GeoTools 3D Extension Guide

출시/ 0.5.0

Soojin Kim, Hyung-Gyu Ryoo

2018년 08월 29일

Modules

1 목차 **3**

1.1 개요 3
1.2 Geometry 4
1.3 DataStore 5
1.4 XSD-GML 6
1.5 빌드 6
1.6 빠르게 시작하는 메뉴얼 7

GeoTools 3D Extension에 방문하신 것을 환영합니다. 이 프로젝트는 GeoTools 라이브러리를 확장하여 3차원 공간 연산을 지원합니다. 이 문서에서는 이 프로젝트에 대한 간단한 설명, 프로젝트를 빌드하는 방법, 그리고 몇가지 튜토리얼을 다루고 있습니다.

CHAPTER 1

목차

1.1 개요

이 프로젝트는 GeoTools의 기능을 확장하여 3차원 기하를 저장하고 처리할 수 있도록 개발한 확장 라이브러리입니다. GeoTools는 지리공간 데이터를 위한 여러가지 툴을 제공하는 대표적인 오픈소스 자바 라이브러리입니다. 그러나 현재 3차원 공간 데이터를 다루는 기능은 거의 지원되지 않고 있습니다. 예를 들어 3차원 기하인 Solid 기하 정보를 저장하는 것이 지원되지 않고, Z 좌표를 가진 기하를 대상으로 공간 질의를 수행하는 경우 Z 좌표를 고려하지 않고 처리되고 있습니다. 그 이유는 GeoTools가 Java Topology Suite(JTS) 라이브러리를 기하 정보를 다루기 위한 데이터 구조로 사용하고 있기 때문입니다.

따라서 프로젝트는 Java Topology Suite(JTS) 라이브러리 대신 ISO 19107 공간 스키마 기반의 기하 라이브러리를 사용하여 3차원 공간 데이터를 사용할 수 있도록 하여 GeoTools에서 제공하고 있는 여러가지 기능들을 확장하였습니다.

이 프로젝트는 다음의 기능들을 제공하는데 초점을 맞추고 있습니다.

- 3차원 기하를 저장이 가능한 데이터 구조
- 3차원 객체에 대한 질의 처리 기능
- 3차원 공간 정보를 저장하기 위한 데이터 저장소의 연결
- 3차원 공간 정보의 공유를 위한 표준 웹 프로토콜 해석 기능 지원

1.1.1 라이센스

이 프로젝트의 라이센스는 다음과 같이 GeoTools의 라이센스를 그대로 따릅니다.

- GNU Lesser General Public License

1.1.2 참조

- GeoTools

- Java Topology Suite

1.2 Geometry

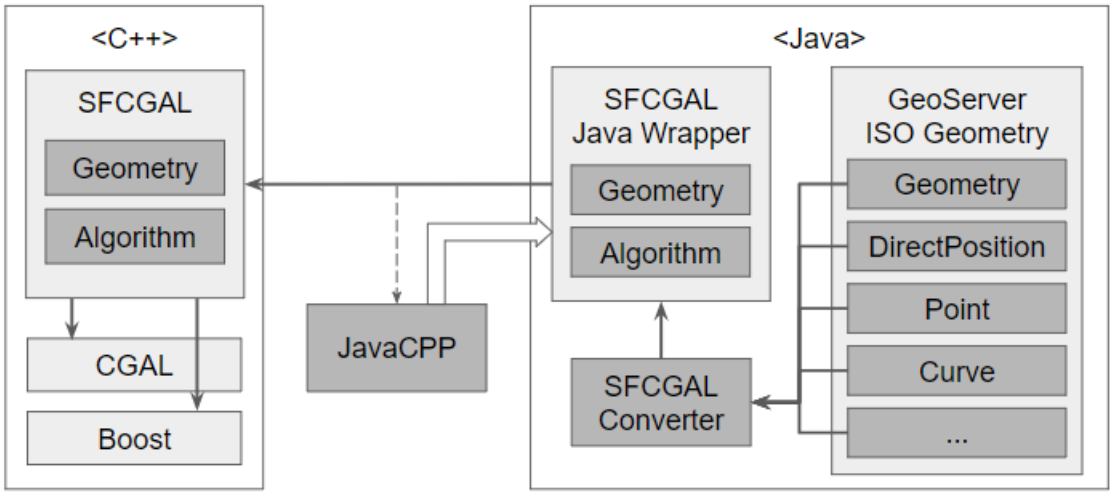
이 프로젝트에서 기하 모델은 ISO 19107 공간 스키마를 기반으로 하고 있습니다. GeoTools에서 이 기하 모델은 OpenGIS (gt-opengis) 모듈에서 자바 인터페이스로 정의되어 있습니다. 그리고 ISO Geometry (gt-geometry) 모듈에서 이를 구현하고 있지만 Solid를 제외한 2.5D 기하까지만 지원하고 있고 3차원 질의도 마찬가지로 지원하고 있지 않습니다. 이 때문에 이 구현을 그대로 사용하는 것은 불가능합니다.

우리는 3차원 공간 데이터와 질의를 제공하기 위해서 ISO 기하 모델에서 모든 기하에 대한 정의와 3차원 공간 연산들을 직접 구현하는 방법이 있지만 이는 엄청난 노력을 필요로 하고 탄탄하고 안정적인 기능을 제공하는 것이 어렵습니다. 이 때문에 우리는 다른 오픈소스 라이브러리가 공간 연산을 담당하도록 하였습니다.

우리가 3차원 기하 연산을 위해서 사용한 기하 라이브러리는 Simple Feature CGAL (SFCGAL)입니다. 이 라이브러리는 CGAL과 Boost를 기반으로 구현되어 있으며 ISO 19107 공간 스키마와 OGC Simple Feature Access 1.2 표준을 기반으로 기하 모델을 정의하고 있기 때문에 Solid 기하도 지원하고 있습니다. 또한 여러가지 3차원 기능들을 CGAL이 제공하는 기능들을 확장하여 구현하고 있습니다.

1.2.1 내부 구현

3차원 기하 연산을 지원하기 위해 SFCGAL 라이브러리를 ISO 19107 공간 스키마를 구현하는 클래스에 연결했습니다. 그 구조는 다음의 그림과 같습니다.



SFCGAL은 C++로 작성된 라이브러리이므로 SFCGAL 라이브러리의 함수를 호출하려면 네이티브 C++와 Java 사이의 인터페이스가 필요합니다. 이를 지원하는 라이브러리 중 우리는 JavaCPP를 사용하였습니다. JavaCPP는 오픈소스 도구이며 C++와 Java 사이를 쉽게 인터페이스 할 수 있습니다. JavaCPP를 이용하여 C++로 작성된 SFCGAL과 대응하는 Java 클래스를 생성하였고 이는 그림에서 SFCGAL Java Wrapper와 같습니다.

그리고 SFCGAL과 GeoTools 간의 연결을 위해서는 SFCGAL Java Wrapper 클래스들과 GeoTools ISO Geometry 간의 모델의 차이 때문에 일련의 변환 프로세스가 필요합니다. 이 기능은 SFCGAL Converter에서 담당하고 있으며 다음의 표를 변환 관계를 기준으로 변환 프로세스를 수행합니다.

Simple Feature Access	SFCGAL	ISO 19107
Coordinate	Coordinate	DirectPosition
Point	Point	Point
LineString	LineString	Curve
Line		
LinearRing		Ring
Polygon	Polygon	Surface
Triangle	Triangle	
PolyhedralSurface	PolyhedralSurface	PolyhedralSurface
	TriangulatedSurface	TriangulatedSurface
TIN	TIN	TIN
-	Solid	Solid
MultiPoint	MultiPoint	MultiPoint
MultiLineString	MultiLineString	MultiCurve
MultiPolygon	MultiPolygon	MultiSurface
-	MultiSolid	MultiSolid
GeometryCollection	GeometryCollection	MultiPrimitive

이제 SFCGAL Java Wrapper 클래스들은 GeoTools ISO Geometry 클래스로 부터 기하 연산이 호출될 때 SFGCAL의 해당 네이티브 메소드를 호출할 수 있습니다. SFCGAL에서 수행이 완료되면 결과는 SFCGAL Java Wrapper 클래스로 반환됩니다. 이는 다시 Java의 타입이나 GeoTools ISO Geometry로 변환되어 반환됩니다.

1.2.2 참조

- GeoTools ISO Geometry
- JavaCPP
- Boost
- CGAL - The Computational Geometry Algorithms Library
- Simple Feature CGAL

<<<<<<< HEAD .. datastore:

1.3 DataStore

TBD

1.4 XSD-GML

TBD

1.5 빌드

이 장에서는 이 프로젝트를 빌드하기 위한 방법을 소개합니다. 이 프로젝트가 사용하는 몇 가지 라이브러리가 c++ 언어 기반이기 때문에 이 프로젝트를 빌드하기 이전에 필요 라이브러리들을 우선적으로 설치하는 과정이 필요합니다.

1.5.1 필요 라이브러리

- CMAKE
- GMP
- MPFR
- Boost
- CGAL
- SFCGAL

1.5.2 Ubuntu 16.04에서 GeoTools 3D를 설치하기

먼저 다음의 명령어로 Git Repository로 부터 프로젝트를 받아옵니다.

$ git clone https://github.com/STEMLab/geotools-3d-extension.git

SFCGAL, CGAL, Boost 라이브러리에서 필요한 라이브러리들을 설치합니다. 이미 이 라이브러리들이 설치되어 있다면 다음의 과정은 생략해도 좋습니다.

$ sudo apt-get install -y cmake libgmp3-dev libmpfr-dev

gt-geometry-ng 모듈 내의 cppbuild.sh 파일을 실행하면 SFCGAL, CGAL, Boost 라이브러리가 시스템에 자동으로 설치됩니다. 시스템에 해당 라이브러리들을 설치하기 위하여 gt-geometry-ng 모듈의 경로로 이동한 후 sudo 권한 요청과 함께 cppbuild.sh를 다음과 같이 실행하십시오.

$ cd unsupported/geometry-ng

$ sudo ./cppbuild.sh

필요 라이브러리가 모두 제대로 설치되었는지 확인하기 위해 gt-geometry-ng 모듈을 다음과 같이 빌드합니다. 모든 빌드가 완료된 후 수행되는 모든 테스트가 성공하면 설치가 완료된 것입니다.

$ mvn clean install

이제 다음과 같이 루트 경로로 돌아가서 전체 프로젝트를 빌드하여 빌드가 성공함을 확인하십시오.

$ cd ../../

$ mvn clean install

1.6 빠르게 시작하는 메뉴얼

빠르게 시작하는 메뉴얼은 3차원 지리공간을 처음 접하는 자바 개발자를 대상으로 설명한다. 자바와 이클립스 설치 및 프로젝트 생성은 GeoTools 페이지를 참고하라 GeoTools Eclipse Quickstart.

1. 프로젝트를 생성한 후 pom.xml파일을 열어라.
2. GeoTools-3d-extention을 사용하기 위해서 pom.xml파일에 다음을 참고하여 dependency를 추가하라.

```
    <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
      <modelVersion>4.0.0</modelVersion>

        <artifactId>your project name</artifactId>

            <parent>
            <groupId>org.geotools</groupId>
            <artifactId>geotools-iso</artifactId>
            <version>15-SNAPSHOT</version>
            <relativePath>..</relativePath>
      </parent>

      <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
      </properties>

      <dependencies>
          <dependency>
          <groupId>org.geotools</groupId>
          <artifactId>gt-main-iso</artifactId>
          <version>${project.version}</version>
        </dependency>
        <dependency>
            <groupId>org.geotools</groupId>
                    <artifactId>gt-csv-iso</artifactId>
                    <version>${project.version}</version>
            </dependency>

        <!-- Provides support for PostGIS. Note the different groupId -->
            <dependency>
          <groupId>org.geotools.jdbc</groupId>
          <artifactId>gt-jdbc-postgis-iso</artifactId>
          <version>${project.version}</version>
        </dependency>

        <!-- Provides GUI components -->
        <dependency>
            <groupId>org.geotools</groupId>
            <artifactId>gt-swing</artifactId>
            <version>${project.version}</version>
        </dependency>
        <dependency>
            <groupId>org.geotools</groupId>
            <artifactId>gt-cql</artifactId>
            <version>${project.version}</version>
        </dependency>
```

(continues on next page)

(이전 페이지에서 계속)

```
        <dependency>
          <groupId>junit</groupId>
          <artifactId>junit</artifactId>
          <version>3.8.1</version>
          <scope>test</scope>
        </dependency>
      </dependencies>

      <build>
        <plugins>
            <plugin>
                      <groupId>org.apache.maven.plugins</groupId>
                      <artifactId>maven-surefire-plugin</artifactId>
                      <configuration>
                        <forkCount>3</forkCount>
                        <reuseForks>true</reuseForks>
                        <argLine>-Xmx1024m -XX:MaxPermSize=256m</argLine>
                      </configuration>
                    </plugin>
        </plugins>
    </build>
    </project>
```

3. 다음의 코드를 프로젝트에 추가하라.

```
    import java.awt.BorderLayout;
    import java.awt.Dimension;
    import java.awt.event.ActionEvent;
    import java.io.IOException;
    import java.util.ArrayList;
    import java.util.Map;
    import javax.swing.ComboBoxModel;
    import javax.swing.DefaultComboBoxModel;
    import javax.swing.JComboBox;
    import javax.swing.JFrame;
    import javax.swing.JMenu;
    import javax.swing.JMenuBar;
    import javax.swing.JOptionPane;
    import javax.swing.JScrollPane;
    import javax.swing.JTable;
    import javax.swing.JTextField;
    import javax.swing.table.DefaultTableModel;

    import org.geotools.data.DataStore;
    import org.geotools.data.DataStoreFactorySpi;
    import org.geotools.data.DataStoreFinder;
    import org.geotools.data.FeatureWriter;
    import org.geotools.data.ISODataUtilities;
    import org.geotools.data.Transaction;
    import org.geotools.data.csv.iso.CSVDataStoreFactory;
    import org.geotools.data.postgis3d.PostgisNGDataStoreFactory;
    import org.geotools.data.simple.SimpleFeatureCollection;
    import org.geotools.data.simple.SimpleFeatureSource;
    import org.geotools.factory.Hints;
    import org.geotools.feature.ISOFeatureFactoryImpl;
    import org.geotools.feature.simple.ISOSimpleFeatureTypeBuilder;
    import org.geotools.feature.simple.SimpleFeatureBuilder;
```

(continues on next page)

(이전 페이지에서 계속)

```
33      import org.geotools.filter.text.cql2.CQL;
34      import org.geotools.filter.text.cql2.CQLException;
35      import org.geotools.referencing.crs.DefaultGeographicCRS;
36      import org.geotools.swing.action.SafeAction;
37      import org.geotools.swing.data.JDataStoreWizard;
38      import org.geotools.swing.table.FeatureCollectionTableModel;
39      import org.geotools.swing.wizard.JWizard;
40
41      import org.opengis.feature.simple.SimpleFeature;
42      import org.opengis.feature.simple.SimpleFeatureType;
43      import org.opengis.filter.Filter;
44      import org.opengis.geometry.ISOGeometryBuilder;
45      import org.opengis.geometry.primitive.Solid;
46
47      public class App  extends JFrame{
48
49              private DataStore dataStore;
50
51              private JComboBox featureTypeCBox;
52
53              private JTable table;
54
55              private JTextField text;
56
57              private static ISOGeometryBuilder builder;
58
59              public static void main(String[] args) throws Exception {
60
61                      Hints h = new Hints();
62                      h.put(Hints.GEOMETRY_VALIDATE, false);
63                      h.put(Hints.CRS, DefaultGeographicCRS.WGS84_3D);
64                      builder = new ISOGeometryBuilder(h);
65
66                      JFrame frame = new App();
67                      frame.setVisible(true);
68
69              }
70
71              public App() {
72
73                      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
74                      getContentPane().setLayout(new BorderLayout());
75
76
77                      text = new JTextField(80);
78                      text.setText("include"); // include selects everything!
79                      getContentPane().add(text, BorderLayout.NORTH);
80
81
82                      table = new JTable();
83                      table.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);
84                      table.setModel(new DefaultTableModel(5, 5));
85                      table.setPreferredScrollableViewportSize(new Dimension(500,␣
    ↪200));
86
87
88                      JScrollPane scrollPane = new JScrollPane(table);
```

(continues on next page)

(이전 페이지에서 계속)

```
89                      getContentPane().add(scrollPane, BorderLayout.CENTER);
90
91
92                      JMenuBar menubar = new JMenuBar();
93                      setJMenuBar(menubar);
94
95
96                      JMenu fileMenu = new JMenu("File");
97                      menubar.add(fileMenu);
98
99
100                     featureTypeCBox = new JComboBox();
101                     menubar.add(featureTypeCBox);
102
103
104                     JMenu dataMenu = new JMenu("Data");
105                     menubar.add(dataMenu);
106
107                     pack();
108
109                     fileMenu.add(new SafeAction("Open csvfile...") {
110                             public void action(ActionEvent e) throws Throwable {
111                                     connect(new CSVDataStoreFactory());
112                             }
113                     });
114
115                     fileMenu.add(new SafeAction("Connect to PostGIS database...") {
116                             public void action(ActionEvent e) throws Throwable {
117                                     connect(new PostgisNGDataStoreFactory());
118                             }
119                     });
120
121                     fileMenu.add(new SafeAction("Insert Solid to PostGIS database...
    ↪") {
122                             public void action(ActionEvent e) throws Throwable {
123                                     insertTable();
124                             }
125                     });
126
127                     fileMenu.addSeparator();
128
129                     fileMenu.add(new SafeAction("Exit") {
130                             public void action(ActionEvent e) throws Throwable {
131                                     System.exit(0);
132                             }
133                     });
134
135                     dataMenu.add(new SafeAction("Get features") {
136                             public void action(ActionEvent e) throws Throwable {
137                                     filterFeatures();
138                             }
139                     });
140             }
141
142             private void connect(DataStoreFactorySpi format) {
143
144                     JDataStoreWizard wizard = new JDataStoreWizard(format);
```

(continues on next page)

(이전 페이지에서 계속)

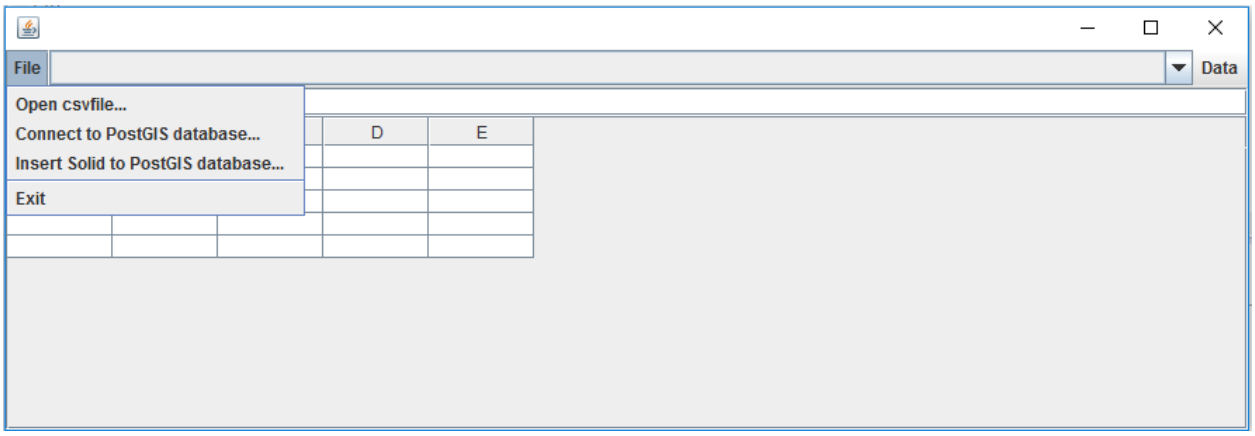
```
145                     int result = wizard.showModalDialog();
146
147                     if (result == JWizard.FINISH) {
148                             Map<String, Object> connectionParameters = wizard.
    ↪getConnectionParameters();
149
150                             try {
151                                     dataStore = DataStoreFinder.
    ↪getDataStore(connectionParameters);
152                                     if (dataStore == null) {
153                                             JOptionPane.showMessageDialog(null,
    ↪"Could not connect - check parameters");
154                                     }
155                                     updateUI();
156
157                             } catch (IOException e) {
158                                     // TODO Auto-generated catch block
159                                     e.printStackTrace();
160                             } catch (Exception e) {
161                                     // TODO Auto-generated catch block
162                                     e.printStackTrace();
163                             }
164
165                     }
166
167             }
168
169             private void insertTable() {
170                     String typeName = "oneSolid";
171                     ArrayList<Solid> al = ISODataUtilities.getSolids(builder);
172
173                     ISOSimpleFeatureTypeBuilder b = new␣
    ↪ISOSimpleFeatureTypeBuilder();
174                     b.setCRS(DefaultGeographicCRS.WGS84_3D);
175                     b.setName( typeName );
176                     b.add("loc", Solid.class);
177
178                     SimpleFeatureType schema = b.buildFeatureType();
179                     SimpleFeatureBuilder builder = new SimpleFeatureBuilder(schema,␣
    ↪new ISOFeatureFactoryImpl());
180                     builder.add( al.get(0) );
181                     SimpleFeature feature = builder.buildFeature( "fid.1" );
182
183                     try {
184                             dataStore.createSchema((SimpleFeatureType)␣
    ↪schema);
185                             FeatureWriter<SimpleFeatureType, SimpleFeature>␣
    ↪fw = dataStore.getFeatureWriterAppend(
186                                             schema.getTypeName(),␣
    ↪Transaction.AUTO_COMMIT);
187                             SimpleFeature newFeature = fw.next();
188                             newFeature.setAttributes(feature.
    ↪getAttributes());
189                             fw.write();
190                             fw.close();
191
192                     } catch (IOException e) {
```

(continues on next page)

(이전 페이지에서 계속)

```
193                             // TODO Auto-generated catch block
194                             System.out.println(e.getMessage());
195                             e.printStackTrace();
196                     } catch (Exception e) {
197                             // TODO Auto-generated catch block
198                             e.printStackTrace();
199                     }
200             }
201             private void updateUI()  {
202                     ComboBoxModel cbm;
203
204                     try {
205                             cbm = new DefaultComboBoxModel(dataStore.getTypeNames());
206                             featureTypeCBox.setModel(cbm);
207                     } catch (IOException e) {
208                             // TODO Auto-generated catch block
209                             e.printStackTrace();
210                     }
211                     table.setModel(new DefaultTableModel(5, 5));
212             }
213
214             private void filterFeatures()  {
215
216                     String typeName = (String) featureTypeCBox.getSelectedItem();
217                     SimpleFeatureSource source;
218
219                     try {
220                             source = dataStore.getFeatureSource(typeName);
221
222                             Filter filter = CQL.toFilter(text.getText());
223                             SimpleFeatureCollection features = source.
    ↪getFeatures(filter);
224
225                             FeatureCollectionTableModel model = new␣
    ↪FeatureCollectionTableModel(features);
226                             table.setModel(model);
227
228                     } catch (IOException | CQLException e) {
229                             // TODO Auto-generated catch block
230                             System.out.println(e.getMessage());
231                             e.printStackTrace();
232                     }
233             }
234     }
```

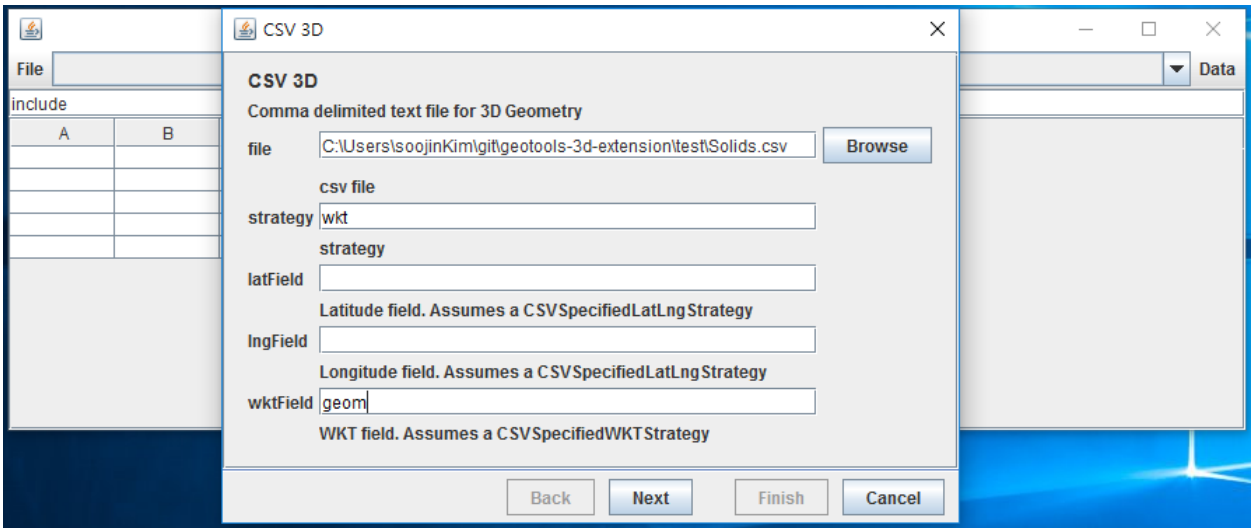
4. 위의 어플리케이션을 실행하면 당신은 CSV 파일을 열거나 PostGIS에 연결함으로써 3차원 DataStore을 만들 수 있다.



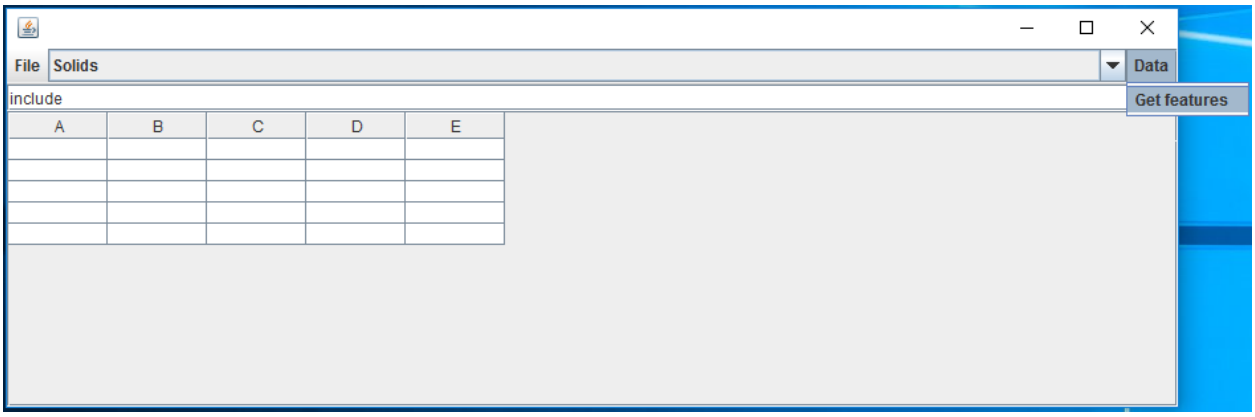
5-1. CSV 파일 DataStore을 만들어 보자. 우선 Open csv file을 클릭하고, 예제 데이터를 열어라.

6-1. CSV DataStore를 생성하는 데 필요한 설정은 다음과 같다. strategy는 기하가 파일에 어떻게 표현되어 있는지를 나타낸다. 만약 파일이 wkt(well known text)형식의 칼럼을 가지고 있어 기하를 해당 칼럼에 저장하고 있다면, strategy에 'wkt'라고 입력하라. 만약 파일이 WGS84좌표의 포인트를 두 개의 칼럼으로 각각 lattitude와 longitude를 저장하고 있다면, strategy에 'latlng'라고 입력하라. 만약 파일이 기하를 가지고 있지 않다면, 당신은 strategy를 입력하지 않아도 된다.

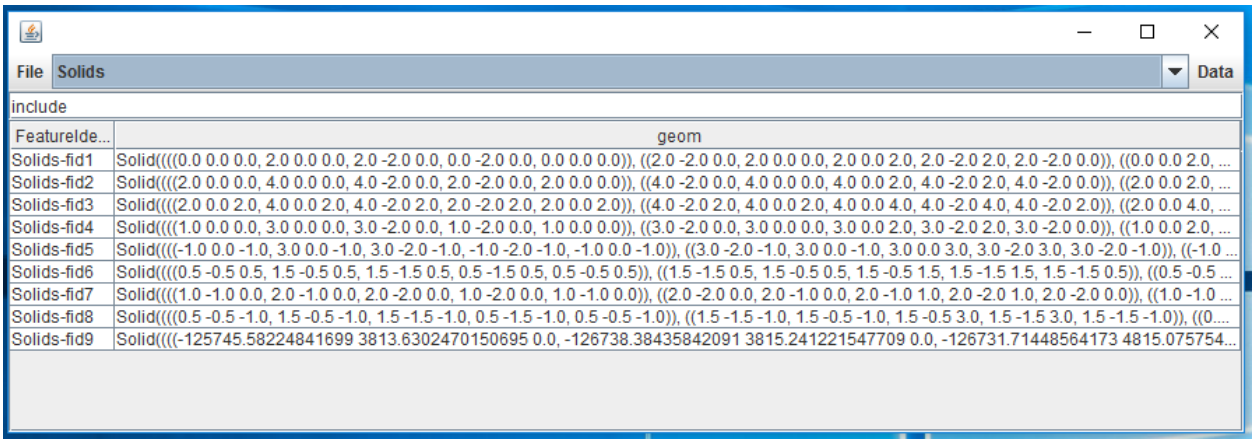
만약 당신이 wkt를 입력하였다면, wkt형식의 칼럼 이름을 wktField에 입력하라. 만약 당신이 latlng을 입력하였다면, lattitude, longitude 칼럼의 이름을 각각 latField, lngField에 입력하라.



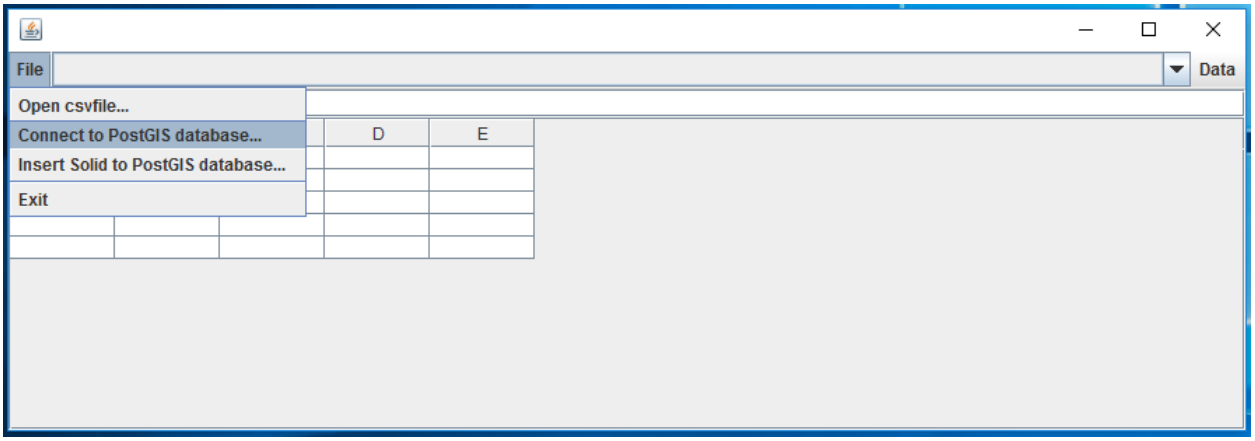
7-1. DataStore에 있는 데이터를 보고싶다면, getfeature 버튼을 눌러라.



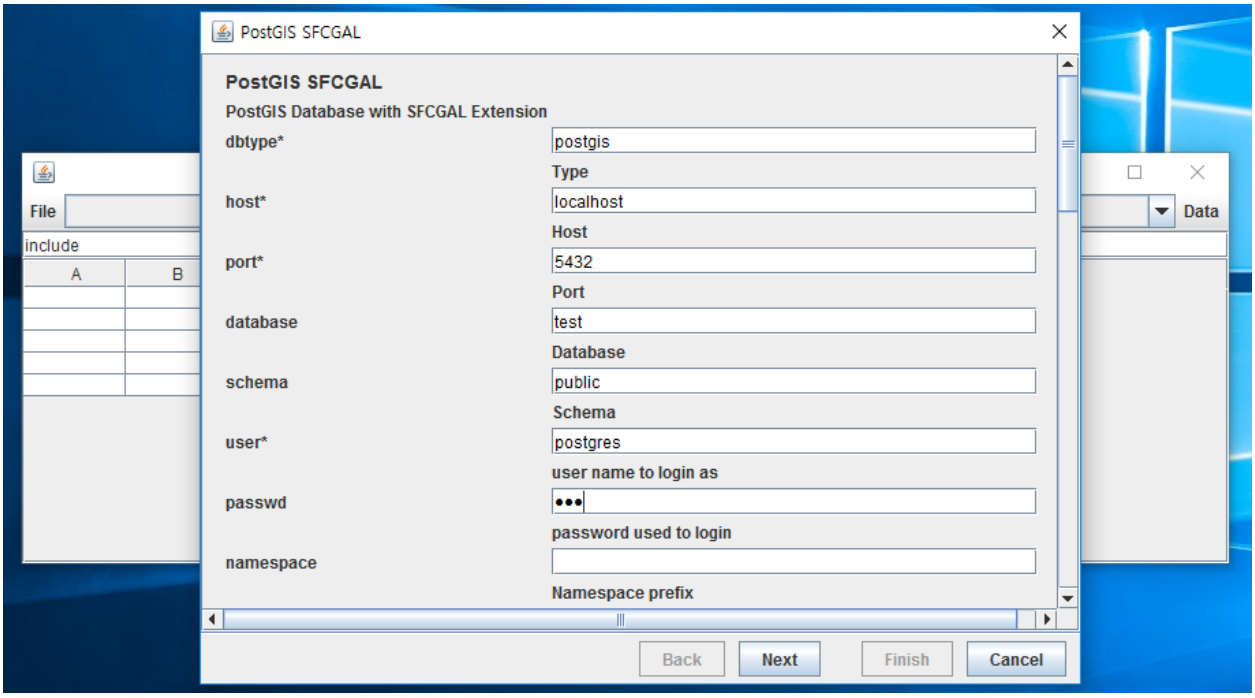
CSV DataStore로부터 getfeature함수를 적용한 결과는 다음과 같다.



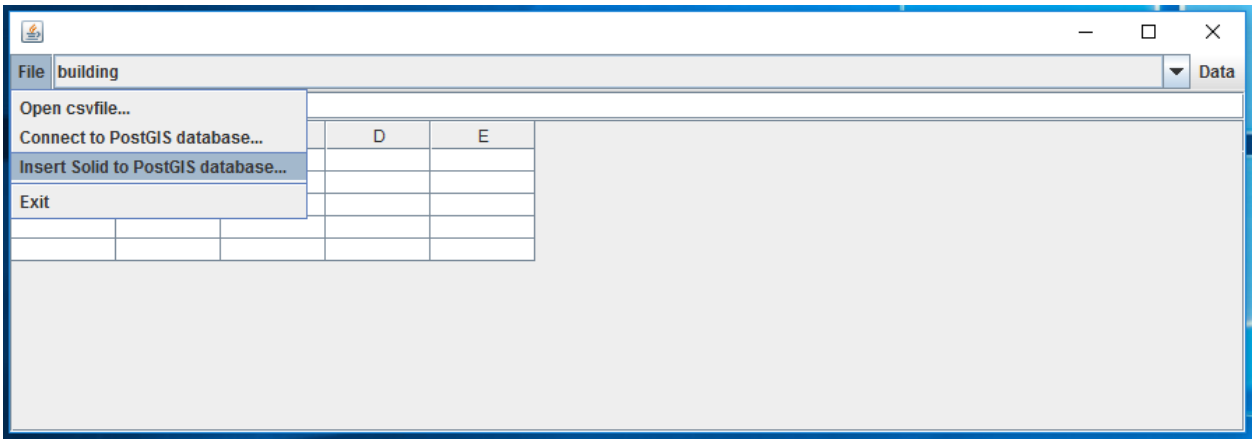
5-2 이번에는 PostGIS DataStore를 만들어 보자. 우선 Connect to PostGIS database를 클릭한다.



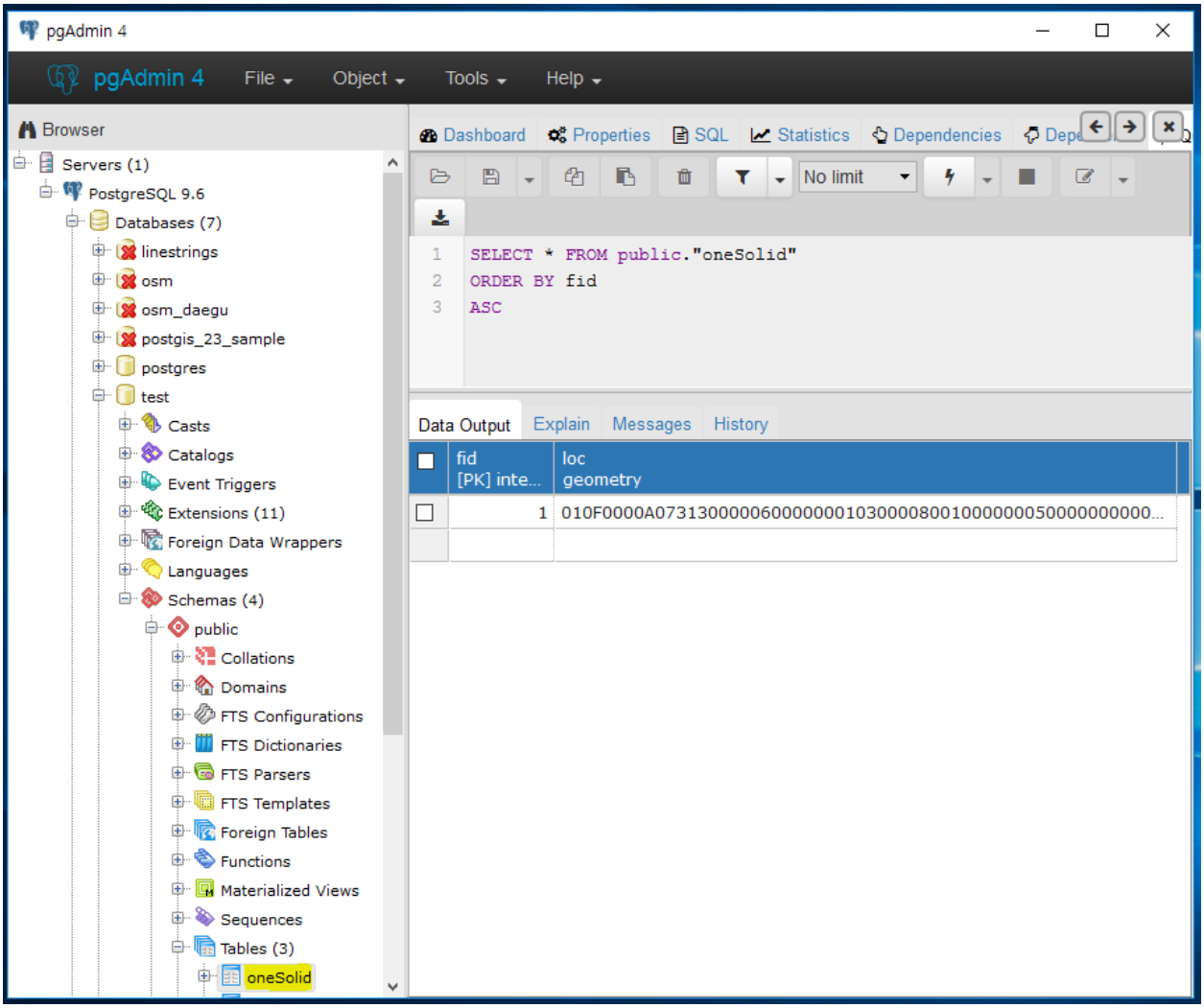
6-2 PostGIS DataStore를 생성하는 데 필요한 설정은 다음과 같다. 비밀번호 이후 설정은 옵션이므로 나머지는 입력하지 않고, finish버튼을 눌러도 무방하다.



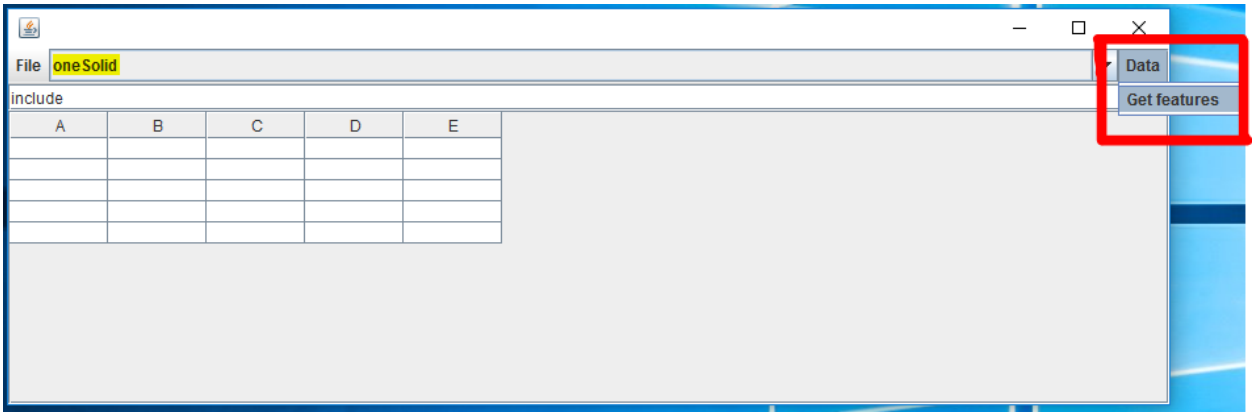
7-2 Insert Solid to PostGIS database버튼을 누르면 PostGIS 데이터베이스에 oneSolid라는 이름으로 id와 geometry를 칼럼으로 가지는 테이블이 하나 생기고, Solid 데이터 하나가 들어간다.



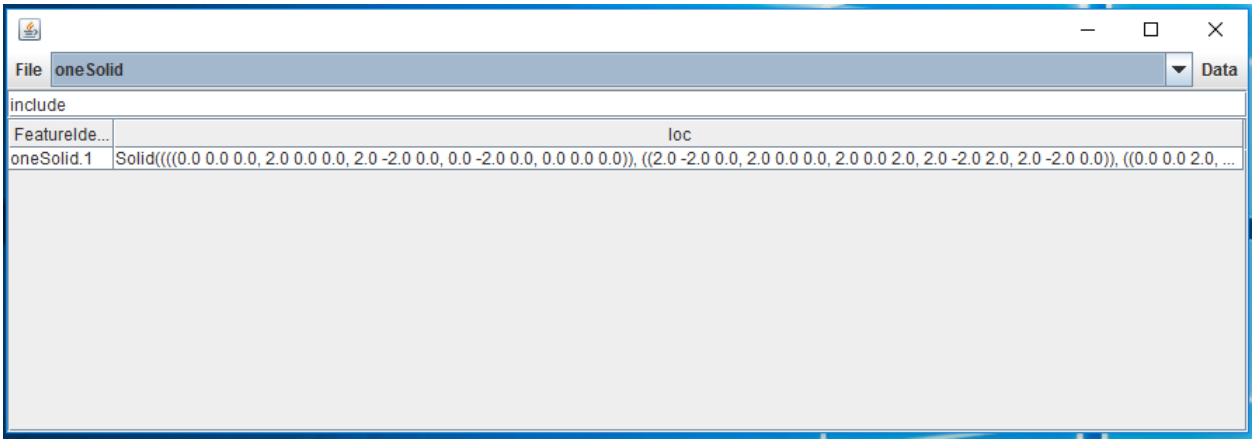
insert의 결과를 다음과 같이 확인할 수 있다.



dropdownlist에는 연결된 데이터베이스의 테이블들을 볼 수 있다. insert버튼을 누른 후에 새로 생성된 oneSolid테이블이 dropdownlist에 나타난 것을 확인할 수 있다. 해당 테이블의 데이터를 보기 위해서 dropdownlist에서 oneSolid 테이블이 선택된 채로 getfeature 버튼을 누른다.



PostGIS DataStore로부터 getfeature함수를 적용한 결과는 다음과 같다.


File
oneSolid
Data
include
FeatureIde...
loc
oneSolid.1
Solid((((0.0 0.0 0.0, 2.0 0.0 0.0, 2.0 -2.0 0.0, 0.0 -2.0 0.0, 0.0 0.0 0.0)), ((2.0 -2.0 0.0, 2.0 0.0 0.0, 2.0 0.0 2.0, 2.0 -2.0 2.0, 2.0 -2.0 0.0)), ((0.0 0.0 2.0, ...