```
execfile( script.getResource("../libs/import_utils.py").getAbsolutePath() )

import os
import os.path

from gvsig import *
from commonsdialog import *
from geom import *

from java.io import File
from java.lang import Thread
from java.awt import Dimension

from org.gvsig.tools import ToolsLocator

from org.gvsig.fmap.geom import GeometryLocator
from org.gvsig.fmap.geom import Geometry

from org.gvsig.fmap.dal import DALLocator
from org.gvsig.fmap.dal import DataTypes


import_from_module("imagegpsmetadata","ImageGPSMetadata")
import_from_module("..libs.relpath","relpath")
import_from_module("..libs.formpanel","ProgressBarWithTaskStatus,FormPanel")


class ProcessFolder(Thread):
  def __init__(self):
    self.__inputFolder = None
    self.__outputFile = None
    self.__addLayerToView = False
    self.__projection = "EPSG:4326"
    self.__onFinish = None


  def setOnFinish(self,onFinish):
    self.__onFinish = onFinish

  def setInputFoldername(self, inputFoldername):
    self.__inputFolder = File(inputFoldername)

  def setOutputFilename(self, outputFilename):
    self.__outputFile = File(outputFilename)

  def setAddLayerToView(self, add):
    self.__addLayerToView = add

  def createFeatureType(self):
    ft = DALLocator.getDataManager().createFeatureType()
    ft.add("id",DataTypes.INT)
    ft.add("text",DataTypes.STRING,100).setAllowNull(True)
    ft.add("fname",DataTypes.STRING,100).setAllowNull(True)
    ft.add("relpath",DataTypes.STRING,200).setAllowNull(True)
    ft.add("abspath",DataTypes.STRING,200).setAllowNull(True)
    ft.add("altitude",DataTypes.STRING,50).setAllowNull(True)
    ft.add("altitudere",DataTypes.STRING,50).setAllowNull(True)
    ft.add("datum",DataTypes.STRING,30).setAllowNull(True)
    ft.add("datestam",DataTypes.STRING,30).setAllowNull(True)
    ft.add("timestam",DataTypes.STRING,30).setAllowNull(True)

    attrg = ft.add("geometry",DataTypes.GEOMETRY)
    attrg.setGeometryType(
      GeometryLocator.getGeometryManager().getGeometryType(
        Geometry.TYPES.POINT,
        Geometry.SUBTYPES.GEOM2D
      )
    )
```

```
    return ft

  def openShape(self):
    dataManager = DALLocator.getDataManager()
    openparams = dataManager.createStoreParameters("Shape")
    openparams.setDynValue("shpFile",self.__outputFile)
    openparams.setDynValue("crs",self.__projection)
    featurestore = dataManager.openStore("Shape",openparams)
    return featurestore

  def createShape(self):
    dataManager = DALLocator.getDataManager()

    serverparams = dataManager.createServerExplorerParameters("FilesystemExplorer")
    serverparams.setDynValue("root", self.__outputFile.getParent())
    server = dataManager.openServerExplorer("FilesystemExplorer", serverparams)
    addparams = server.getAddParameters("Shape")
    addparams.setDefaultFeatureType(self.createFeatureType())
    addparams.setDynValue("shpFile",self.__outputFile)
    addparams.setDynValue("crs",self.__projection)
    addparams.setDynValue("geometryType",Geometry.TYPES.POINT)
    server.add("Shape",addparams, False)

  def run(self):
    root = self.__outputFile.getParentFile()
    metadata_reader = ImageGPSMetadata()
    files = self.__inputFolder.list()
    if self.__outputFile.exists():
        self.__onFinish()
        return

    self.createShape()

    store = self.openShape()
    store.edit()
    n = 0
    for fname  in files:
      n += 1
      base, ext = os.path.splitext(fname)
      if not ext.lower() in (".jpg",".png",".jpeg"):
        continue
      f  = File(self.__inputFolder,fname)

      metadata_reader.load(f)
      feature = store.createNewFeature()
      feature.setDefaultGeometry(metadata_reader.getPoint())
      feature.set("id", n)
      feature.set("text", fname)
      feature.set("fname", fname)
      feature.set("relpath", relpath(f.getAbsolutePath(),root.getAbsolutePath()))
      feature.set("abspath", f.getAbsolutePath())
      feature.set("altitude", metadata_reader.getAltitude(""))
      feature.set("altitudere", metadata_reader.getAltitudeRef(""))
      feature.set("datum", metadata_reader.getDatum(""))
      feature.set("datestam", metadata_reader.getDate(""))
      feature.set("timestam", metadata_reader.getTime(""))
      store.insert(feature)

    store.finishEditing()
    if self.__addLayerToView:
      layer = MapContextLocator.getMapContextManager().createLayer(
        self.__outputFile.getName(),
        store
      )
      currentView().getMapContext().getLayers().addLayer(layer)
    self.__onFinish()



class Photo2shapePanel(FormPanel):
```

```
  def __init__(self):
    FormPanel.__init__(self,script.getResource("photo2shape3.xml"))
    self.getPanel().setPreferredSize(Dimension(500,140))
    self.txtInputFolder.setText(script.getResource("data/test-images").getAbsolutePath())
    self.txtOutputFile.setText(script.getResource("data/photos.shp").getAbsolutePath())

  def showWindow(self):
    FormPanel.showWindow(self, "Photo2shape")

  def btnInputFolder_click(self, *args):
    f = openFolderDialog("Select input folder")
    if f == None or len(f)<1:
      return
    self.txtInputFolder.setText(f[0].getAbsolutePath())

  def btnOutputFile_click(self, *args):
    f = saveFileDialog("Select output shape")
    if f == None or len(f)<1:
      return
    self.txtOutputFile.setText(f[0].getAbsolutePath())

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

  def btnMakeShape_click(self, *args):
    if self.chkRemoveShape.isSelected():
      try:
        f = os.path.splitext(self.txtOutputFile.getText())[0]
        os.remove(f+".shp")
        os.remove(f+".shx")
        os.remove(f+".dbf")
      except:
        pass
    if os.path.exists(self.txtOutputFile.getText()) :
      msgbox("output shapefile already exist")
      return

    self.btnMakeShape.setEnabled(False)
    self.btnClose.setEnabled(False)

    process = ProcessFolder()
    process.setInputFoldername(self.txtInputFolder.getText())
    process.setOutputFilename(self.txtOutputFile.getText())
    process.setAddLayerToView(self.chkInsertLayer.isSelected())
    process.setOnFinish(self.processFinished)
    process.start()

  def processFinished(self):
    self.btnMakeShape.setEnabled(True)
    self.btnClose.setEnabled(True)



def main(*args):
    win = Photo2shapePanel()
    win.showWindow()
```