TP XML

1 Procédure de préparation de l'environnement

Lancer Jbuilder .
Eventuellement, fermer les projets qui se sont ouverts automatiquement.

Procédure d'ajout de la bibliothèque Xerces dans Jbuilder

1. Dans le menu « **Projet** », choisir « **Propriétés du projet par défaut** ».
2. Dans l'onglet « **Chemin** », vérifier si Xerces est déjà installé. Si tel est le cas, vous pouvez quitter la procédure d'ajout de la bibliothèque Xerces dans JBuilder.
3. Appuyer sur le bouton « **Ajouter** » puis sur le bouton « **Nouveau** » de la nouvelle fenêtre.
4. La fenêtre « **Bibliothèque Java à ajouter** » apparaît.
5. Saisir « Xerces » pour le champ « **Nom** ».
6. Cliquer sur « **…** » à coté de la zone de saisie « **Chemin de la classe** ».
7. Dans la fenêtre « **Modification du chemin de classes de la bibliothèque** », cliquer sur « **Ajouter une archive** ».
8. Dans la fenêtre de sélection de fichier, chercher et sélectionner le fichier « **xerces.jar** ».
9. Valider et retourner dans la fenêtre « **Bibliothèque Java à ajouter** ».
10. Ne rien saisir dans le champ « **Chemin des sources** ».
11. Cliquer sur « **…** » à coté de la zone de saisie « **Chemin de la documentation** ».
12. Dans la fenêtre « **Modification du chemin de la documentation de la bibliothèque** », cliquer sur « **Ajouter une archive** ».
13. Dans la fenêtre de sélection de fichier, chercher et sélectionner le fichier « **xerces-apidoc.zip** ».
14. Valider la fenêtre ainsi que la fenêtre « **Propriétés du projet par défaut** ».

Pour ce TP, n'hésitez surtout pas à consulter l'aide en ligne de Xerces :

1. Positionnez votre souris sur la déclaration d'une classe que vous souhaitez utiliser (par exemple, une classe située au niveau des imports) puis affichez le menu contextuel (bouton droit de la souris) et sélectionnez « Chercher le symbole sous le curseur »
2. Vous obtenez alors la visualisation du squelette de la classe.
3. Sélectionnez l'onglet « doc » en bas du squelette de la classe et vous verrez la documentation en ligne associée à cette classe.

2 Etape 1

Des erreurs se sont glissées dans le document XML «data/personal.xml ». Votre objectif consiste à corriger ce document.

Pour ce faire, vous allez valider un document XML vis à vis de sa DTD à l'aide de l'API SAX 2.

Les classes définies dans le package tpxml.etape1 constituent le support de travail.

Classe MonErrorHandler

Complétez cette classe afin d'émettre dans le flux de sortie (attribut out de cette classe) les messages d'avertissement, d'erreur récupérable et d'erreur non-récupérable avec les comportements personnalisés suivants :

- pour chaque type d'erreur, vous devez indiquer :
  - le type d'erreur,
  - le nom du fichier xml dans lequel se trouve l'erreur,
  - la ligne et la colonne où se situe l'erreur dans le fichier,
  - le message associé à l'erreur,
- pour les avertissements et les erreurs récupérables : les traitements de validation doivent continuer,
- pour les erreurs fatales : arrêtez définitivement les traitements après avoir signalé l'erreur.

Conseils :

- *pour émettre des messages dans le flux de sortie, utilisez la fonction out.println (…)*
- *pour quitter définitivement les traitements, utilisez la fonction exit(…) de la classe System*
- *consultez l'aide en ligne de SAXParseException pour atteindre les caractéristiques des erreurs.*

Classe Validateur

Complétez la procédure main(String[] args) afin :

- D'instancier le parser sax dont le nom de classe est "org.apache.xerces.parsers.SAXParser",
  - D'indiquer au parser que la lecture doit valider le document,
  - D'enregistrer le handler d'erreur,
  - De lancer la lecture du document.

Conseils : aidez-vous de l'aide en ligne, les importations définies dans l'en-tête du fichier vous montrent tout ce dont vous avez besoin.

Lorsque vous n'avez plus d'erreur dans le document xml, vous pouvez passer à l'étape suivante.

3 Etape 2

Cette étape consiste à transformer le document xml précédent en un document HTML en utilisant l'interface ContentHandler de l'API SAX 2.

La conversion se base sur l'exemple suivant : la conversion du fichier xml ci-dessous

```
<ELEMENT1  attr1="toto" attr2="deuxieme">
        <ELEMENT2 attr1="attribut">
                texte1
        </ELEMENT2>
        texte2
        <ELEMENT3 attr1="attribut" />
</ELEMENT1>
```

doit aboutir au document HTML suivant :

```
<!DOCTYPE HTML SYSTEM "essai.dtd">
<HTML><HEAD><TITLE> Creation d'un document HTML simple a partir de SAX
</TITLE></HEAD>
<BODY>
<ul>
        <li><b>ELEMENT1</b></li>
        <ul>
                <li>attr1=toto</li>
                <li>attr2=deuxieme</li>
                <li><b>ELEMENT2</b></li>
                <ul>
                        <li>attr1=attribut</li>
                        <li> texte : texte1</li>
                </ul>
                <li> texte : texte2</li>
                <li><b>ELEMENT3</b></li>
                <ul>
                        <li>attr1=attribut</li>
                </ul>
        </ul>
</ul>
</BODY>
</HTML>
```

Les classes définies dans le package tpxml.etape2 constituent le support de travail.

Classe MyContentHandler

Complétez cette classe afin d'implémenter les événements de lecture qui permettent la construction du fichier destination (à travers l'attribut out de cette classe).

Classe SimpleHTMLWriter

Compléter la procédure main(String[] args) afin :

- D'instancier le parser sax dont ne nom de classe est "org.apache.xerces.parsers.SAXParser",
  - D'indiquer au parser que la lecture doit valider le document,
  - D'enregistrer le handler de lecture,
  - De lancer la lecture du document.

Conseils : vous pouvez valider le document HTML produit avec la classe Validateur de l'étape précédente afin de vérifier la conformité du document vis-à-vis de la DTD « essai.dtd »

4 Etape 3

Dans cette étape, vous devrez :

- Charger un document xml pour obtenir un DOM.
- Ecrire le DOM ainsi obtenu dans un autre fichier xml.

Les classes définies dans le package tpxml.etape3 constituent le support de travail.

Classe DOMWriter

Cette classe est instanciée en spécifiant le flux de sortie.
La méthode writeDocument (…) doit être complétée de manière à écrire le DOM dans le flux de sortie.

L'écriture du DOM doit produire un document xml conforme et consiste à écrire :

- l'entête du document xml : « <?xml version="1.0" encoding="UTF-8"?> »
- la ligne doctype à l'aide des informations contenues dans le DOM <!DOCTYPE HTML SYSTEM "nomFichierDTD">
- à écrire les balises ouvrantes et fermantes de chaque élément
- à écrire les attributs de chaque élément
- à écrire les portions de texte

Conseils :

- *L'entête peut être écrite en dur par l'application.*
- *Pour la ligne DOCTYPE, le nom du fichier DTD peut être déterminé à l'aide de la fonction Document.getDocType().getSystemId().*

Classe LecteurDOM

Complétez la procédure main(String] args) de cette classe afin

- D'instancier le parser DOM (org.apache.xerces.parsers.DOMParser),
- De parser le fichier source,
- De récupérer la racine du DOM produit lors de la lecture du fichier source,
- D'exécuter la méthode writeDocument de l'instance de DOMWriter.