Creating data containers

Content

1.Introducción

Vector data containers in gvSIG scripting extension are 2, the Table object, and the object Layer. Both objects contain records and the main difference between them about the data they contain is the record of a layer containng the field 'GEOMETRY'.

We have seen how to access objects and Layer Table via both functions currentTable () and currentLayer () and by the Project object by the method getTable and Object View by the method getLayer(), respectively. Now we will see how to create these objects.

2. El esquema (*Schema*)

When creating a data container, the first thing we do is to create a schema that represents the data. This scheme should at least have a Geometry type field if we want to create a layer (Layer).

CreateSchema function ()

There is a function that allows to create data definition quickly, either copying an existing definition or a whole new definition, which will add the properties we want.

The function syntax is:

```
createSchema([schema])
```

- createSchema (schema): Returns a new instance of a Schema.
    - schema, Schema (optional): If invoked passing as parameter an object Schema, it will try to make a copy of it and it will be return in edit mode, if it can't get back a new Schema return it will return an empty schema in edit mode.

Adding fields to a data definition. The append method

The *append* method of a data definition allows to add a new attribute to the data definition and define its properties. If the data definition is not in edit mode by invoking this method it will activate itselve.

```
schema.append(name, type, size, default, precision)
```

- name, string: Name of the property
- type, string: data type to contain. The most common values supported are:
    - "BOOLEAN"
    - "DOUBLE"
    - "FLOAT"
    - "INTEGER"
    - "LONG"
    - "STRING"
    - "GEOMETRY"

There are more types of data that can be used, check them in the documentation DataTypes.

- size, int (optional): Size of the property. Determines the maximum size of data type string. Also used in the representation of the data.
  The default size of the field that will be created is 1, so it is advisable to indicate the size of the field.
- default, object (optional): Is the default value of the property, in the case there is no other entries.
  The value by default in a numeric types is 0 (zero), and the strings is "" (empty string)
- Accuracy, int (optional): Sets the number of decimal data type, if this is numeric and has decimal support.

  The default value fro the attribute is 4, so if you need a higher precision is necessary to indicate it.

Examples

For example, we create a data definition that will be used later in a table dxf and we just want to have a field called "ID" of type LONG

```
schema = createSchema()
schema.append("ID", "LONG", 15)
```

We may also want to get a copy of the data structure of the active layer and add an additional property, for example a type string, with a size of 25 characters and a default value "No change".

```
schema = createSchema(currentLayer.getSchema())
schema.append("Campo", "STRING", size=25, default="Sin modificar")
```

The following code creates a Schema with 3 fields, "ID", "Name", "Geometry"

```
from gvsig import *
def main():
  schema = createSchema()
  schema.append("ID", "Integer", 5)
  schema.append("Nombre", "String", 50)
  schema.append("GEOMETRY", "Geometry")
```

3. Create a layer (shape)

To create a shape layer type, it is necessary to note that the schema has to be necessarily a field called "GEOMETRY" type "Geometry". If this field is not defined in the schema, it will fail.

You can create a shape using *createShape* function, which returns the newly created layer and whose syntax is:

```
createShape(schema, filename [, CRS] [,geometryType])
```

- schema, Schema: Definition of data.
- filename, string: Absolute path where we saved the new shapefile
- CRS, string (optional): code reference system. If not specified a default will be set as WGS 84.

- geometryType, GeometryType (optional): type shape geometry. If the type is not specified the default geometry is POINT. Other values are:
  - POINT
  - LINE
  - POLYGON
  - MULTIPOINT
  - MULTILINE
  - MULTIPOLYGON

  Continuing with our example, once we have our *schema* we will create a shape in the "/tmp /pruebaShape.shp" (you can change the path and use any other where you have writing permissions), using as CRS “EPSG: 23030 ".

  Note

  The following code creates only one layer without features

```
import tempfile
import os.path

from gvsig import *
from geom import *

def main():
  ruta = os.path.join(tempfile.gettempdir(),"pruebaShape.shp")

 schema = createSchema()
 schema.append("ID", "Integer", 5)
 schema.append("Name", "String", 50)
 schema.append("GEOMETRY", "Geometry")

  newShape = createShape(
      schema,
      shpFile=ruta,
      CRS="EPSG:23030",
      geometryType=POINT
  )
```

4. Creating a table (DBF)

Like creating a layer (shape), the first thing to do is to create a table (DBF) and establish the structure of data, using a Schema object.

The most notable difference when creating a table is not necessary that we have defined a field "GEOMETRY" in the Schema, so the code in order to create a dbf as in the above example would be

the following:

-
```
import tempfile
import os.path

from gvsig import *

def main():

  ruta = os.path.join(tempfile.gettempdir(), "pruebaDBF.dbf")

  schema = createSchema()
  schema.append("ID", "Integer", 5)
  schema.append("Name", "String", 50)

  newTable = createTable(
      schema,
      shpFile=ruta,
      CRS="EPSG:23030",
  )
```

Let's suppose we have a charged layer in the *View* and we want to create a table whose fields would be "ID" to be an account, another field "IDFeature" which will be the value of the "ID" of the input layer, and a final field "COMMENTS" with a size of 200 characters.

```
import tempfile
import os.path

from gvsig import *

def main():

  #To establish where we'll save the dbf
  ruta = os.path.join(tempfile.gettempdir(),"ejemploDBF.dbf")

  #to obtain the input layer
  layer = currentLayer()

  #TO obtain the code of the projection of the layer
  crs = layer.getProjectionCode()

  #To obtain data defining the input layer
  layerSchema = layer.getSchema()
```

```
#To get the attribute 'ID'
attr = layerSchema.get("ID")
```

```
#To obtain the data type and size of the attribute
tipoDeDato = attr.getDataTypeName()
 size = attr.getSize()

 #To create the data definition
 schema = createSchema()
 schema.append("ID", "Long", 10)
 schema.append("IDFeature", tipoDeDato, size)
 schema.append("COMENTARIOS", "String", 200)

 #Creating a table
 newTable = createTable(
     schema,
     shpFile=ruta,
     CRS=crs,
 )

 #TO get the phenomena of the input layer
 features = layer.features()

 #Initialize variables
 values = dict()

 for index, feature in enumerate(features):

   # TO obtain data on new records
   values["ID"] = index
   values["IDFeature"] = feature.get("ID")
   values["COMENTARIOS"] =""

   #Adding records to the new table
   newTable.append(values)

 #To finish editing the table and saving changes
 newTable.commit()
```

gvSIG Association

Plaza Don Juan de Villarrasa 14-5,

46001, Valencia (Spain)

Registro Nacional de Asociaciones (National Register of Associations): 596206

e-mail : info@gvsig.com

Web: www.gvsig.com

Project website: http://www.gvsig.org

Documentation made by Víctor Acevedo. Translated by Elisabet Adeva

Distribution Lists

There are three mailing lists in order to facilitate communication between all stakeholders in gvSIG project. The first two are for users and developers, mainly oriented to the Hispanic community, with the use of the Castilian lenguage preferred. The third list, is a international list, aimed at other communities where is preferd the use of English.

- **Members List.** . Here you can leave your opinions on the operation of the software: what things would you like to be developed, doubts related to the use of gvSIG and anything you believe should be in discussion and accommodates in the list of users. The link to subscribe to the user list is the following:
  http://listserv.gva.es/mailman/listinfo/gvsig_usuarios

- **Developers list.** It is geared/oriented to anyone interested in knowing how gvSIG is developed. The link to subscribe to this list is:
  http://listserv.gva.es/mailman/listinfo/gvsig_desarrolladores

- **International list.** It is intended for both users and developers witch do not speak Spanish. The preferably language used is English. The link to subscribe to this list is:
  http://listserv.gva.es/mailman/listinfo/gvsig_internacional