```
# encoding: utf-8

import gvsig
from gvsig import currentLayer
from gvsig import currentView
from gvsig import createFeatureType
from gvsig import createShape
from gvsig import getResource

from gvsig.geom import POLYGON
from gvsig.geom import POINT, MULTIPOINT
from gvsig.geom import D2
from gvsig.geom import createPoint

from gvsig.commonsdialog import msgbox

from gvsig.libs.formpanel import FormPanel


def agregarID():
  layer = currentLayer()
  store = layer.getFeatureStore()
  store.edit()

  ft = store.getDefaultFeatureType().getEditable()
  ft.append("ID","INTEGER", 10)
  store.update(ft)

  features = store.getFeatureSet()
  n = 1
  for f in features:
    f = f.getEditable()
    f.set("ID",n)
    features.update(f)
    n += 1
  store.finishEditing()



def listadoDePuntos():
  layer = currentLayer()
  print layer.getName()
  features = layer.features()
  for f in features:
    print f.get("ID"), f.getDefaultGeometry().getX(), f.getDefaultGeometry().getY()

def crearShapeConBuffer():
  schema = createFeatureType()

  schema.append("GEOMETRY", "GEOMETRY")
  schema.get("GEOMETRY").setGeometryType(POLYGON, D2)
  schema.append("ID","INTEGER",10)
  output = createShape(schema,prefixname="buffer")

  layer = currentLayer()
  print layer.getName()
  #Iniciamos edicion en la capa de salida
  outputstore = output.getFeatureStore()
  outputstore.edit()
  for feature in layer.features():
    geom = feature.getDefaultGeometry()
    # Para cada featura de entrada creamos una de salida
    # y copiamos los datos de origen en ella
    newfeature = outputstore.createNewFeature()
    newfeature.set("ID",feature.get("ID"))
    # A la geoemtria le aplicamos un buffer
    newfeature.setDefaultGeometry(geom.buffer(50))
    # y insertamos la nueva feature en la capa de salida
    outputstore.insert(newfeature)
```

```
  # Al terminar de procesar las features, terminamos la edicion
  # en la capa de salida para consolidar los datos.
  outputstore.finishEditing()

  currentView().addLayer(output)


def crearShapeConBufferYDistancia():
  schema = createFeatureType()

  schema.append("GEOMETRY", "GEOMETRY")
  schema.get("GEOMETRY").setGeometryType(POLYGON, D2)

  #+1
  schema.append("DISTANCIA","DOUBLE",10)

  schema.append("ID","INTEGER",10)
  output = createShape(schema,prefixname="buffer")

  #+3
  refpoint = createPoint(D2)
  refpoint.setX(725000)
  refpoint.setY(4370000)

  layer = currentLayer()
  print layer.getName()
  #Iniciamos edicion en la capa de salida
  outputstore = output.getFeatureStore()
  outputstore.edit()
  for feature in layer.features():
    geom = feature.getDefaultGeometry()
    # Para cada featura de entrada creamos una de salida
    # y copiamos los datos de origen en ella
    newfeature = outputstore.createNewFeature()
    newfeature.set("ID",feature.get("ID"))

    #+1
    newfeature.set("DISTANCIA",geom.distance(refpoint))

    # A la geoemtria le aplicamos un buffer
    newfeature.setDefaultGeometry(geom.buffer(50))
    # y insertamos la nueva feature en la capa de salida
    outputstore.insert(newfeature)

  # Al terminar de procesar las features, terminamos la edicion
  # en la capa de salida para consolidar los datos.
  outputstore.finishEditing()

  currentView().addLayer(output)
  # Aplicar una leyenda de intervalos para ver el resultado
  # de forma grafica

def crearShapeConBufferYDistanciaFiltrando():
  schema = createFeatureType()

  schema.append("GEOMETRY", "GEOMETRY")
  schema.get("GEOMETRY").setGeometryType(POLYGON, D2)
  schema.append("DISTANCIA","DOUBLE",10)
  schema.append("ID","INTEGER",10)
  output = createShape(schema,prefixname="buffer")

  refpoint = createPoint(D2)
  refpoint.setX(725000)
  refpoint.setY(4370000)

  #+2
  # Obtenemos una referencia a la capa de manzanas
  manzanas = currentView().getLayer("manzanas_pob")

  layer = currentLayer()
```

```
  print layer.getName()
  features = layer.features()
  #Iniciamos edicion en la capa de salida
  outputstore = output.getFeatureStore()
  outputstore.edit()
  for feature in features.iterator():
    geom = feature.getDefaultGeometry()
    #+3 y sangrar
    for manzana in manzanas.features():
      geom_manzana = manzana.getDefaultGeometry()
      if geom.intersects(geom_manzana):
        # Para cada featura de entrada creamos una de salida
        newfeature = outputstore.createNewFeature()
        newfeature.set("ID",feature.get("ID"))
        newfeature.set("DISTANCIA",geom.distance(refpoint))
        # A la geoemtria le aplicamos un buffer
        newfeature.setDefaultGeometry(geom.buffer(50))
        # y insertamos la nueva feature en la capa de salida
        outputstore.insert(newfeature)

  # Al terminar de procesar las features, terminamos la edicion
  # en la capa de salida para consolidar los datos.
  outputstore.finishEditing()

  currentView().addLayer(output)
  # Aplicar una leyenda de intervalos para ver el resultado
  # de forma grafica

def crearShapeDesplazado():
  schema = createFeatureType()

  schema.append("GEOMETRY", "GEOMETRY")
  schema.get("GEOMETRY").setGeometryType(POINT, D2)
  schema.append("ID","INTEGER",10)
  output = createShape(schema, prefixname="shift")

  layer = currentLayer()
  print layer.getName()

  outputstore = output.getFeatureStore()
  outputstore.edit()
  for feature in layer.features():
    geom = feature.getDefaultGeometry()

    newfeature = outputstore.createNewFeature()
    newfeature.set("ID",feature.get("ID"))
    point = createPoint(D2)
    point.setX(geom.getX()+100)
    point.setY(geom.getY()+100)
    newfeature.setDefaultGeometry(point)

    outputstore.insert(newfeature)

  outputstore.finishEditing()

  currentView().addLayer(output)

def crearShapeDesplazado2(layer, despX, despY):
  if layer == None:
    msgbox("Debera indicar la capa origen")
    return

  schema = createFeatureType()

  schema.append("GEOMETRY", "GEOMETRY")
  schema.get("GEOMETRY").setGeometryType(POINT, D2)
  schema.append("ID","INTEGER",10)
  output = createShape(schema, prefixname="shift")

  outputstore = output.getFeatureStore()
```

```
  outputstore.edit()
  id=1
  for feature in layer.features():
    geom = feature.getDefaultGeometry()

    newfeature = outputstore.createNewFeature()
    newfeature.set("ID",id)
    point = createPoint(D2)
    point.setX(geom.centroid().getX()+despX)
    point.setY(geom.centroid().getY()+despY)
    newfeature.setDefaultGeometry(point)
    id+=1
    outputstore.insert(newfeature)

  outputstore.finishEditing()
  currentView().addLayer(output)

from org.gvsig.tools.swing.api import ToolsSwingLocator
from org.gvsig.app import ApplicationLocator

def setLayersComboBoxModel(combo):
  layersModel = ApplicationLocator.getManager().createProjectLayersTreeModel()
  ToolsSwingLocator.getToolsSwingManager().setTreeModel(combo,layersModel)

class DesplazarPanel(FormPanel):
  def __init__(self):
    FormPanel.__init__(self,getResource(__file__,"desplazarPanel.xml"))
    setLayersComboBoxModel(self.cboLayers)
    self.setPreferredSize(300,150)

  def btnCancelar_click(self,*args):
    self.hide()

  def btnDesplazar_click(self,*args):
    try:
      despX = int(self.txtDesplazamientoX.getText())
      despY = int(self.txtDesplazamientoY.getText())
    except:
      msgbox("Valores incorrectos para desplazamiento X o y")
      return
    layer = self.cboLayers.getSelectedItem()
    if layer == None:
      msgbox("Debera seleccionar una capa")
      return
    msgbox("Desplazar la capa %r, desplazamiento X %s, desplazamiento y %s" % (
        layer.getName(),
        despX,
        despY
      )
    )
    crearShapeDesplazado2(layer, despX, despY)

def main(*args):
  #listadoDePuns()
  #agregarID()
  #crearShapeConBuffer()
  #crearShapeConBufferYDistancia()
  #crearShapeConBufferYDistanciaFiltrando()
  #crearShapeDesplazado()
  #crearShapeDesplazado2(currentView().getLayer("manzanas_puntos"), 200, 200)
  panel = DesplazarPanel()
  panel.showWindow("Desplazar")
```