Contenedores de datos. Clases Table y Layer .

Índice de contenido

1 Los "contenedores" de datos. Objetos Layer y Table

Existen 2 objetos que tienen como característica común que almacenan los datos, pese a que tienen grandes diferencias. Estos objetos son, la *Tabla* (se corresponde con el objeto java TableDocument) y la *Capa* (se corresponde con el objeto java FLayer). Hay que resaltar que pese a que comparten los métodos de gestión de los datos son objetos diferentes, por lo que el resto de propiedades no es el mismo. Veámoslo con un ejemplo. Cargamos un documento Tabla desde el gestor de proyectos de gvSIG, que se corresponda con un archivo dbf de un shape. A continuación abrimos ese shape en una vista, y abrimos su tabla de atributos. En el gestor de proyectos si comprobamos los documentos tabla, aparecen 2 documentos, cuyos nombres no coinciden y cuyos atributos tampoco (el documento que hemos abierto como un documento Tabla no tiene campo GEOMETRY), pese a que son el mismo archivo dbf.

Abriendo cada uno de los elementos desde la consola de Jython podemos ver que son diferentes

```
Jython Completion Shell
Jython 2.5.2 (Release_2_5_2:7206, Mar 2 2011, 23:12:06)
[Java HotSpot(TM) Server VM (Sun Microsystems Inc.)] on java1.6.0_26
>>> from gvsig import *
>>> currentTable().getName()
u'areaDeInfluencia'
>>> currentTable().getSchema().getAttrNames()
[u'Id_muni', u'Influencia']
>>> currentTable().getName()
u'Tabla de atributos: areaDeInfluencia.shp'
>>> currentTable().getSchema().getAttrNames()
[u'Id_muni', u'Influencia', u'GEOMETRY']
>>>
```

Como hemos dicho, tanto la *Capa* como la *Tabla* contienen un conjunto de datos o registros, la diferencia, respecto sus datos, es que en la capa existe el campo "geometry", que es la definición del elemento cartográfico, y la tabla no, por lo que los métodos de gestión de sus colecciones de datos son los mismos.

Estos métodos son:

```
- features([expresion][, sortBy][, asc])
- edit()
- append( values )
- updateSchema( schema )
- getSchema()
- commit()
- abort()
```

```
- getSelection()
```

- features([expresion][, sortBy][, asc]): Devuelve la colección de datos.
    - expresion, string (opcional): Condición que tiene que cumplir el fenómeno para ser incluido en la colección que se devuelve.
    - sortBy, string (opcional): Identificador del campo que se va a utilizar para que se ordene la colección de fenómenos que se devuelve.
    - asc, boolean (opcional): True si se ordenan en orden ascendente, False en caso contrario. Por defecto se usa True.

  El parámetro expresión de tipo string, define un filtro que se evalúa para determinar si un fenómeno debe ser incluido en la colección que se devuelve. Por ejemplo:

```
expresion = "ID > 10 AND ID < 20"
features = currentLayer().features(expresion, 'ID', True)
```

  Este código devuelve la colección de fenómenos de la capa activa cuyo campo *ID* es mayor que 10 y menor que 20, ordenado por el campo *ID* ascendente. Si se utiliza una expresión para filtrar los fenómenos y esta expresión no es válida, devuelve None. Si la expresión es válida pero no produce resultados devuelve un conjunto de 0 elementos.

  Si únicamente queremos ordenar la colección por un campo en sentido descendente el código sería:

```
features = currentLayer().features(sortBy='ID', asc=True)
```

- edit(): Activa el modo edición del objeto. Es necesario que el objeto esté en edición tanto para actualizar como para añadir objetos nuevos a la colección de datos.

  Una vez que no se necesite hacer cambios será necesario invocar a los métodos commit si se quieren persistir los cambios o abort si no se quieren guardar.
- append(values): Crea un nuevo fenómeno y lo añade a la colección de datos.
    - values, dict: Añade en la propiedad *key* del fenómeno, el valor correspondiente.

  Si el objeto no está en estado de edición al usar este método se cambiará el estado a modo edición.
- updateSchema(schema): Actualiza el modelo de datos del objeto con el nuevo modelo que se pasa como parámetro. El objeto debe estar en modo edición.
    - schema, Schema: Modelo de datos que usará la tabla o capa
- getSchema(): Devuelve el modelo de datos del objeto.
- commit(): Termina el modo edición manteniendo los cambios
- abort(): Termina el modo edición descartando los cambios.
- getSelection(): Devuelve el subconjunto de fenómenos que estén seleccionados en el objeto del total de elementos de la colección. Si no hay elementos seleccionados devolverá un objeto *Selection* con 0 fenómenos
- select(selection): Añade un fenómeno o un conjunto de ellos a la selección.

    - selection, Feature, FeatureSet: El parámetro selection puede ser un fenómeno (Feature) o un conjunto de fenómenos (FeatureSet)
- deselect(selection): Elimina un fenómeno o un conjunto de ellos a la selección.
    - selection, Feature, FeatureSet: El parámetro selection puede ser un fenómeno (Feature) o un conjunto de fenómenos (FeatureSet)
- isSelected(feature): Devuelve True si el fenómeno está seleccionado.
    - feature, Feature: Fenómeno del que queremos saber si está seleccionado.

Otros métodos comunes son:

```
- getProjectionCode()
- getName()
```

- getProjectionCode(): Devuelve un string con el código de la proyección de la capa/tabla
- getName(): Devuelve el nombre de la capa en la Tabla de Contenidos de la vista (TOC en sus siglas en inglés, Table Of Contents)

Métodos exclusivos del objeto Layer

```
- getTypeVectorLayer()
```

- getTypeVectorLayer(): Devuelve el tipo de geometría de la capa.

Métodos exclusivos del objeto Table

```
- getAssociatedLayer()
```

- getAssociatedLayer(): Devuelve el objeto Layer asociado a la tabla de atributos o None si no existe.

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