Creación de contenedores de datos

Índice de contenido

1 Introducción....3

2 El esquema (Schema)....3

Función createSchema()....3

Añadir campos a una definición de datos. El método append....3

Ejemplos....4

3 Crear una capa (shape)....4

4 Creación de una tabla (DBF)....5

1 Introducción

Los contenedores de datos vectoriales desde la extensión de scripting de gvSIG son 2, el objeto Table, y el objeto Layer. Ambos objetos contienen registros y la principal diferencia entre ellos respecto los datos que contienen es que un registro de una capa contiene un campo 'GEOMETRY'.

Hemos visto como acceder a los objetos Table y Layer tanto a través de las funciones currentTable() y currentLayer() y a través del objeto Project, mediante el método getTable, y del objeto Vista mediante el método getLayer(), respectivamente. Ahora veremos cómo crear estos objetos.

2 El esquema (*Schema*).

A la hora de crear un contenedor de datos, lo primero que deberemos hacer es crear un esquema que represente esos datos. Este esquema como mínimo deberá tener un campo de tipo *Geometry* si vamos a crear una capa (Layer).

Función createSchema()

Existe una función que nos permite crar una definición de datos de forma rápida, ya sea una copia de una definición existente o una definición completamente nueva a la que iremos añadiendo las propiedades que deseemos.

La sintaxis de la función es:

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

- createSchema(schema): Devuelve una nueva instancia de un Schema.
    - schema, Schema (opcional): Si se invoca pasándole como parámetro un objeto Schema, tratará de realizar una copia del mismo y la devolverá en modo edición, si no puede realizar la copia devolverá un Schema nuevo vacío en modo edición.

Añadir campos a una definición de datos. El método append

El método *append* de una definición de datos nos permite añadir un atributo nuevo a la definición de los datos y definir sus propiedades. Si la definición de datos no está en modo edición al invocar este método se activará.

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

- name, string: Nombre de la propiedad
- type, string: Tipo de dato que va a contener. Los valores admitidos más usuales son:
    - "BOOLEAN"
    - "DOUBLE"
    - "FLOAT"
    - "INTEGER"
    - "LONG"
    - "STRING"
    - "GEOMETRY"

Existen más tipos de datos que pueden usarse y puedes consultarlos en la documentación de los DataTypes

- size, int (opcional): Tamaño de la propiedad. Determina el tamaño máximo de los datos de tipo string. También se utiliza en la representación del dato.
  El tamaño por defecto del campo que se va a crear es 1 por lo que es recomendable indicar el tamaño del campo.
- default, object (opcional): Valor por defecto de la propiedad si no se indica ningún otro.
  El valor por defecto en tipos numéricos es el 0 (cero), y en cadenas es "" (cadena vacía)
- precision, int (opcional): Establece el número de decimales del tipo de datos, si este es numérico y admite decimales.

  El valor por defecto del atributo precision es 4, por lo que si necesitamos una precisión mayor será necesario indicarlo.

Ejemplos

Por ejemplo, queremos crear una definición de datos que usaremos luego en una tabla dxf y solo queremos que tenga un campo llamado "ID" de tipo LONG

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

También podemos queremos obtener una copia de la estructura de datos de la capa activa y añadirle una propiedad más, por ejemplo de tipo string, con un tamaño de 25 carácteres y con valor por defecto "Sin modificar".

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

El siguiente código crea un *Schema* con 3 campos, "ID", "Nombre", "Geometry"

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

3 Crear una capa (shape)

Para crear una capa de tipo shape, es necesario tener en cuenta que el schema tiene que tener obligatoriamente un campo llamado "GEOMETRY" de tipo "Geometry". En caso de que este campo no esté definido en el Schema, se producirá un error.

Puede crearse un shape mediante la función *createShape* , que devuelve la capa recien creada y cuya sintaxis es:

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

- schema, Schema: Definición de los datos.
- filename, string: Ruta absoluta donde debe guardarse el nuevo archivo shape
- CRS, string (opcional): Código del sistema de referencia. Si no se indica uno por defecto se

establecerá WGS 84.

- geometryType, GeometryType (opcional): Tipo de geometría del shape. Si no se indica el tipo de geometría el valor por defecto es POINT. Los otros valores admitidos son:
  - POINT
  - LINE
  - POLYGON
  - MULTIPOINT
  - MULTILINE
  - MULTIPOLYGON

Siguiendo con nuestro ejemplo, una vez que tenemos nuestro *schema* crearemos un shape en la carpeta "/tmp/pruebaShape.shp" (puedes cambiar la ruta y usar cualquier otra en la que tengas permisos de escritura), usando como CRS "EPSG:23030".

Note

El siguiente código crea únicamente una capa sin 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("Nombre", "String", 50)
  schema.append("GEOMETRY", "Geometry")

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

4 Creación de una tabla (DBF)

Al igual que para crear una capa (shape), lo primero que debemos hacer para crear una tabla (DBF), es establecer su estructura de datos, mediante un objeto Schema.

La diferencia más notable a la hora de crear una tabla es que no es necesario que haya definido un campo "GEOMETRY" en el Schema, por lo que el código para crear un dbf igual que en el ejemplo anterior sería:

```
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("Nombre", "String", 50)

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

Supongamos que tenemos una capa cargada en la *Vista* y queremos crear una tabla, cuyos campos serían "ID" que será un contador, otro campo "IDFeature" que será el valor del campo "ID" de la capa de entrada, y un último campo "COMENTARIOS" con un tamaño de 200 caracteres.

```
import tempfile
import os.path

from gvsig import *

def main():

  #Establecemos donde vamos a guardar el dbf
  ruta = os.path.join(tempfile.gettempdir(),"ejemploDBF.dbf")

  #Obtenemos la capa de entrada
  layer = currentLayer()

  #Obtenemos el codigo de la proyeccion de la capa
  crs = layer.getProjectionCode()

  #Obtenemos la definición de datos de la capa de entrada
  layerSchema = layer.getSchema()

  #Obtenemos el atributo 'ID'
  attr = layerSchema.get("ID")
```

```
#Obtenemos el tipo de dato y el tamaño del atributo
tipoDeDato = attr.getDataTypeName()
size = attr.getSize()

#Creamos la definición de datos
schema = createSchema()
schema.append("ID", "Long", 10)
schema.append("IDFeature", tipoDeDato, size)
schema.append("COMENTARIOS", "String", 200)

#Creamos la tabla
newTable = createTable(
    schema,
    shpFile=ruta,
    CRS=crs,
)

#Obtenemos los fenómenos de la capa de entrada
features = layer.features()

#Inicializamos las variables
values = dict()

for index, feature in enumerate(features):

  # Obtenemos los datos de los nuevos registros
  values["ID"] = index
  values["IDFeature"] = feature.get("ID")
  values["COMENTARIOS"] =""

  #Añadimos los registros a la tabla nueva
  newTable.append(values)

#Terminamos la edición de la tabla guardando los cambios
newTable.commit()
```

gvSIG Association

Plaza Don Juan de Villarrasa 14-5,

46001, Valencia (Spain)

Registro Nacional de Asociaciones: 596206

e-mail : info@gvsig.com

Web: www.gvsig.com

Web del proyecto: http://www.gvsig.org

Documentación realizada por Víctor Acevedo.

Listas de Distribución

Existen tres listas de distribución con el objeto de facilitar la comunicación entre todos los interesados en el proyecto gvSIG. Las dos primeras, la de usuarios y la de desarrolladores, están principalmente orientadas a la comunidad de habla hispana, siendo el castellano el idioma preferente a utilizar en las mismas. La tercera de ellas, lista internacional, está orientada principalmente al resto de comunidades y la lengua preferente a utilizar es la inglesa.

- **Lista de usuarios**. Aquí podéis hacer llegar vuestra opinión sobre el funcionamiento: qué cosas os gustaría que se desarrollaran, dudas en el uso de gvSIG y todo aquello que penséis que tiene cabida en una lista de usuarios. El enlace para la suscripción a la lista de usuarios es:
  http://listserv.gva.es/mailman/listinfo/gvsig_usuarios

- **Lista de desarrolladores**. Está orientada para todos los interesados en conocer cómo está desarrollado el gvSIG. El enlace para la suscripción a esta lista esa:
  http://listserv.gva.es/mailman/listinfo/gvsig_desarrolladores

- **Lista internacional**. Está orientada tanto para usuarios como para desarrolladores de habla no hispana. El idioma a utilizar es preferentemente inglés. El enlace para la suscripción a esta lista es:
  http://listserv.gva.es/mailman/listinfo/gvsig_internacional