Import/Export de données au format XML

Le but de ce TP est de permettre l'importation et l'exportation de données XML dans une base de données (BD). Nous considérerons une BD composée de deux tables *Equipe* et *Match* définies de la manière suivante :

Equipe(*id integer*, *nom varchar*2(100), *ville varchar*2(100))
Match(*#equipeL integer*, *#equipeV integer*, *dateMatch date*, *butsL integer*, *butsV integer*)

Un exemple de fichier XML qui pourra être importé ou être le résultat d'un export est disponible à l'adresse `http://www.lisi.ensma.fr/ftp/enseignement/Oracle/L3PRO/` (`matchs.xml`).

A- Conception d'une DTD XML

A-1. Créez les tables *Match* et *Equipe* en utilisant le script `create_table.sql` ;

A-2. Ecrivez une DTD `matchs.dtd` correspondant au fichier `matchs.xml`. Vous devrez pour cela consulter le cours sur la partie XML ;

A-3. Associez le fichier XML à votre DTD par une référence externe ;

A-4. Vérifiez que le fichier XML valide bien votre DTD en utilisant un programme JAVA (voir aide ci-dessous).

Aide pour valider un fichier XML

Pour valider du XML vous pourrez utiliser l'API dom4j (`http://dom4j.sourceforge.net`). Le code suivant montre comment valider le fichier `matchs.xml` qui référence une DTD.

```
SAXReader reader = new SAXReader();
reader.setValidation(true);
Document document = reader.read("C:\\matchs.xml");
```

L'exécution de ce code nécessite d'ajouter l'archive `dom4j-1.6.1.jar` (section download du site `http://dom4j.sourceforge.net`) au classpath de votre projet JAVA. Si le fichier est valide par rapport à la DTD ce code s'exécutera sans lever d'exception (et inversement si le fichier est invalide).

B- Import de données XML

B-1. Ecrivez une programme JAVA permettant de lire un fichier XML respectant la DTD écrite dans la partie précédente et d'insérer les données correspondantes dans la table `Match`.

Aide pour lire le fichier XML

Pour lire du XML, vous pourrez également utiliser l'API dom4j. Voici un exemple de code qui montre comment afficher la date de chaque journée de championnat stockée dans le fichier `matchs.xml` (d'autres exemples sont disponibles à l'adresse `http://dom4j.sourceforge.net/dom4j-1.6.1/guide.html`) :

```
SAXReader reader = new SAXReader();
Document doc = reader.read(new File("C:\\matchs.xml"));
Element matchsElement = doc.getRootElement();
Iterator<Element> it = matchsElement.elementIterator();
while(it.hasNext()) { // pour chaque journée
    Element journeeElement = it.next();
        // affichage de la date de la journée
    System.out.println(journeeElement.attributeValue("date");
}
```

Aide pour la connexion à la Base de Données Oracle et l'insertion de données

Pour la connexion à Oracle, il faut ajouter l'archive `ojdbc6.jar` (disponible sur vos machines dans le répertoire locale `C->app->user->product ... JDBC->lib`).

Si votre base de données a pour nom `orcl`, la connexion se fait par le code suivant :

```
String urlOracle = "jdbc:oracle:thin:@192.168.49.6:1521:orcl";
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection cnxOracle = DriverManager.getConnection(urlOracle, "L3PROX"
    , "L3PRO");
```

Une fois un objet `connection` obtenu vous pourrez exécuter des instructions SQL via un objet `PreparedStatement` (voir `http://java.developpez.com/faq/jdbc/?page=instructions`). Le code suivant montre comment insérer une ligne dans la table `Match` (votre table `Match` utilisant des identifiants en clés primaires, vous pourrez utiliser des sous-requêtes dans l'instruction insert) .

```
/* Creation du PreparedStatement */
PreparedStatement cmdUpdate = cnxOracle.prepareStatement(
    "INSERT␣INTO␣Match␣(equipeL,␣equipeV,␣butsL,␣butsV)␣VALUES␣(?,␣?,␣
       ?,␣?");

/* Remplacement des ? par des valeurs */
cmdUpdate.setString(1,"OM");
cmdUpdate.setString(2,"PSG");
cmdUpdate.setInt(3,2);
cmdUpdate.setInt(4,2);

/* Execution de la requete */
cmdUpdate.executeUpdate();
```

Aide pour la conversion de types

Le code suivant montre comment créer une date (`java.sql.Date`) à partir d'une chaîne de caractères (`dateString`) :

```
SimpleDateFormat sd = new SimpleDateFormat("dd-MM-yyyy");
Date d = new Date(sd.parse(dateString).getTime());
```

Le code suivant montre comment créer un entier à partr d'une chaîne de caractères (`stringValue`) :

```
int resultat = Integer.parseInt(stringValue);
```

C- Export de la base de données au format XML

C-1. Ecrivez une programme JAVA permettant d'exporter les données de la table `Match` dans un format XML correspondant à la DTD écrite section A.

Aide pour la lecture de données dans la Base de Données Oracle

Le code suivant illustre comment exécuter une requête sur la table `Match` et afficher le résultat :

```
/* Creation du Statement */
Statement cmdSelect = cnxOracle.createStatement();

ResultSet rset = cmdSelect.executeQuery("SELECT␣*␣From␣Match");
while (rset.next()) { // pour chaque ligne du résultat
    // affichage de la valeur de la première colonne du résultat
    System.out.println(rset.getString(1));
}
```

Aide pour la création du fichier XML

Le code suivant permet de rédiriger l'affichage de l'instruction `System.out.println` dans le fichier `matchs_export.xml` (qui sera automatiquement créé).

```
OutputStream output = new FileOutputStream("matchs_export.xml");
PrintStream printOut = new PrintStream(output,true,"UTF8");
System.setOut(printOut);
```

Aide pour la manipulation des dates

Vous utiliserez la classe `GregorianCalendar` décrite à la page : `https://docs.oracle.com/javase/7/docs/api/java/util/GregorianCalendar.html`. Par exemple, le code suivant permet de récupérer la journée d'une date donnée (`date`) sur deux caractères :

```
Calendar calendar = new GregorianCalendar();
calendar.setTime(date);
NumberFormat formatter = new DecimalFormat("00");
String jour = formatter.format(calendar.get(Calendar.DAY_OF_MONTH));
```