```
# encoding: utf-8

import gvsig
from gvsig import logger, LOGGER_WARN
from gvsig import openStore
from gvsig import getResource

from java.awt import GridBagConstraints
from java.beans import PropertyVetoException
from javax.swing.event import AncestorListener

from org.gvsig.tools import ToolsLocator
from org.gvsig.app import ApplicationLocator
from org.gvsig.app.project.documents.view import ViewManager
from org.gvsig.fmap.mapcontrol.tools.Behavior import PointBehavior
from org.gvsig.fmap.mapcontrol.tools.Listeners import AbstractPointListener
from org.gvsig.fmap.crs import CRSFactory

from blocks import getBlockByPoint, BlockPanel

MYTOOL_NAME = "MyApplication.Infotool"

class ApplicationInitializer(object):

  def __init__(self):
    pass

  def createViewWindow(self):
    i18nManager = ToolsLocator.getI18nManager()
    application = ApplicationLocator.getManager()

    projectManager = application.getProjectManager()

    # 1. Create a new view and set the name.
    viewManager = projectManager.getDocumentManager(ViewManager.TYPENAME)
    view = viewManager.createDocument()
    view.setName(i18nManager.getTranslation("My Application View"))
    view.setProperty("MyApplicationView", True)

    # Setting view's projection to shapefile's known CRS
    view.getMapContext().setProjection(CRSFactory.getCRS("EPSG:23030"))

    # 2. Create a new layer with the blocks
    blocksStore = openStore(
      "Shape",
      shpFile=getResource(__file__,"data","blocks.shp"),
      CRS="EPSG:23030"
    )

    layer = application.getMapContextManager().createLayer(
      i18nManager.getTranslation("Blocks"),
      blocksStore
    )

    # Add a new property to the layer to identify.
    layer.setProperty("MyApplicationLayer", True)

    # 3. Add this layer to the mapcontext of the new view.
    view.getMapContext().getLayers().addLayer(layer)
    layer.setActive(True)
    # 4. Add the view to the current project.
    projectManager.getCurrentProject().add(view)

    # 5. Force to show the view's window.
    viewWindow = viewManager.getMainWindow(view)

    application.getUIManager().addWindow(viewWindow, GridBagConstraints.CENTER)
    try:
      application.getUIManager().setMaximum(viewWindow, True)
    except PropertyVetoException, e:
```

```
      logger("Problemas maximizando la vista", LOGGER_WARN, e)

    # 6. Register my tool in the mapcontrol of the view.
    listener = PropertiesOfBlockListener(blocksStore)
    viewWindow.getMapControl().addBehavior(MYTOOL_NAME, PointBehavior(listener))

    return viewWindow

class PropertiesOfBlockListener(AbstractPointListener,AncestorListener):
  def __init__(self, blocksStore):
    self.__blockPanel = None
    self.__blocksStore = blocksStore

  def point(self, event):
    """from PointListener"""
    block = getBlockByPoint(self.__blocksStore, event.getMapPoint())
    if block==None:
      return
    if self.__blockPanel == None:
      self.__blockPanel = BlockPanel()
      self.__blockPanel.asJComponent().addAncestorListener(self)
      self.__blockPanel.setBlock(block)
      self.__blockPanel.showTool("Informacion sobre la manzana")
    else:
      self.__blockPanel.setBlock(block)

  def ancestorRemoved(self, event):
    """from AncestorListener"""
    self.__blockPanel = None

  def ancestorMoved(self, event):
    """from AncestorListener"""
    pass

  def ancestorAdded(self, event):
    """from AncestorListener"""
    pass

def main(*args):
    initializer = ApplicationInitializer()
    initializer.createViewWindow()
```