personne.FormulaireBean].

Si on fait les modifications 1 et 2 décrites ici, le test de la vue [formulaire.personne.jsp], paragraphe III.4.8 - page 38, doit réussir.

1.1 27 Juin 2005 - balise <data-source>

Certaines erreurs apparaissent lorsqu'on utilise la version 1.2.4 de Struts, celle-ci amenant des changements incompatibles avec la version 1.1 utilisée dans le tutoriel dans l'utilisation de certaines balises.

Ainsi, page 122, paragraphe IX.2 du tutoriel on trouve le fichier [struts-config.xml] suivant :

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

```
<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">

<struts-config>
  <data-sources>
    <data-source key="dbarticles">
      <set-property property="driverClass" value="com.mysql.jdbc.Driver"></set-property>
      <set-property property="url" value="jdbc:mysql://localhost/dbarticles"></set-property>
      <set-property property="user" value="admarticles"></set-property>
      <set-property property="password" value="mdparticles"></set-property>
      <set-property property="minCount" value="2"></set-property>
      <set-property property="maxCount" value="5"></set-property>
    </data-source>
  </data-sources>

  <action-mappings>
    <action path="/liste" type="istia.st.struts.articles.ListeArticlesAction">
      <forward name="afficherListeArticles" path="/vues/listarticles.jsp"/>
      <forward name="afficherErreurs" path="/vues/erreurs.jsp"/>
    </action>
  </action-mappings>

  <message-resources parameter="istia.st.struts.articles.ApplicationResources"
    null="false" />

</struts-config>
```

Ce fichier configure une source de données Struts à l'aide de la balise <data-source>. La syntaxe de cette balise a changé avec Stuts 1.2.4 et l'application du paragraphe IX du tutoriel ne fonctionne plus, sans qu'il y ait par ailleurs de messages d'erreurs explicites. On sait simplement que la source de données [dbarticles] ci-dessus n'a pu être ouverte.

Lorsqu'on consulte la documentation de la balise <data-source> dans Struts 1.2.4, on constate que sa syntaxe a changé. Le fichier [struts-config.xml] précédent devient le suivant :

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

<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">

<struts-config>
  <data-sources>
    <!-- configuration for commons BasicDataSource -->
    <data-source type="org.apache.commons.dbcp.BasicDataSource"
      key="dbarticles">
      <set-property property="driverClassName" value="com.mysql.jdbc.Driver" />
      <set-property property="url"
        value="jdbc:mysql://localhost:3306/dbarticles" />
      <set-property property="username" value="admarticles" />
      <set-property property="password" value="mdparticles" />
      <set-property property="maxActive" value="10" />
      <set-property property="maxWait" value="5000" />
      <set-property property="defaultAutoCommit" value="false" />
      <set-property property="defaultReadOnly" value="false" />
    </data-source>
  </data-sources>
  <action-mappings>
    <action path="/liste" type="istia.st.struts.articles.ListeArticlesAction">
      <forward name="afficherListeArticles" path="/vues/listarticles.jsp"/>
      <forward name="afficherErreurs" path="/vues/erreurs.jsp"/>
    </action>
  </action-mappings>
  <message-resources parameter="istia.st.struts.articles.ApplicationResources"
    null="false" />
</struts-config>
```

Cette correction faite, on suivra la démarche du tutoriel pour déployer l'application jusqu'à son terme.

1.2 28 Juin 2005 - classe ActionErrors

La classe **ActionErrors** a subi des modifications. La façon d'ajouter une erreur à un objet **ActionErrors** décrite dans le document (par exemple page 55 mais dans beaucoup d'autres exemples) est la suivante :

```
// l'âge doit être non vide
String age = (String)this.get("age");
if (age == null || age.trim().equals("")) {
  erreurs.add("agevide", new ActionError("personne.formulaire.age.vide"));
}
```

Cette forme de la méthode [ActionErrors.add] est désormais "deprecated" et sera abandonnée dans les futures versions de Struts. On écrira plutôt :

```
// l'âge doit être non vide
String age = (String)this.get("age");
if (age == null || age.trim().equals("")) {
  erreurs.add("agevide", new ActionMessage("personne.formulaire.age.vide"));
}
```

1.3 Novembre 2005 - classe [ForwardAction]

La classe [ForwardAction] qui permettait d'indiquer lors d'une action que l'on souhaitait simplement afficher une vue ne semble plus acceptée par les versions récentes de Struts. Ainsi le fichier [struts-config.xml] de la page 22 du tutoriel :

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

<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">

<struts-config>
  <action-mappings>
    <action
      path="/main"
      parameter="/vues/main.html"
      type="org.apache.struts.actions.ForwardAction"
    />
  </action-mappings>
</struts-config>
```

doit être réécrit de la façon suivante :

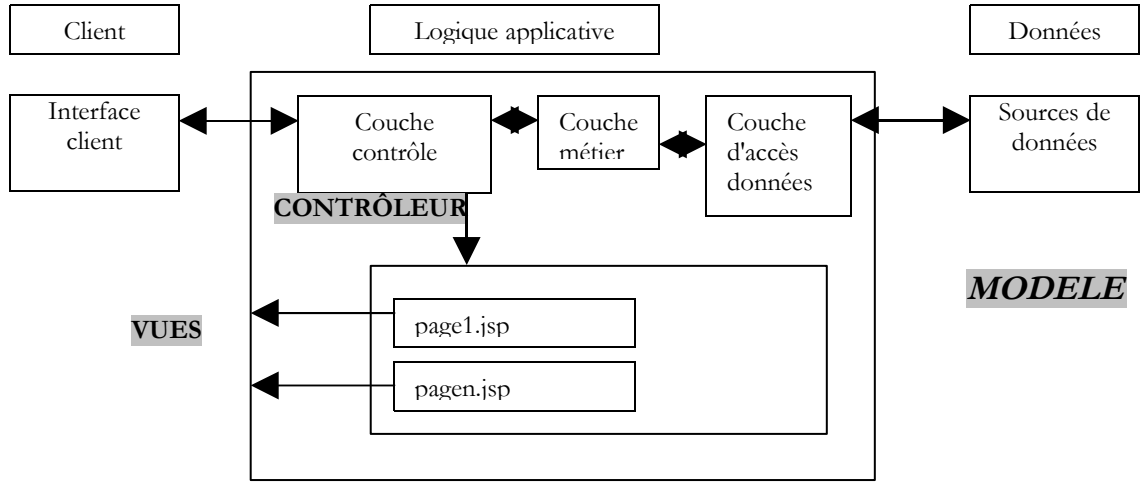
```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_2.dtd">

<struts-config>
    <action-mappings>
        <action forward="/vues/main.html" path="/main"/>
    </action-mappings>
</struts-config>
```

1.4 décembre 2005 - schéma MVC

Le schéma MVC présenté dans l'article est partiellement incorrect. Ce schéma est le suivant :

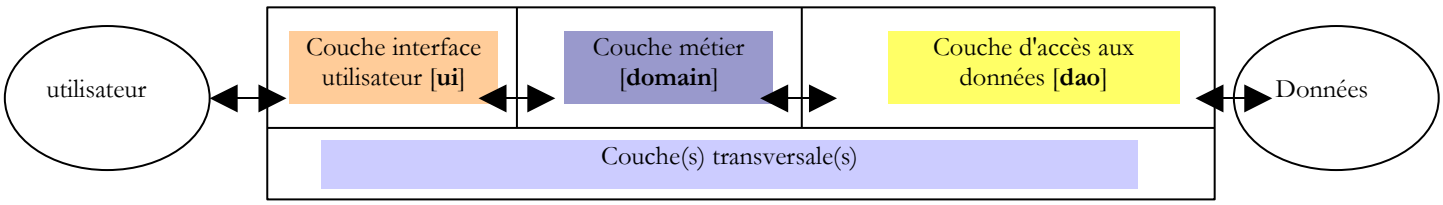


La position du modèle M dans ce schéma est incorrecte. Mon premier contact avec le modèle MVC s'est fait au travers d'un livre exposant la méthodologie Struts. Au travers de cette lecture, j'avais cru comprendre que le modèle M du MVC était constitué par le

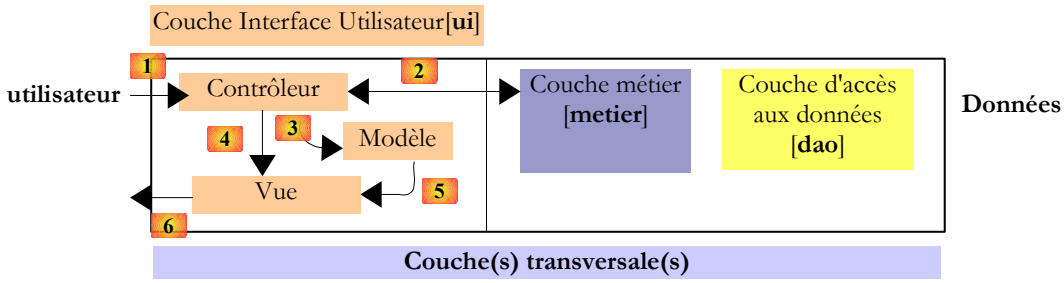
modèle des données de l'application, aussi bien le modèle des données du SGBD utilisé par la couche DAO que le modèle objet utilisé par les couches [métier] et [dao]. Cette interprétation est reflétée dans tous les articles écrits en 2004 et 2005 où le modèle est d'abord placé du côté du SGBD (article Struts) puis s'étend progressivement à gauche pour recouvrir la totalité des couches [metier] et [dao] (articles Spring / java, Spring / dotnet).

En abordant la techologie web MVC de Spring, j'ai réalisé que Spring ne conçoit pas du tout le modèle de cette façon. Le modèle M est simplement l'ensemble des données que fournit le contrôleur C à une vue V pour s'afficher. Le modèle M d'une vue est alors simplement l'ensemble de ses éléments dynamiques auxquels on doit fournir une valeur avant de demander à la vue de s'afficher. On pourrait tout simplement appeler cela les paramètres de la vue.

Un schéma plus correct serait le suivant. On a tout d'abord une architecture 3 tier :



Sur cette architecture 3tier l'architecture MVC se superpose de la façon suivante :



Le traitement d'une demande d'un client se déroule selon les étapes suivantes :

1. le client fait une demande au contrôleur qui voit passer toutes les demandes des clients. C'est la porte d'entrée de l'application. C'est le **C** de MVC.
2. le contrôleur traite cette demande. Pour ce faire, il peut avoir besoin de l'aide de la couche métier. Le contrôleur reçoit une réponse de la couche métier. La demande du client a été traitée. Celle-ci peut appeler plusieurs réponses possibles. Un exemple classique est :
    - une page d'erreurs si la demande n'a pu être traitée correctement
    - une page de confirmation sinon
3. le contrôleur choisit la réponse (= vue) à envoyer au client. Choisir la réponse à envoyer au client nécessite plusieurs étapes :
    - choisir l'objet qui va générer la réponse. Ce choix dépend en général du résultat de l'exécution de l'action demandée par l'utilisateur.
    - lui fournir les données dont il a besoin pour générer cette réponse. En effet, celle-ci contient le plus souvent des informations calculées par le contrôleur. C'est ce qu'on appelle le modèle de la vue, le **M** de MVC.

    L'étape 3 ci-dessus consiste donc en le choix d'une vue et en la construction du modèle nécessaire à celle-ci.
4. le contrôleur demande à la vue choisie de s'afficher. Il s'agit le plus souvent de faire exécuter une méthode particulière de l'objet chargé de générer la réponse au client. Ce générateur de vue est le **V** de MVC. Nous appelons vue ici, aussi bien l'objet qui génère la réponse au client que cette réponse elle-même. La littérature MVC n'est pas explicite sur ce point.
5. le générateur de vue utilise le modèle préparé par le contrôleur pour initialiser les parties dynamiques de la réponse qu'il doit envoyer au client.
6. la réponse est envoyée au client. La forme exacte de celle-ci dépend du générateur de vue. Ce peut être un flux HTML, PDF, Excel, ...

Dans le schéma ci-dessus, Struts est **entièrement** dans la couche [ui].