Index

Module commonsdialog reference

Contents

Module commonsdialog

Window message, msgbox

Shows window message with acept button only.

msgbox(message [, title, messageType])

- msgbox(message [, title, messageType])
  - param message: Text to show
  - type message: string
  - param title (optional): Window title.
  - type title: string
  - messageType, integer [FORBIDEN| IDEA| WARNING| QUESTION]: Show window icon to emphasize message. Reconized constant values
    - FORBIDEN: Forbiden symbol
    - IDEA: Lamp symbol
    - WARNING: Warning symbol
    - QUESTION: question symbol

Confirm window, confirmDialog

Shows a message to the user to choose one option

confirmDialog(message, [title, optionType, messageType])

- confirmDialog(message, [title, optionType, messageType])
  - param message: Text to show
  - type message: string
  - param title (optional): Window title.
  - type title: string
  - optionType, integer [YES_NO | YES_NO_CANCEL | ACEPT_CANCEL] (optional): Buttons to show
    - YES_NO: Yes/no buttons.
    - YES_NO_CANCEL: yes/no/cancel buttons.
    - ACEPT_CANCEL: accept/cancel buttons.
  - type messageType: integer
  - param messageType [FORBIDEN | IDEA | WARNING | QUESTION] (optional): Show window icon to emphasize message.
    - FORBIDEN: Forbiden symbol
    - IDEA: Lamp symbol
    - WARNING: Warning symbol
    - QUESTION: question symbol

Input window , inputbox

Displays an input box to ask the user to enter a string

inputbox(message, [title, messageType, initialValue])

- inputbox(message, [title, messageType, initialValue])
  - param message: Text to show
  - type message: string
  - param title (optional): Window title.
  - type messageType: integer
  - param messageType [FORBIDEN | IDEA | WARNING | QUESTION] (optional): Show window icon to emphasize message.
    - FORBIDEN: Forbiden symbol

        - IDEA: Lamp symbol
        - WARNING: Warning symbol
        - QUESTION: question symbol
    - param initialValue: Initial text to show in the text area
    - type initialValue: string

Module gvsig reference

gvSIG Documents reference

class Project

Represents a gvSIG project

- getView([name]): Return active view, view called 'name' or None
  - param name: (Optional) View name
  - type name: string
- getTable([name]): Return active Table Document, or Table Document called 'name' or None
  - param name: (Optional) Table Document name
  - type name: string
- getProjectionCode(): Return Project projection name

class View

Represents gvSIG view document

- getLayer([name]): Return one of the view layers documents. If name is None return active view layer if the view has one. If name is not None return view layer called name. Else return None.
  - param name: (Optional) View name
  - type name: string
- getMap(): Return view mapContext
- addLayer(layer): Add a gvSIG layer to the view
  - param layer: layer to add
  - type name: layer
- getLayers(): Return iterable view layers set
- getGraphicsLayer(): Return view graphics layer

class Table

Represents gvsig TableDocument It is able to manage its own data set

- features([expresion]):Return table features set
  - param expresion: (Optional) Filter to apply to the feature set to select determinates features that match with expression
  - type expresion: string
  - return: FeatureSet
- edit(): Set data store in edition mode
- append(values): Create a new feature from given values and insert it in the feature set
  - param values: dictionary with name property value or list named params
  - type values: dict
- updateSchema(schema): Update table data definition with schema.
  - param schema: Modified table schema
  - type schema: Editable Schema
- update(feature): Update exist feature in the layer featureSet
  - param feature: Feature to modify in data set
  - type feature: Feature
- getSchema(): Return layer schema definition
- commit(): Finish layer edition
- abort(): Cancel layer edition
- getSelection(): Return layer features selected set

class Layer

Represents gvsig view layer . It is also is able to manage its own data set

- features([expresion]): Return layer features set
  - param expresion: (Optional) Filter to apply to the feature set to select determinates features that match with expression

    - type expresion: string
    - return: FeatureSet
- edit(): Set data store in edition mode
- append(values): Create a new feature from given values and insert it in the feature set
    - param values: dictionary with name property value or list named params
    - type values: dict
- updateSchema(schema): Update table data definition with schema.
    - param schema: Modified table schema
    - type schema: Editable Schema
- update(editableFeature): Update exist feature in the layer featureSet
    - param editableFeature: editableFeature
    - type editableFeature: Java editableFeature
- getSchema(): Return layer schema definition
- commit(): Finish layer edition
- abort(): Cancel layer edition
- getSelection(): Return layer features selected set (FeatureSet)
- getTypeVectorLayer(): Return layer geometry type.

Documents util Functions

- currentProject(): Return the current gvSIG proyect
- currentDocument(): Return the current active document (View or TableDocument)
- currentView(): Return the current active view document or None. If no view active view raise RuntimeException
- currentLayer(): Return current view active layer

Vectorial data reference

class Table

Represents gvsig TableDocument It is able to manage its own data set

- features([expresion]): Return table features set
  - param expresion: (Optional) Filter to apply to the feature set to select determinates features that match with expression
  - type expresion: string
  - return: FeatureSet
- edit(): Set data store in edition mode
- append(values): Create a new feature from given values and insert it in the feature set
  - param values: dictionary with name property value or list named params
  - type values: dict
- updateSchema(schema): Update table data definition with schema.
  - param schema: Modified table schema
  - type schema: Editable Schema
- update(feature): Update exist feature in the layer featureSet
  - param feature: Feature to modify in data set
  - type feature: Feature
- getSchema(): Return layer schema definition
- commit(): Finish layer edition
- abort(): Cancel layer edition
- getSelection(): Return layer features selected set

class Layer

Represents gvsig view layer . Also is able to manage its own data set

- features([expresion]): Return layer features set
  - param expresion: (Optional) Filter to apply to the feature set to select determinates features that match with expression
  - type expresion: string
  - return: FeatureSet
- edit(): Set data store in edition mode
- append(values): Create a new feature from given values and insert it in the feature set
  - param values: dictionary with name property value or list named params
  - type values: dict
- updateSchema(schema): Update table data definition with schema.
  - param schema: Modified table schema
  - type schema: Editable Schema
- update(editableFeature): Update exist feature in the layer featureSet
  - param editableFeature: editableFeature
  - type editableFeature: Java editableFeature
- getSchema(): Return layer schema definition
- commit(): Finish layer edition
- abort(): Cancel layer edition
- getSelection(): Return layer features selected set (FeatureSet)
- getTypeVectorLayer(): Return layer geometry type.

class FeatureSet

Return Layer/Table set of features

- getCount(): Return set number of ocurrences

class Feature

Represents layer/table feature data

- geometry(): Return feature default geometry
- getValues(): Return dictionary with key name property and value of the feature attributes
- edit(): Return editable feature instance #FIXME

class Schema

Layer/table feature definition

- append(name, type[, size][, default]): Adds property to feature properties definition.
    - param name: Feature property name
    - type name: String
    - param type: Feature property type
    - type name: String
    - param size: Feature property size
    - type size: int
    - param default: Feature property default value
    - return: new atribute
- get(name, default=None): Looks for property name named and if not find it return default. Return a feature attribute descriptor that contains information about one of the attributes in a feature, such as its name, data type or precision.
    - param name: Property name
    - type name: string
    - param default: Return value if not find property name
    - type value: Object
    - return: AttributeDescriptor
- modify(): set definition in edition mode to add new properties
- getCopy(): Return schema copy

Vectorial data util functions

- createSchema([schema]): Return layer definition. If schema is not None the definition is used to create new schema
    - param schema: (Optional)
    - type: schema: Schema
- createShape(definition, filename, geometryType[, CRS]): Create a new shape layer. If CRS is not defined will use "wsg84"
    - param definition: layer data definition
    - type definition: Schema
    - param filename: absolute path for shape files.
    - type filename: string
    - param geometryType: geometry type for shape
    - type geometryType: string
    - param CRS: Projection name
    - type CRS: string
    - return: new shape layer
    - rtype: Layer
- createDBF(definition, DbfFile[, CRS]): Create a new dbf file that could be added to the gvSIG project as a TableDocument
    - param definition: layer data definition
    - type definition: Schema
    - param DbfFile: absolute path for shape files.
    - type DbfFile: string
    - param geometryType: geometry type for shape
    - type geometryType: string
    - return: new dbf
    - rtype: Table

Simbology reference

Contents

Simple symbology simbols

- SimplePointSymbol(color): Return simple point symbol using parameter color
  - param color: Color name
  - return: gvSIG point symbol
- SimpleLineSymbol(color): Return simple line symbol using parameter color
  - param color: Color name
  - return: gvSIG line symbol
- SimplePoligonSymbol(color): Return simple poligon symbol using parameter color
  - param color: Color name
  - return: gvSIG poligon symbol

Module geom reference

Contents

Note

To get more information about Geometry methods see Geometry javadoc

Commons geometries methods

Interface Geometry: This interface is equivalent to the GM_Object specified in ISO 19107. A geometric object shall be a combination of a coordinate geometry and a coordinate reference system.

- area(): Return geometry area
- buffer(distance): Computes a buffer area around this geometry having the given width
- centroid(): Return geometry centroid
- distance(geometry): Returns the minimum distance between this Geometry and the specified geometry.
- getEnvelope(): Returns the minimum bounding box for this Geometry.
- getGeometryType(): Instance of the GeometryType associated to this geometry
- getDimension(): Returns the largest number n such that each direct position in a geometric set can be associated with a subset that has the direct position in its interior and is similar (isomorphic) to Rn, Euclidean n-space

Geometry types

Point

This interface is equivalent to the GM_Point specified in ISO 19107. It is one of the basic data type for a geometric object consisting of one and only one point. They do not have volume, area, length, or any other higher-dimensional analogue.

- getX(): Returns the X coordinate
- getY(): Returns the Y coordinate
- getCoordinateAt(dimension): Gets the coordinate in a concrete dimension
    - param dimension: Dimension coordinate to get
    - type dimension: integer
    - allowed values: 0=X, 1=Y, 2=Z
- setX(value): Sets the X coordinate
    - param value: Value of the dimension X
    - type value: double
- setY(value): Sets the Y coordinate
    - param value: Value of the dimension Y
    - type value: double
- setCoordinateAt(dimension, value): Sets a ordinate in a concrete dimension
    - param dimension: Dimension coordinate to set
    - type dimension: integer
    - param value: Value of the dimension
    - type value: double

Line

This interface is equivalent to the GM_OrientableCurve specified in ISO 19107. Curves are continuous, connected, and have a measurable length in terms of the coordinate system.

- addVertex(point): Adds a vertex to the curve
    - param point: Point to addLayer
    - type: Point
- getNumVertices(): Gets the number of vertex of the curve
- getVertex(index): Gets a vertex
    - param index: Geometry vertex index
    - type index: integer

Polygon

This interface is equivalent to the GM_Surface specified in ISO 19107. This is the basis for 2-dimensional geometry

- addVertex(point): Adds a vertex to the curve
    - param point: Point to addLayer
    - type: Point
- getNumVertices(): Gets the number of vertex of the curve
- getVertex(index): Gets a vertex
    - param index: Geometry vertex index
    - type index: integer
- getSurfaceAppearance(): Gets surface appearance
- setSurfaceAppearance(appearance): Sets surface appearance

Multigeometry

This interface is equivalent to the GM_Aggregate specified in ISO 19107

- getPrimitiveAt(int i): Returns one of the Primitive's that is in a concrete position. - param i: geometry position - type i: integer
- getPrimitivesNumber(): Returns the number of Primitive's that composes this multi geometry.

Geometry util functions

- createGeometry(type[, subtype]): Create a new geometry with a concrete type and subtype

    - param type: geometry type
    - type type: integer
    - param subtype: (Optional) geometry subtype
    - type type: integer
    - return: geometry
    - rtype: Geometry
- createPoint([x][, y][, subtype]): Create a new point with a subtype dimensions and sets the value for 'X' and 'Y' coordinates. If is used without parameters will return new Point2D with (0,0) coordinates
    - param x:(Optional) X coordinate value
    - type x: double
    - param y:(Optional) Y coordinate value
    - type y: double
    - param subtype: (Optional) geometry dimensions
    - type: integer
    - return: Point
    - rtype: Point
- createMultiPoint(points[, subtype]): Create a new multipoint with a subtype from a list values of X and Y like ([x1, y1], [x2, y2], ..., [xn, yn])
    - param points: list of tuples with X and Y values
    - type points: list
    - param subtype: (Optional) geometry dimensions
    - type: integer
    - return: multipoint
    - rtype: multipoint