Phenomena of a tabular data sourcing

Content

1. Access to the phenomena of a tabular data source

To access the events from a tabular data source method is used *features*. This method returns an object that allows us to iterate over the data set layer. However, we must bear in mind that reuses the iterator instance for the new value, so that during an iteration we must guard the value of the instance, if you obtain a copy of this by the method *getCopy* the phenomenon. The reason for reusing the instance is that this technique increases the execution speed and makes far fewer instances of objects, which improves the performance of the Java virtual machine.

2. Collection of phenomena. The FeatureSet class

The code for the phenomena of the active layer would be:

```
layer = currentLayer()
features = layer.features()
```

The most important methods which have the object are featureSet

```
- getCount()
- update(feature)
```

- getCount (): returns the number of events that has the data source
- update (feature): updates a phenomenon within the collection.

Let's see how to traverse a layer phenomena with an example. Suppose we have an active layer whose phenomena have a property *ELEVATION*, we want to get the *feature* that corresponds to the maximum value of this attribute.

The steps followed in the example are (*example 1*):

- Obtain the active layer
- Get the set iterator layer phenomena
- Initialize variables
- Go trought the set of phenomena by iteration
- Check the ELEVATION attribute value
- If it is greater than the one we have stored keep the phenomenon

```
layer = currentLayer()
features = layer.features()
fmax = 0.0
newFeature = None
for feature in features:
  if feature.ELEVACION > fmax:
    newFeature = feature.getCopy()
    #
    #more code
```

Remember to make a copy of the feature and not a direct marked.

3. Management selection. Class Selection

We can get a specific subset of phenomenas in a layer or table using a filter on the attributes of the phenomena. To do this we need to invoke the method 'features' of a layer using the condition we want to apply to all data to obtain our subset.

For example, the code to apply a filter to our data set could be (*example 2*):

```
layer = currentLayer()
expresion = "ID >= 50 AND ID < 100"
features = layer.features(expresion)
#
#more code
```

It may also happen that we want to operate a data set that we have selected in the layer. In this case, we use the method GetSelection layer (*Example 3*).

```
layer = currentLayer()
features = layer.getSelection()
#
#other code
```

It is possible, too, that we want to explore the phenomena and to add some to the selection. In this case, the first thing to do is to get the object 'selection' and add the phenomena we want using the method 'select' (*Example 4*).

```
layer = currentLayer()
features = layer.features()
for feature in features:
  featureCopy = feature.getCopy()
  layer.getSelection().select(featureCopy)
#
#more code
```

The above example would select all phenomena of the layer, which is equivalent to the method 'selectAll' (*Example 5*).

```
layer = currentLayer()
features = layer.getSelection().selectAll()
```

To select the events that meet a certain condition can do it this way. This example adds a selection phenomenon that has the highest value in the field ELEVATION (*Example 6*).

```
layer = currentLayer()
features = layer.features()
fmax =0
featureCopy = None
for feature in features:
  if feature.ELEVACION > fmax:
```

```
    featureCopy = feature.getCopy()
if featureCopy:
  layer.getSelection().select(featureCopy)
#
#more code
```

4. Feature object

Represents a phenomenon of tabular data source. Its main methods are:

```
- geometry()
- getValues()
- edit()
- set(nombre, valor)
```

- geometry (): Returns the default geometry of the phenomenon.
- getValues (): Returns a dictionary with the name and value of the properties of the phenomenon
- edit (): puts the phenomenon in edit mode.
- set (name, value): Set in the property set by parameter name passed as a parameter value.
    - name string: Name of the property of the phenomenon
    - value object: the Value that must be assigned to the property

5. Modification of the attributes of a phenomenon

Phenomena are not themselves editable. To edit a phenomenon is necessary to invoke the method *edit (),* following it we make any changes we want to be apply on the phenomenon.

Internally editing a feature is done using a copy of this to avoid ambiguities in the internal state of the phenomenon.

The code to modify an attribute of the phenomenon is *(Example 7)*

```
fuente = currentTable()
features = fuente.features()
for feature in features:
  feature.edit()
  feature.set("NombreDelAtributo", valor)
  fuente.update(feature)


fuente.commit()
```

The code to modify a layer is the same as calling the function currentLayer.

6. Add a record to the data source

To add a record or new phenomenon tabular data source we must use the *append* method of the instance of the data source.

```
table = currentTable()
table.append(values)
table.update()
table.commit()
```

- append (values): Creates a new phenomenon and adds it to the collection of data.
  - Values, dict: Add the key property of the phenomenon, the corresponding value.

  If the object is not in the edit mode, it will change the state to edit mode by it selves.

  For example, suppose we have a dxf with no records in the field "Id", of integer type, and we want to add 10 new records whose id is increased by one unit (*Example 8*).

```
table = currentTable()
values = dict()
for i in range(1,11):
  values["Id"] = i
  table.append(values)
table.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