ACube

Prototype
Spring MVC+ACubeSerializer

Version 1.0 du 20/01/2008

Etat : xxx

SUIVI DES MODIFICATIONS

Version	Rédaction	Description	Vérification	Date
1.0	B. Leperchey	Initialisation		20/01/08
		Document validé dans sa version xxx		

LISTE DE DIFFUSION

Organisation	Nom	Info	Commentaire	Validation
		☐	☐	☐
		☐	☐	☐
		☐	☐	☐

SOMMAIRE

1 OBJET DU DOCUMENT

Ce document présente le prototype de serveur ACube basé sur Spring MVC et ACubeSerializer.

2 INTERET DE LA SOLUTION

2.1 SPRING MVC

Par rapport à Struts 2 : éviter un framework supplémentaire, qui n'est de toute façon par utilisé à 100% dans le cas ACube.

2.2 ACUBESERIALIZER

Par rapport à StrutsCX et la transformation XSLT de Struts2 :

- basé sur des bibliothèques à jour (StrutsCX est bloqué sur JDOM0.9)
- contrôle de la sérialisation des objets Java en XML, ce qui permet de se passer dans la majorité des cas de feuilles de style et, si besoin, de contrôler à 100% la génération du XML (en écrivant le code Java nécessaire)

Caractéristiques :

- petit composant : une trentaine de classe petites ou moyennes,
- dépendances limitées : aucune sur le reste de LISE, JDOM 1.0, Log4J et commons-digester (pour la compilation, mais non utilisé par le prototype)
- configurable : toute la sérialisation des objets Java est configurable ou remplaçable par du code spécifique (une interface à implémenter)
- facile à intégrer : 2 petites classes pour l'utiliser comme vue Spring.

2.3 LIMITATIONS PAR RAPPORT AU PROTOTYPE STRUTS2

API « POJO » limitée : l'interface « ThrowawayController », qui permet de définir des actions indépendantes de l'API servlet, n'a pas d'implémentation qui offre des fonctionalités de base (remplissage et validation du bean de formulaire)

L'interface « Controller », avec des implémentations proposant des fonctionalités équivalentes à celles de Struts, repose sur l'API servlet, mais les packages « mock » de Spring permettent de faire quand même des tests unitaires.

API de validation limitée : le « Validator » doit être écrit en Java, et l'API fournie par Spring est très limitée (rien à part le test de chaînes vides). La documentation de Spring explique que l'API de validation permet simplement d'utiliser d'autres bibliothèques (commons-validator). En particulier, pas de description XML des règles de validation.

3 DESCRIPTION DU PROTOTYPE

3.1 ORGANISATION DES FICHIERS

Source :

- « acube.framework.spring.support » : classes pour intégrer ACubeSerializer à Spring
- « acube.framework.strutsacube » : copie des packages pour ACubeSerializer (encore non packagé séparément), à l'identique excepté quelques détails (setters supplémentaires pour faciliter la configuration avec Spring)
- « acube.projet.action » : « traduction » des actions du prototype Strus2/Spring/IBatis
- Autres packages : identiques à ceux du proto Strus2/Spring/IBatis

Un fichier pour remplacer struts.xml : /WEB-INF/spring-dispatcher-servlet.xml. On y trouve toute la configuration du MVC.

3.2 FICHIERS DE RESSOURCES ET GESTION DES LANGUES

Standard Spring. A noter : ACubeSerializer n'a pas accès à l'ApplicationContext, il faut donc récupérer les messages dans les contrôleurs (cf Form.java)

3.3 CONTROLEURS

Standard Spring. Dans le prototype, les URLs sont mappées sur le bean du même nom. La gestion des exception est aussi centralisée, avec un ExceptionResolver spécifique qui transforme les exceptions en ErrorVO avant de les envoyer sur la vue « error ».

3.4 VUES

Un (petit) peu de code spécifique à ce niveau, pour générer

Utilisation d'un ViewResolver spécifique, basé sur UrlBasedViewResolver (qui permet de mapper le nom de la vue sur la feuille de style : la vue « x/y » est mappée sur « /WEB-INF/xsl/x/y.xsl ») : définition du type des vues (voir plus bas) et passage du « dictionnaire » ACubeSerializer.

Le ViewResolver crée des vues de type ACubeSerializerView, qui dérivent de XsltView (standard Spring) : la source de la transformation est obtenue avec ACubeSerializer au lieu d'être cherchée comme un des objets fournis.

Le fichier de configuration Spring contient aussi le paramétrage du dictionnaire ACubeSerializer, qui contrôle la transformation des objets Java en DOM. Le paramétrage par Spring n'a demandé que des adaptations cosmétiques du code ACubeSerializer (un setter supplémentaire) et une classe qui fait office d'« adaptateur ».

Il reste que le format de description des beans Spring est nettement plus bavard que le format « maison » de ACubeSerializer (qui utilise commons-digester). L'exemple fourni est plus une preuve de faisabilité qu'une solution à reprendre telle quelle.

3.5 GESTION DES EXCEPTIONS

Une classe spécifique, qui transforme l'exception en ErrorVO avant de l'envoyer vers la vue « error ».

3.6 COUCHES SERVICE, DAO

Identiques au prototype Struts2+Spring+Ibatis