# 第一章

## 1. QGIS python 控制台的使用

### 1.1创建一个点的内存图层：

layer = QgsVectorLayer('Point?crs=epsg:4326', 'MyPoint' ,'memory')  
pr = layer.dataProvider()  
pt = QgsFeature()  
point1 = QgsPoint(20,20)  
pt.setGeometry(QgsGeometry.fromPoint(point1))  
pr.addFeatures([pt])  
layer.updateExtents()  
QgsMapLayerRegistry.instance().addMapLayers([layer])

## 2 pythono ScriptRunner插件的使用

## 3 搭建QGIS IDE

# 第二章 查询

## 1． 加载矢量文件

### 1.1 加载shp文件

QgsVectorLayer\* pLayer=new QgsVectorLayer(tr("C:\\Users\\zhangzl\\Desktop\\new2\\bbb.shp"),"new","ogr");

if(!pLayer->isValid())

{

mQGisIface->messageBar()->pushMessage("not valid");

}

QgsMapLayerRegistry::instance()->addMapLayer(pLayer);

### 1.2 从数据库加载

1）postgre

QgsDataSourceURI\* uri=new QgsDataSourceURI();

uri->setConnection(tr("spacialdb.com"),tr("9999"),"lzmjzm\_hwpqlf","lzmjzm\_hwpqlf","0e9fcc39");

uri->setDataSource("public","islands","wkb\_geometry","");

QgsVectorLayer\* pLayer=new QgsVectorLayer(uri->uri(),"Islands","postgres");

if(!pLayer->isValid())

{

mQGisIface->messageBar()->pushMessage(tr("not valid"));

}

QgsMapLayerRegistry::instance()->addMapLayer(pLayer);

2）sqlite

QgsDataSourceURI\* uri=new QgsDataSourceURI();

uri->setConnection(tr("spacialdb.com"),tr("9999"),"lzmjzm\_hwpqlf","lzmjzm\_hwpqlf","0e9fcc39");

uri->setDatabase("abc.sqlite");

3) MySQL

QString uri="MySQL:ddname,host=localhost,port=3306,user=root,password=xxx | layername=my\_table";

QgsVectorLayer\* pLayer=new QgsVectorLayer(uri,"my\_table","ogr");

4)WFS

uri="http://localhost:8080/geoserver/wfs?srsname=EPSG:23030&typename=union&version=1.0.0&...";

QgsVectorLayer\* pLayer=new QgsVectorLayer("my\_wfs\_layer","WFS");

## 2 检出矢量图形

QgsVectorLayer\* pLayer=(QgsVectorLayer\*)mQGisIface->mapCanvas()->currentLayer();

if(!pLayer->isValid())

{

mQGisIface->messageBar()->pushMessage("not valid");

}

QgsMapLayerRegistry::instance()->addMapLayer(pLayer);

QgsFeatureIterator feaIter=pLayer->getFeatures();

QgsFeature fea;

feaIter.nextFeature(fea);

QgsGeometry\* pGeo=fea.geometry();

QgsPoint pt=pGeo->asPoint();

## 3 检出矢量属性

QgsVectorLayer\* pLayer=new QgsVectorLayer(tr("E:\\beifen\\new2\\bbb.shp"),"new","ogr");

QgsFeatureIterator features=pLayer->getFeatures();

QgsFeature f;

features.nextFeature(f);

QgsAttributes attrs=f.attributes();

QList<QgsField> flds=f.fields()->toList();

QString fldName=flds.at(0).name();

## 4 用几何要素筛选图层

QgsVectorLayer\* lyrPolys=new QgsVectorLayer(tr("E:\\testData\\tutor\\polygon.shp"),"polys","ogr");

QgsVectorLayer\* lyrPts=new QgsVectorLayer(tr("E:\\testData\\tutor\\point.shp"),"pts","ogr");

QgsMapLayerRegistry::instance()->addMapLayer(lyrPts);

QgsMapLayerRegistry::instance()->addMapLayer(lyrPolys);

QgsFeatureIterator feaIter=lyrPolys->getFeatures();

QgsFeature feat;

while(feaIter.nextFeature(feat))

{

QgsFeatureIterator ptFeaIter=lyrPts->getFeatures(QgsFeatureRequest().setFilterRect(feat.geometry()->boundingBox()).setFlags(QgsFeatureRequest::ExactIntersect));

QgsFeature ptFeat;

while(ptFeaIter.nextFeature(ptFeat))

{

if(ptFeat.geometry()->within(feat.geometry()))

{

lyrPts->select(ptFeat.id());

}

}

}

mQGisIface->setActiveLayer(lyrPolys);

mQGisIface->zoomToActiveLayer();

## 5 用属性筛选图层

QgsVectorLayer\* lyrPts=new QgsVectorLayer(tr("E:\\testData\\tutor\\point.shp"),"pts","ogr");

QgsMapLayerRegistry::instance()->addMapLayer(lyrPts);

QString strSQL=tr("\"classID\"='a'");

QgsFeatureIterator selection=lyrPts->getFeatures(QgsFeatureRequest().setFilterExpression(strSQL));

QgsFeatureIds fids;

QgsFeature fea;

while(selection.nextFeature(fea))

{

fids<<fea.id();

}

lyrPts->setSelectedFeatures(fids);

mQGisIface->mapCanvas()->zoomToSelected();

## 6 基于要素构建缓冲区

QgsVectorLayer\* lyrPts=new QgsVectorLayer(tr("E:\\testData\\tutor\\point.shp"),"pts","ogr");

QgsMapLayerRegistry::instance()->addMapLayer(lyrPts);

QgsFeatureIterator fts=lyrPts->getFeatures();

QgsFeature fea;

fts.nextFeature(fea);

lyrPts->setSelectedFeatures(QgsFeatureIds()<<fea.id());

QgsGeometry\* pGeo=fea.geometry()->buffer(20000,8);

QgsVectorLayer\* buffLyr=new QgsVectorLayer("Polygon?crs=EPSG:4326","Buffer","memory");

QgsVectorDataProvider\* pr=buffLyr->dataProvider();

QgsFeature pNewFea;

pNewFea.setGeometry(pGeo);

pr->addFeatures(QgsFeatureList()<<pNewFea);

buffLyr->updateExtents();

buffLyr->setLayerTransparency(60);

QgsMapLayerRegistry::instance()->addMapLayer(buffLyr);

## 7 测量两点间的距离

QgsVectorLayer\* lyrPts=new QgsVectorLayer(tr("E:\\testData\\tutor\\point.shp"),"pts","ogr");

QgsMapLayerRegistry::instance()->addMapLayer(lyrPts);

QgsFeatureIterator fts=lyrPts->getFeatures();

QgsFeature feaFirst;

fts.nextFeature(feaFirst);

QgsFeature feaSecond;

fts.nextFeature(feaSecond);

QgsDistanceArea da;

double dLen=da.measureLine(feaFirst.geometry()->asPoint(),feaSecond.geometry()->asPoint());

QGis::UnitType inType=QGis::UnitType::Degrees;

da.convertMeasurement(dLen,inType,QGis::UnitType::Meters,false);

## 8 测量一条多段线的长度

QgsVectorLayer\* lyrPts=new QgsVectorLayer(tr("E:\\testData\\tutor\\line.shp"),"pts","ogr");

QgsMapLayerRegistry::instance()->addMapLayer(lyrPts);

QgsFeatureIterator fts=lyrPts->getFeatures();

QgsFeature feaFirst;

fts.nextFeature(feaFirst);

QgsDistanceArea da;

da.setEllipsoidalMode(TRUE);

double d=da.measureLine(feaFirst.geometry()->asPolyline().toList());

QGis::UnitType inUnit=QGis::UnitType::Degrees;

da.convertMeasurement(d,inUnit,QGis::UnitType::Meters,FALSE);

## 9 计算多边形的面积

QgsVectorLayer\* lyrPts=new QgsVectorLayer(tr("E:\\testData\\tutor\\polygon.shp"),"pts","ogr");

QgsMapLayerRegistry::instance()->addMapLayer(lyrPts);

QgsFeatureIterator fts=lyrPts->getFeatures();

QgsFeature feaFirst;

fts.nextFeature(feaFirst);

QgsPolyline poly=feaFirst.geometry()->asPolygon().at(0);

QgsDistanceArea da;

double d=da.measurePolygon(poly.toList());

QGis::UnitType inUnit=QGis::UnitType::Degrees;

da.convertMeasurement(d,inUnit,QGis::UnitType::Meters,TRUE);

## 10 创建空间索引（加快空间操作速度）

QgsVectorLayer\* lyrPts=new QgsVectorLayer(tr("E:\\testData\\tutor\\point.shp"),"pts","ogr");

QgsMapLayerRegistry::instance()->addMapLayer(lyrPts);

QgsFeatureIterator fts=lyrPts->getFeatures();

QgsFeature feaFirst;

QgsSpatialIndex index;

while(fts.nextFeature(feaFirst))

{

index.insertFeature(feaFirst);

}

QList<QgsFeatureId> lst=index.nearestNeighbor(feaFirst.geometry()->asPoint(),4);

## 11 计算直线的方位

QgsVectorLayer\* lyrPts=new QgsVectorLayer(tr("E:\\testData\\tutor\\point.shp"),"pts","ogr");

QgsMapLayerRegistry::instance()->addMapLayer(lyrPts);

QgsFeatureIterator fts=lyrPts->getFeatures();

QgsFeature feaFirst;

fts.nextFeature(feaFirst);

QgsDistanceArea da;

da.setEllipsoidalMode(TRUE);

QgsPolyline points=feaFirst.geometry()->asPolyline();

QgsPoint first=points.at(0);

QgsPoint last=points.at(-1);

double r=da.bearing(first,last);

## 12 从电子表格加载数据

略

# 第三章 矢量编辑

## 1 创建一个内存图层

QgsVectorLayer\* vectorLyr=new QgsVectorLayer("Point?crs=epsg:4326&field=city:string(25)&field=popullation:nt","Layer 1","memory");

QgsMapLayerRegistry::instance()->addMapLayer(vectorLyr);

## 2 矢量图层中添加一个要素

QgsVectorLayer\* pLayer=new QgsVectorLayer("E:\\testData\\tutor\\polygon.shp","polygon","ogr");

QgsVectorDataProvider\* vpr=pLayer->dataProvider();

QgsPoint pt1(-123.26,49.06);

QgsPoint pt2(-127.19,43.07);

QgsPoint pt3(-120.70,35.21);

QgsPoint pt4(-115.89,40.02);

QgsPoint pt5(-113.04,48.47);

QgsPoint pt6(-123.26,49.06);

QgsPolyline pLine;

pLine<<pt1<<pt2<<pt3<<pt4<<pt5<<pt6;

QgsPolygon pPolygon;

pPolygon<<pLine;

QgsGeometry\* pGeo=QgsGeometry::fromPolygon(pPolygon);

QgsFeature f;

f.setGeometry(pGeo);

vpr->addFeatures(QgsFeatureList()<<f);

QgsMapLayerRegistry::instance()->addMapLayer(pLayer);

## 3 为矢量新要素赋予属性

QgsVectorLayer\* pLayer=new QgsVectorLayer("E:\\testData\\tutor\\point.shp","polygon","ogr");

QgsVectorDataProvider\* vpr=pLayer->dataProvider();

QgsGeometry\* pGeo=QgsGeometry::fromPoint(QgsPoint(-74.13401,40.62148));

QgsFields flds=vpr->fields();

QgsFeature f(flds);

f.setGeometry(pGeo);

f.setAttribute("classID","mama");

vpr->addFeatures(QgsFeatureList()<<f);

QgsMapLayerRegistry::instance()->addMapLayer(pLayer);

## 4 为矢量增加一个字段

QgsVectorLayer\* pLayer=new QgsVectorLayer("E:\\testData\\tutor\\point.shp","polygon","ogr");

QgsVectorDataProvider\* vpr=pLayer->dataProvider();

vpr->addAttributes(QList<QgsField>()<<QgsField("Admission",QVariant::Double));

pLayer->updateFields();

QgsMapLayerRegistry::instance()->addMapLayer(pLayer);

## 5 移动要素

QgsVectorLayer\* pLayer=new QgsVectorLayer("E:\\testData\\tutor\\point.shp","point","ogr");

QgsGeometry\* pGeo=QgsGeometry::fromPoint(QgsPoint(-74.20378,40.89642));

QgsGeometryMap map;

map.insert(0,\*pGeo);

pLayer->dataProvider()->changeGeometryValues(map);

QgsMapLayerRegistry::instance()->addMapLayer(pLayer);

## 6 修改要素的属性

QgsVectorLayer\* pLayer=new QgsVectorLayer("E:\\testData\\tutor\\point.shp","point","ogr");

int id1=0;

int id2=1;

int fldIdx1=pLayer->fieldNameIndex("classID");

int fldIdx2=pLayer->fieldNameIndex("Admission");

QgsChangedAttributesMap maps;

QgsAttributeMap map;

map.insert(fldIdx1,"first");

map.insert(fldIdx2,1);

QgsAttributeMap map2;

map2.insert(fldIdx1,"second");

map2.insert(fldIdx2,2);

maps.insert(id1,map);

maps.insert(id2,map2);

pLayer->dataProvider()->changeAttributeValues(maps);

QgsMapLayerRegistry::instance()->addMapLayer(pLayer);

## 7 删除要素

QgsVectorLayer\* pLayer=new QgsVectorLayer("E:\\testData\\tutor\\point.shp","point","ogr");

pLayer->dataProvider()->deleteFeatures(QgsFeatureIds()<<0<<1<<2);

QgsMapLayerRegistry::instance()->addMapLayer(pLayer);

8 删除字段

QgsVectorLayer\* pLayer=new QgsVectorLayer("E:\\testData\\tutor\\point.shp","point","ogr");

pLayer->dataProvider()->deleteAttributes(QgsAttributeIds()<<1);

pLayer->updateFields();

QgsMapLayerRegistry::instance()->addMapLayer(pLayer);

# 第五章 创建动态地图

## 1. 获取及改变画布单位

QgsMapCanvas canvas=mQGisIface->mapCanvas();

QSize size=canvas.size();

QGis::UnitType type=canvas.mapUnits();

canvas.setMapUnits(QGis::Meters);

type=canvas.mapUnits();

size=canvas.size();

## 2．遍历图层