

ENSEIRB

Programmation Web / XML

Evaluation 1/2

Version 1.00 du 18 décembre 2014

Etat : Travail

SOPRA GROUP

Historique :

Version	Date	Origine de la mise à jour	Rédigée par	Validée par
1.0	18/12/2014	Création	Mathieu Lombard	

Sommaire :

1. INTRODUCTION	4
1.1. L'application de test (rappel)	4
1.2. L'envoi des résultats	4
1.3. Indépendance des questions	4
2. TESTS SUR LES TDS PRÉCÉDENTS (RAPPEL)	5
3. NOUVELLES FONCTIONNALITÉS	7
3.1. Changement d'affectation d'une tâche via POST	7
3.2. Validation du flux « ownership » via XSD	8
3.3. Récupération de données via XPath	8
3.4. Formatage des données en CSV	9

1. INTRODUCTION

1.1. L'application de test (rappel)

Une application (fournie sur la page moodle du cours) permettra de tester automatiquement la réalisation des élèves.

Hypothèses de l'application de test :

- Les URLs auxquelles réagit cette WebApp sont strictement identiques à ce qui a été demandé lors des différents TDs

Pour utiliser l'application de test, il faut tout d'abord la décompresser (à l'endroit de votre choix, hors de l'arborescence du serveur web si possible) puis lancer, à la racine, la commande suivante :

```
java -jar EnseirbWebXMLEval1.jar <webAppName> <serverPort> <resultDirPath>
```

- <webAppName> correspond au nom de la WebApp, à priori « EnseirbWebXMLWebapp »
- <serverPort> correspond au port du serveur, par exemple « 8080 »
- <resultDirPath> correspond à un chemin vers un répertoire (qui doit exister) dans lequel un fichier résultat sera écrit

Il est à noter que le nom du serveur utilisé sera « localhost ».

Par ailleurs, les noms des élèves d'un binôme sont récupérés automatiquement via l'URL */about*, il faut donc bien s'assurer que celle-ci est opérationnelle (sinon, le fichier de résultat ne permettra pas **d'identifier les élèves** et cela sera considéré comme une absence de fichier).

Note importante : dans le cadre de cette évaluation, un test automatique par question (identifiable par le nom de la classe de test dans les traces de l'application) échouera afin de limiter les possibilités de tricherie.

Ainsi, à la fin de l'évaluation, **il devra y avoir autant de tests échoués qu'il y a de questions** (soit 4 en tout sur les 33 tests).

1.2. L'envoi des résultats

Une fois les exercices terminés, le fichier comportant les derniers résultats devra suivre 2 étapes :

- Être zippé (**sans le renommer** au préalable)
- Puis être envoyé à l'adresse suivante : mathieu.lombard@sopraconsulting.com

Si aucun fichier n'est reçu pour un binôme donné à l'issue du temps de l'évaluation, **la note sera alors de 0.**

1.3. Indépendance des questions

Les questions de cette évaluation sont indépendantes et peuvent être traitées dans n'importe quel ordre.

2. TESTS SUR LES TDS PRÉCÉDENTS (RAPPEL)

Une série de tests automatiques permet de vérifier la bonne implémentation des 5 TDs précédents. Notamment, les éléments suivants vont être vérifiés :

- XSD
 - Tâche
 - Les attributs « title », « deadline », « priority » et « done » sont obligatoires
 - Les attributs « id » et « creationDate » sont facultatifs (**note importante** : ces 2 attributs doivent être renvoyés dans le cas d'une lecture des tâches, mais ne sont pas obligatoires dans le cas d'une écriture car c'est la base de données qui décide de leur valeur)
 - La priorité doit être comprise entre 1 et 5 inclus
 - Aucun autre attribut que ceux cités ci-dessus n'est autorisé
 - Les seuls sous-éléments autorisés sont « description », « owner » et « asker »
 - Utilisateur
 - L'attribut « name » est obligatoire
 - Aucun sous-élément n'est autorisé
- Toutes les dates manipulées respectent le format jj/MM/aaaa (que ce soit en entrée de la WebApp ou en sortie)
- Accès aux données « brutes » (cas des url commençant par /about)
 - Cela s'effectuera via l'URL /about avec la gestion des paramètres d'URL comme indiqué dans le tableau du §1.2.2 du support de TD
 - Note : les données ne doivent pas être formatées et doivent uniquement renvoyer l'élément demandé sans ajout de texte (par exemple, pour accéder au nombre d'élèves, il faut uniquement renvoyer un chiffre et non une phrase du type « 2 élèves »)
- Accès aux données XML
 - Toutes les tâches au format XML seront accessibles via l'URL /task/list/xml
 - Une tâche dont l'id est <taskId> le sera via l'URL /task/list/xml?id=<taskId>
 - Tous les utilisateurs le seront via l'URL /user/list/xml
 - Les statistiques le seront via l'URL /stats/xml (sachant que une tâche est estimée en retard si sa « deadline » est strictement inférieure à la date du jour ; en d'autres termes si deadline => date du jour, alors on la considère comme « intime »)
- Accès aux données HTML
 - La page d'accueil sera accédée via l'URL /
 - Le formulaire d'affectation d'un professeur le sera via l'URL /about/teacher
 - La liste des utilisateurs le sera via l'URL /user/list
 - La liste des tâches le sera via l'URL /task/list
 - Le formulaire de saisie d'un nouvel utilisateur le sera via l'URL /user/create
 - Le formulaire de saisie d'une nouvelle tâche le sera via l'URL /task/create
 - Le formulaire de visualisation d'une tâche dont l'id est <taskId> le sera via l'URL /task/view?id=<taskId>
 - Le formulaire d'édition d'une tâche dont l'id est <taskId> le sera via l'URL /task/modify?id=<taskId>
- Envoi de données au serveur
 - L'affectation du professeur se fera via l'URL /about/teacher/post et des données postées (pas en paramètre d'URL et pas non plus sous la forme d'un flux XML, uniquement le nom envoyé directement sans aucun autre formatage) ; cf. §1.3.2 du support de TD.

- La création d'un utilisateur s'effectuera via l'URL /user/create/url avec des paramètres ; cf. §2.4 du support de TD
- L'ajout ou la modification d'une tâche s'effectuera via l'URL /task/create/url avec des paramètres ; cf. §2.5 du support de TD
- **Note importante** : quand l'id est passé en paramètre, le comportement doit être celui d'une modification d'une tâche déjà existante et dont l'id est celui passé en paramètre
- Quand l'id n'est pas passé, le comportement doit être celui de la création d'une nouvelle tâche

3. NOUVELLES FONCTIONNALITÉS

L'objectif des fonctionnalités ci-dessous est d'ajouter des nouveaux services au serveur sur la manipulation des affectations tâches / responsables (task / owner).

Le format du flux XML qui sera utilisé est le suivant :

```
<changeowner taskId="1" newOwnerName="newOwner"/>
```

3.1. Changement d'affectation d'une tâche via POST

Classe de test	OwnershipPostTestCase
Nombre de tests	3

Il s'agit de pouvoir modifier le champ « owner » d'une « task » en interprétant le flux XML ci-dessus et en répondant aux URLs suivantes :

- /task/user/changeowner/post
- /user/task/changeowner/post

Il faut alors :

- Créer une Servlet OwnershipServlet et la référencer dans le fichier web.xml selon les URLs ci-dessus
- La faire réagir sur les 2 URLs pour qu'elle parse le flux afin de récupérer les valeurs d'id de la tâche et du nouveau responsable
- Vous pouvez vous inspirer du code suivant en remplaçant les <???

```
String postData = <??>;

Document docPostData = XMLToolkit.parseDocument(postData);
Element ownerShipElt = docPostData.getDocumentElement();

int taskId = Integer.parseInt(ownerShipElt.getAttribute(<??>));
String userName = ownerShipElt.getAttribute(<??>);

String taskXML = XMLMediator.getTask(taskId);
Document docTask = XMLToolkit.parseDocument(taskXML);
Element taskElt = docTask.getDocumentElement();

((Element) taskElt.getElementsByTagName(<??>).item(0)).setTextContent(userName);

XMLMediator.addOrModifyTask(taskElt);
```

3.2. Validation du flux « ownership » via XSD

Classe de test	OwnershipXSDTestCase
Nombre de tests	7

Il s'agit ici de valider les données utilisées par le POST précédent en offrant un service sur les URLs suivantes :

- /task/user/changeowner/post/isValid
- /user/task/changeowner/post/isValid

Il faut alors :

- Compléter le web.xml et la servlet OwnershipServlet pour réagir à ces 2 URLs
- Créer un schéma XSD qui validera que
 - taskId et newOwnerName sont 2 attributs obligatoires
 - taskId doit être un entier
 - newOwnerName doit être une chaîne de caractères
 - aucun autre attribut ou enfant n'est autorisé
- Formater puis renvoyer le résultats de la validation via la méthode *XMLToolkit.createPostResult(message, success)*
 - message est une chaîne de caractères indiquant un message libre
 - success est le booléen renvoyé par *XMLToolkit.isXMLValid*

3.3. Récupération de données via XPath

Classe de test	OwnershipXPathTestCase
Nombre de tests	8

Il s'agit ici de permettre au client de récupérer les données suivantes :

- La liste des id des tâches pour un *owner* donné via une URL du type */user/ownership?ownerName=<owner>*
- Le *owner* affecté à un id de tâche donnée via une URL du type */task/ownership?taskId=<id>*

Il faut alors :

- Compléter le web.xml et la servlet OwnershipServlet pour réagir à ces 2 URLs
 - Pour que le résultat soit bien interprété en texte, la Servlet devra contenir le code suivant pour ces 2 URLs :

```
response.setHeader("Content-Type", "text/plain");
```

- Utiliser des requêtes XPath en se basant sur
 - Le flux des tâches, récupéré via *XMLMediator.getTasks()*
 - Les paramètres passés dans l'URL
- Renvoyer la chaîne de caractères issue de la méthode *sortAndFormat* de *StringFormatUtil* qui prend en paramètre la liste de String renvoyée par *XMLToolkit.getXPathValues*

3.4. Formatage des données en CSV

Classe de test	OwnershipCSVTestCase
Nombre de tests	2

Il s'agit ici d'offrir au client la possibilité d'obtenir des données formatées en CSV (avec le caractère « ; » pour séparer les données).

La méthode utilisée sera de passer par une transformation XSL qui, à partir d'un flux XML, renverra du « texte » au format CSV.

Afin que le moteur XSL n'insère pas de balise XML dans le flux résultat, la feuille XSL devra contenir l'instruction suivante après la balise *xsl:stylesheet* :

```
<xsl:output method="text" omit-xml-declaration="yes"/>
```

De même, la Servlet (qui traite la demande du client) devra indiquer au client (le navigateur) que le flux de données correspond à du texte ; cela s'effectue via l'instruction suivante :

```
response.setHeader("Content-Type", "text/plain");
```

Afin de pouvoir créer un retour à la ligne (nécessaire en fin de chaque ligne), l'instruction suivante pourra être utilisée :

```
<xsl:text>  
</xsl:text>
```

Pour information, les espaces et les tabulations sont ignorés lors des comparaisons avec le résultat attendu lors du passage des tests automatiques.

Pour cet exercice, il faut alors :

- Compléter le fichier web.xml et la servlet OwnershipServlet pour réagir à l'URL */task/ownership/csv* et renvoyer le flux CSV en se basant sur une transformation XSL
- Créer un nouveau fichier XSL qui, à partir du flux des tâches (obtenu via la méthode *getTasks()* de *XMLMediator*) formate les données comme suit :

```
taskId;owner  
id1;owner1  
id2;owner2
```

- Où la 1^{ère} ligne correspond aux intitulés des colonnes (texte en dur)
- Où chaque ligne indique les informations suivantes (dans l'ordre et en séparant chaque donnée d'un point virgule) relatives à une tâche
 - Son id
 - Son owner
- La liste doit être triée selon les ids des tâches