GML und Bibliothek oracle.sdoapi

Nachfolgend werden die Java-Klassen aus dem GML-Kapitel unter Verwendung der alten Klassenbibliothek `oracle.sdoapi` dargestellt.

Erzeugung von GML 3.1 aus SDO_GEOMETRY-Objekten

```
// Klasse zur Erzeugung von GML aus SDO_GEOMETRY-Objekten
import java.sql.*;             // Import der JDBC-Klassen
import oracle.sql.STRUCT;      // Import von STRUCT
// Import der Klassen des SDO-API:
import oracle.sdoapi.OraSpatialManager;
import oracle.sdoapi.adapter.GeometryAdapter;
import oracle.sdoapi.geom.*;

public class GenerateGML {
   static private Connection db;             // Die Datenbankverbindung.
   static private GeometryAdapter adapter;   // Der Adapter.
  // Initialisierung der Klasse.
  static {
    try {
      // Verbindung zur Datenbank aufbauen
      db = DriverManager.getConnection("jdbc:default:connection:");
      // GeometryAdapter holen
      adapter = OraSpatialManager.getGeometryAdapter
                                ("SDO","8.1.6",STRUCT.class,null,null,db);
    } catch (Exception ex) {
      System.err.println("Fehler: "+ex);
    }
  }

  // Umwandlung einer SDO_GEOMETRY in GML 3.1
  public static String toGML (STRUCT spatialObj) {
    try {
      Geometry geom = adapter.importGeometry(spatialObj);
      if (geom instanceof Point) {
        Point point = (Point)geom;
        return "<Point><pos>" + point.getX() + " " + point.getY() +
               "</pos></Point>";
      }
      else if (geom instanceof LineString) {
        LineString line = (LineString)geom;
        String res = "<LineString><posList>";
        for (int i=0; i<line.getNumPoints(); i++) {
          CoordPoint p = line.getPointAt(i);
          res += p.getX() + " " + p.getY() + " ";
        }
        res += "</posList></LineString>";
        return res;
      }
```

```
      else if (geom instanceof CurveString) {
        CurveString line = (CurveString)geom;
        String res = "<CompositeCurve>";
        Segment[] seg = line.getSegmentArray();
        // Bestandteile des zusammengesetzten Linienzugs bearbeiten
        for (int i=0; i<seg.length; i++) {
          res += "<curveMember>";
          // Fall 1: lineares Segment
          if (seg[i] instanceof LinearSegment) {
            LinearSegment l = (LinearSegment)seg[i];
            res += "<LineString><posList>" +
              l.getStartPoint().getX()+" " + l.getStartPoint().getY()+" " +
              l.getEndPoint().getX() + " " + l.getEndPoint().getY() +
              "</posList></LineString>";
          }
          // Fall 2: Kurvensegment
          else {
            CircularArc arc = (CircularArc)seg[i];
            res += "<Curve><segments><ArcString><posList>" +
              a.getStartPoint().getX()+" " + a.getStartPoint().getY()+" " +
              a.getMidPoint().getX() + " " + a.getMidPoint().getY() + " " +
              a.getEndPoint().getX() + " " + a.getEndPoint().getY() +
              "</posList></ArcString></segments></Curve>";
          }
          res += "</curveMember>";
        }
        res += "</CompositeCurve>";
        return res;
      }
      return "<Error>" + geom.getClass().getName() +
             " nicht unterstützt.</Error>";
    } // try
    catch (Exception ex) {
      return "<Error>" + ex + "</Error>";
    }
  }
}
```

Die Klasse besitzt mit der Datenbankverbindung `db` und dem Adapter `adapter` zwei Attribute, die beim Laden der Klasse initialisiert werden. Da die Klasse in Oracle gespeichert werden soll, kann der *JDBC Server-Side Internal Driver* als JDBC-Treiber verwendet werden. Der Adapter erlaubt die Umwandlung in Geometrien des Pakets `oracle.sdoapi.geom`. Dies erfolgt in der Klassenmethode `toGML`, die als Parameter ein `SDO_GEOMETRY`-Objekt vom Typ `oracle.sql.STRUCT` erhält. Nach der Umwandlung in ein `Geometry`-Objekt wird je nach Geometrietyp die GML-Zeichenkette gebildet. Die Beispielmethode zeigt dies für Punkte und Linienzüge auf; für die anderen Geometrietypen kann die Erzeugung in analoger Weise programmiert werden.

Damit die Klasse in der Datenbank ausgeführt werden kann, müssen die Zip-Datei mit der `sdoapi`-Bibliothek und die Klasse `GenerateGML` über das Werkzeug `loadjava` von der Kommandozeile aus geladen werden:

```
loadjava -user <Benutzer>/<Passwort>@<Dienstname> -resolve sdoapi.zip
loadjava -user <Benutzer>/<Passwort>@<Dienstname>
         -andresolve GenerateGML.java
```

Außerdem wird eine PL/SQL-Funktion benötigt, die auf die Java-Methode `toGML` verweist:

```
CREATE OR REPLACE FUNCTION gml (geom MDSYS.SDO_GEOMETRY) RETURN VARCHAR2 AS
   LANGUAGE JAVA NAME 'GenerateGML.toGML (oracle.sql.STRUCT)
                                              return java.lang.String';
/
```

Diese Funktion `gml` kann nun in SQL-Anweisungen genutzt werden, wobei nur für die von der Java-Methode `toGML` berücksichtigten Klassen GML erzeugt wird.

```
SELECT name, gml(geo) FROM GeoDbLand;
NAME        GML(GEO)
----------- ----------------------------------------------------------------
Flaggenmast <Point><pos>10.0 11.0</pos></Point>
Steg        <LineString><posList>
              4.0 4.0 5.0 7.0 8.0 8.0
            </posList></LineString>
Baumreihe   <Error>
              oracle.sdoapi.geom.impl.MultiPointImpl nicht unterstützt.
            </Error>
Wall        <CompositeCurve><curveMember><Curve><segments><ArcString>
            <posList>1.0 15.0 2.414 15.586 3.0 17.0</posList>
            </ArcString></segments></Curve></curveMember>
            <curveMember><Curve><segments><ArcString>
            <posList>3.0 17.0 3.586 18.414 5.0 19.0</posList>
            </ArcString></segments></Curve></curveMember>
            <curveMember><LineString>
            <posList>5.0 19.0 7.0 19.0</posList>
            </LineString></curveMember></CompositeCurve>
Haus        <Error>
              oracle.sdoapi.geom.impl.PolygonImpl nicht unterstützt.
            </Error>
Land        <Error>
              oracle.sdoapi.geom.impl.MultiPolygonImpl nicht unterstützt.
            </Error>
```

Überführung von GML-Punkten in die Tabelle „POIPositionen“

Die Überführung eines GML-Koordinatenelements ist am einfachsten in einer Java-Methode zu programmieren. Im nachfolgenden Beispiel erhält die Klassenmethode `insertPos` den Text, der in einem `pos`-Element enthalten ist, um daraus einen `SDO_GEOMETRY`-Punkt zu erzeugen. Dieser Punkt wird dann zusammen mit dem Gemeindekennzeichen und dem POI-Namen in die Tabelle „POIPositionen“ eingefügt.

```
// Klasse für Überführung von GML-Punkten in die Tabelle POIPositionen
import java.sql.*;                      // Import der JDBC-Klassen
import oracle.sql.STRUCT;               // Import von STRUCT
import oracle.sdoapi.OraSpatialManager; // Import von SDO-API
import oracle.sdoapi.adapter.GeometryAdapter;
import oracle.sdoapi.geom.*;
```

```
import oracle.sdoapi.sref.SpatialReference;
import oracle.sdoapi.sref.SRManager;

public class ProcessGML {
   static private Connection db;             // Die Datenbankverbindung.
   static private GeometryAdapter adapter;   // Der Adapter.
   static private GeometryFactory gf;        // Der Geometrie-Erzeuger.
   static private PreparedStatement insert;  // Einfüge-Anweisung.
  // Initialisierung der Klasse.
  static {
    try {
      // Verbindung zur Datenbank aufbauen
      db = DriverManager.getConnection("jdbc:default:connection:");
      // GeometryAdapter, Bezugssystem und GeometryFactory holen
      adapter = OraSpatialManager.getGeometryAdapter
                    ("SDO","8.1.6",STRUCT.class,STRUCT.class,null,db);
      SRManager srm = OraSpatialManager.getSpatialReferenceManager(db);
      SpatialReference cs = srm.retrieve(4326);   // vor 10.2: 8307
      gf = OraSpatialManager.getGeometryFactory(cs);
      // Insert vorbereiten
      String sql =
                "INSERT INTO POIPositionen (gkz,name,pos) VALUES (?,?,?)";
      insert = db.prepareStatement(sql);
    }
    catch (Exception ex) {
      System.err.println("Fehler: " + ex);
    }
  }

  // Einfügen der Koordinaten in die Tabelle "POIPositionen".
  public static void insertPos (int gkz, String name, String coords) {
    try {
      System.out.print("insertPos " + gkz + " " + name +
                       " " +name.length());
      insert.setInt(1,gkz);
      insert.setString(2,name);
      coords = coords.trim();
      double x = Double.parseDouble(
                                coords.substring(0,coords.indexOf(' ')));
      double y = Double.parseDouble(
                             coords.substring(coords.lastIndexOf(' ')+1));
      Point poi = gf.createPoint(x,y);
      Object obj = adapter.exportGeometry(STRUCT.class,poi);
      insert.setObject (3,obj);
      insert.executeUpdate();
    } // try
    catch (Exception ex) {
      System.out.print("Fehler: " + ex);
    }
  }
}
```

Bei der Initialisierung der Klasse wird zusätzlich gegenüber dem vorherigen Beispiel eine `GeometryFactory` angelegt und eine `INSERT`-Anweisung als `PreparedStatement` vorbereitet,

die es erlaubt, einen neuen Datensatz in die Tabelle „POIPositionen“ einzufügen. In der Methode `insertPos` wird aus der Zeichenkette `coords` der x- und der y-Wert extrahiert, ein neuer Punkt der Klasse `oracle.sdoapi.geom.Point` erzeugt, dieser in den Typ `oracle.sql.STRUCT` umgewandelt und dann ein entsprechender neuer Datensatz in die Tabelle „POIPositionen“ eingefügt.