Localization
in MarushkaDesign

GEOVAP

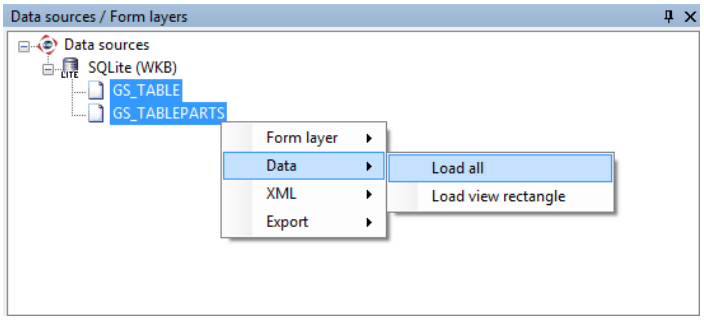
CONTENTS

1 Aim of the Example

In this example we will demonstrate a few types of localization in MarushkaDesign. This example was created in version 4.0.1.0, so it does not have to be compatible with older versions.

2 Working with Example

- Unzip the **Localization_EN.zip** into **c:\MarushkaExamples** folder. The target folder must be respected due to interconnection of paths with the project. In the case of placing the files in the different folder, it would not be possible to work with an example.
- Open the project **Localization_CZ.xml** in MarushkaDesign environment.
- Select both form layers, in context menu choose Data – Load all:



- In map window choose „Fit all“:

- Launch the local web server:

3 Dialog Box Sample

Fig 1: Result of the query ***1 Localization***

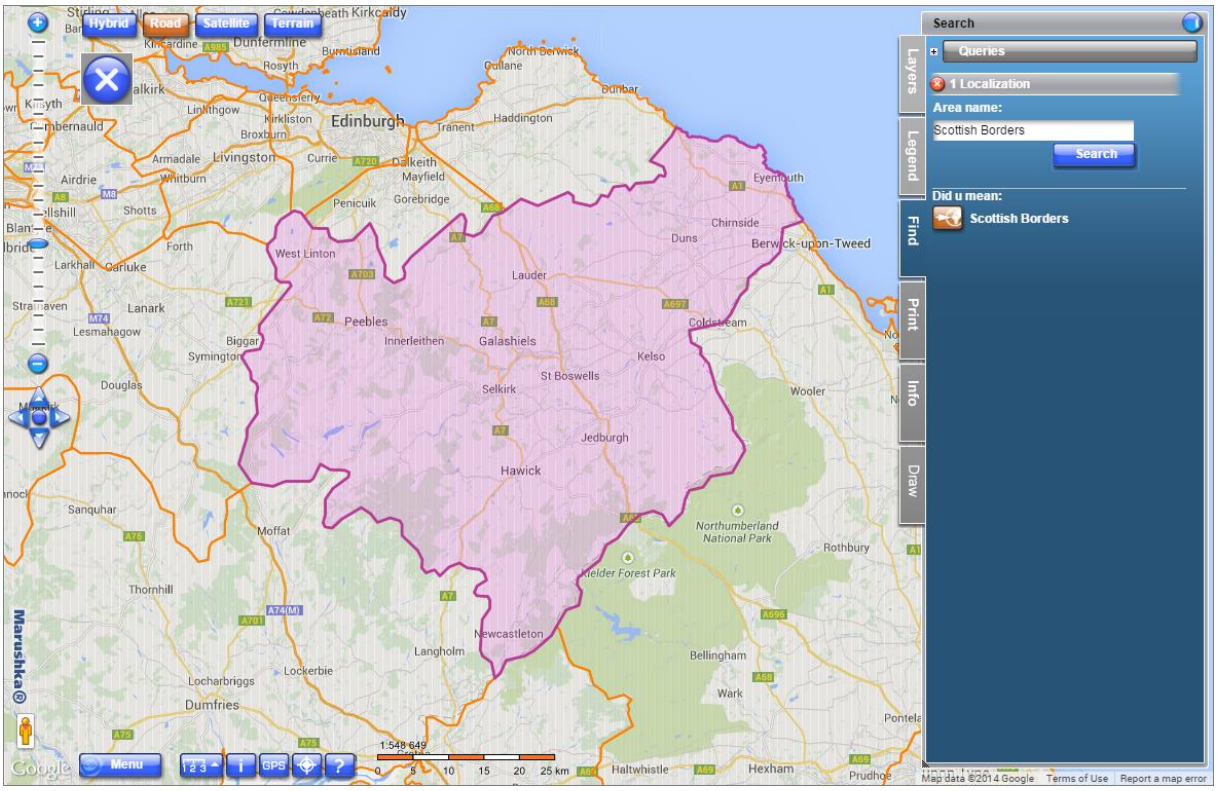


Fig 2: Result of the query ***2 Localization polygons***

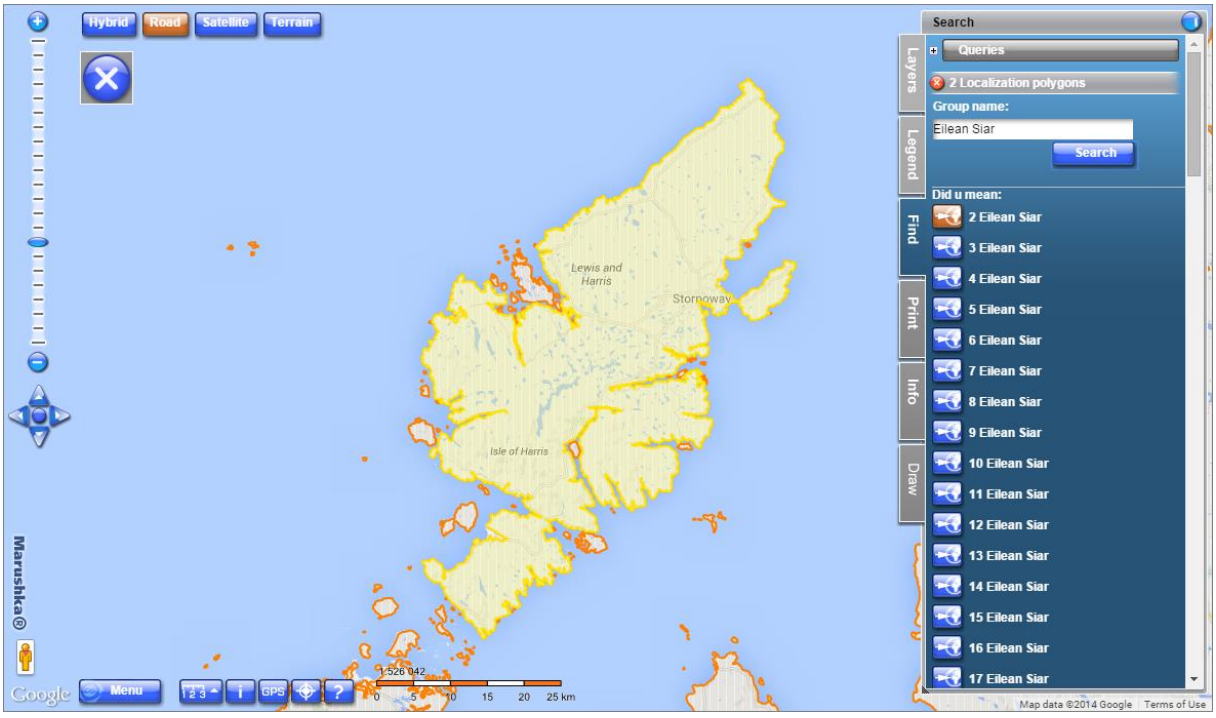


Fig 3: Result of the query ***3 Localize group***

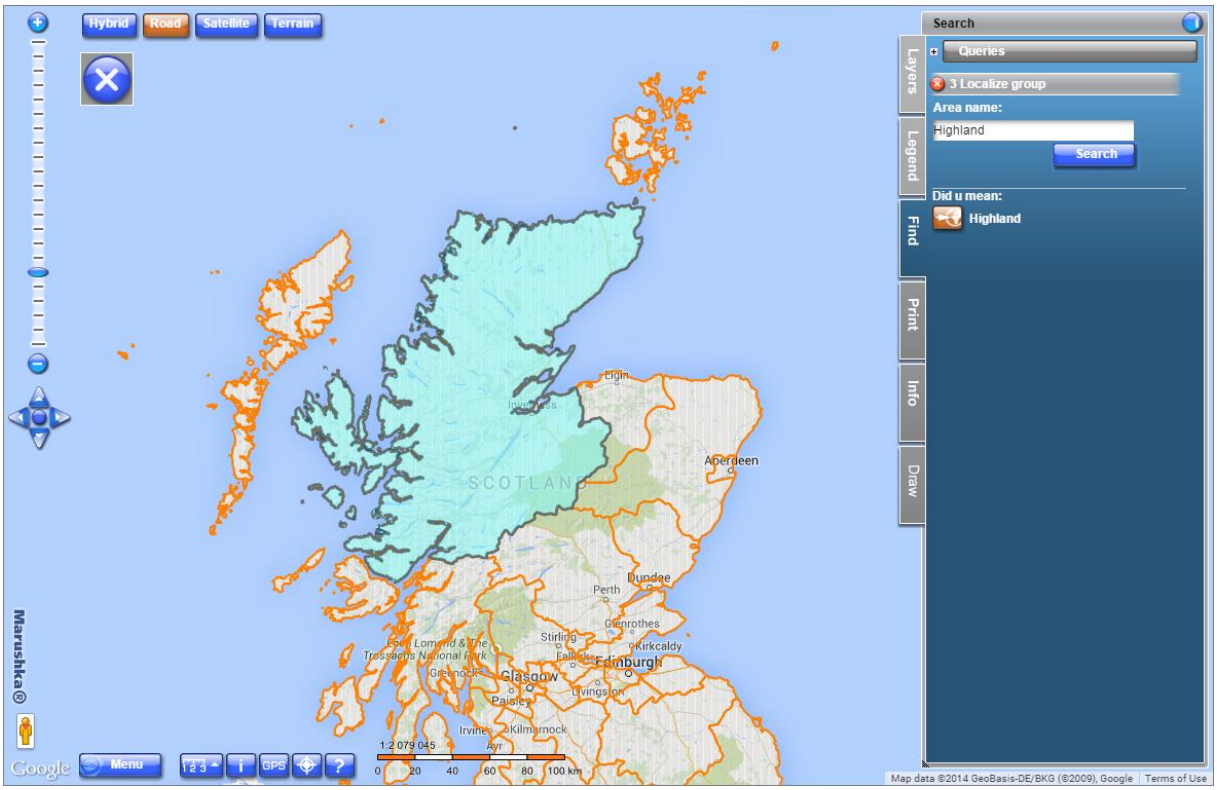
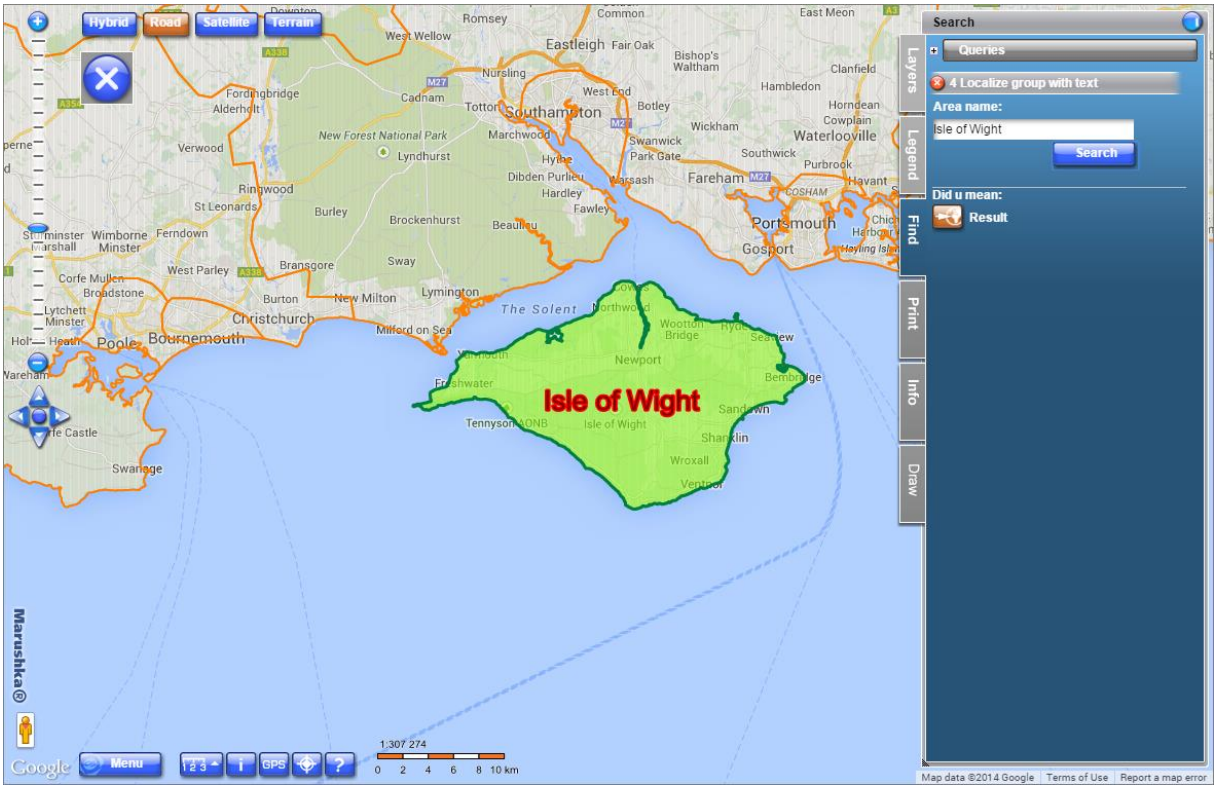


Fig 4: Result of the query ***4 Localize group with text***



4 A Brief Description of the Example in MarushkaDesign

Localization queries are queries for locating element, or territory. The result of localization queries is **an area** (limiting rectangle) in which after evaluation of query the window shifts, and **form layer** highlighting localized element or a group of elements, which is displayed as a query result. Each localization query may contain several parameters. The query parameter can be a simple text object or may be dependent on code list entries. In the second case, it is possible to use the search suggestion, which offers result based on typed text. In the web publication are queries displayed on the tab "*Search*".

The query may contain one (or both) of two queries *List of values* and *List of static values*, which are used as auxiliary queries for localization queries. These auxiliary queries define the permitted values that are on input specified as user-defined parameters ~ i ~ ... ~ j ~, where i ... j are indexes of order of the parameters listed in `QUERY_PARAMETERS`. Parameters may be also in a cascade…see example *Cascading list of values*.

A) Localization query general properties:

LayerName – the form layer name to which is the given localization query assigned, if not set, just `RANGE` is located

QueryBuf - maximum number of results per query, the default value is 1

QueryParameters – definition of parameters

SlqStmtTemplate – SQL template of query, the query must contain ("return") the following columns:

`XMIN,XMAX,YMIN,YMAX` – limiting rectangle (`RANGE`)

`ID` – id of the element, which should be highlighted after localization (if `ID` is `NULL` or if it is not a number, just the `RANGE` is located)

`LABEL` – description of localized element

It is also possible to define pseudo columns `SET_PARS` for each record.

The resulting limiting rectangle for each record is `XMIN…XMAX`.

The resulting form layer for each result: *Name = LayerName to WhereClause* is assigned *condition id = resulting id*, highlight is generated by SQL:

```
(SELECT * FROM LayerName WHERE id = 'resulting id')
```

B) Group localization query general properties:

LayerName – is not used and it is not necessary to define it

QueryBuf – ignore, because the result is always zero or one

QueryParameters – definition of parameters

SlqStmtTemplate – SQL template of query, the query must contain ("return") the following columns:

`XMIN,XMAX,YMIN,YMAX` – limiting rectangle (`RANGE`)

`LABEL` – optional, if specified, the value from the first row is used, otherwise is used the first row of the table.

System columns specifically for SQLite are: `ID, GEOM`, **these columns are different for other data stores!!!**

It is also possible to define pseudo columns `SET_PARS_` for each record.

The resulting limiting rectangle is calculated as `MIN(XMIN)…MAX(XMAX)` from all the results of the localization query.

The resulting formal layer of the result is in the form:

`SELECT * FROM (SlqStmtTemplate with all the parameters substituted)`

This example contains a SQLite database with two form layers. Form layer **GS_TABLE** displays each country **as one element** (row in the table). The geometry is represented as a COLLECTION of multi polygons. While the form layer **GS_TABLEPARTS** displays for each country from 1 to n polygons (rows in the table).

In the query library are created 4 localization queries and 4 lists of values for them.

1) Query ***1 Localization*** is a simple localization query that returns a single record from the database. Specifically, the name of the country in the United Kingdom, in the table GS_TABLE is this column called NAME2.

The whole query looks as follows:

```
SELECT xmin-25000 XMIN, ymin-25000 YMIN, xmax+25000 XMAX, ymax+25000 YMAX, id ID, NAME2 LABEL FROM gs_table WHERE NAME2 like ~(string)1~
```

The first part of the query includes a definition of limiting rectangle `XMIN,XMAX,YMIN,YMAX` (so called `RANGE`), `ID` – `ID` of the element to be highlighted after the localization (if the `ID` is not specified, only the `RANGE` is localized), `LABEL` – description of the localized element.

Part of the query after `WHERE` includes a parameter definition. In this example there is only one parameter that is to be added to the property *QueryParameters*. Parameters are indexed from 1; you can change their order by moving them with the left mouse button.

Query actually returns a new form layer that is highlighted in purple outline with pink fill (Fig 1). This was accomplished by setting property *Symbology* in the query. To display symbology it is also necessary to fill property *LayerName*.

Resulting form layer executes:

```
SELECT * FROM LayerName WHERE id = „resulting id"
```

The query includes a list of values with a search suggestion, country names are type `STRING`, and therefore they contain only letters. Tips for searching.

2) Query ***2 Localize polygons*** is also an example of simple localization, and it presents the difference between localization and group localization. Localization allows you to search 0 to N results, while the group localization allows you to search just 0 or 1 results.

Individual polygons are displayed as separate results. The difference with other types of localization is that it is possible to browse between single polygons of the query result by clicking on  in the tab "Find" below the search dialog box for entering the name of the group.

The whole query looks as follows:

```
SELECT xmin-50000 XMIN, ymin-50000 YMIN, xmax+50000 XMAX, ymax+50000 YMAX, id ID, ID||' '||name2 LABEL, GEOM FROM GS_TABLEPARTS WHERE name2 like ~(string)1~||'%' order by NAME2 asc
```

The result of query is a set of polygons with the name of the country with the same name (NAME2).

The query returns a set of elements according to the specified group name (column NAME2). Results are offered by element ID that is concatenated with a column value NAME2 and are alphabetically sorted. The results are highlighted with a yellow outline with pale yellow fill (Fig 2).

The query includes a list of values with a search suggestion, therefore there are applied Tips for searching.

The maximum number of results is returned for results: Orkney Islands, Shetland Islands and Eilean Siar, so it is recommended to search these countries.

3) The Query ***3 Localize group*** is an example of group localization.

This query is executed above the table GS_TABLEPARTS. Query finds a group of elements with the given property and displays them as one result. In the table GS_TABLEPARTS the query finds all the polygons of the given country, unites their limiting rectangle, moves the resulting window to that area and displays all the polygons of the country as one result (Fig 3).

The whole query looks as follows:

```
SELECT xmin-25000 XMIN, ymin-25000 YMIN, xmax+25000 XMAX, ymax+25000 YMAX, id ID, NAME2 LABEL, geom GEOM FROM GS_TABLEPARTS WHERE name2 like ~(string)1~
```

The query is almost identical to the query *1. Localization*, in addition, it contains column GEOM and ID, that makes it possible to localize group (XMIN…YMIN, ID, GEOM are system columns in SQLite).

The resulting form layer for highlighting then executes, in the case that the parameter 1 has value "Highland":

```
SELECT * FROM (SELECT xmin-25000 XMIN, ymin-25000 YMIN, xmax+25000 XMAX, ymax+25000 YMAX, id ID, NAME2 LABEL, geom GEOM FROM GS_TABLEPARTS WHERE name2 like „Highland“)
```

The query includes a list of values with a search suggestion, therefore there are applied Tips for searching.

4) Query ***4 Localize group with text*** is group localization. It also generates geometry with texts/descriptions to the returned polygons.

The whole query looks as follows:

```
select xmin-50000 XMIN, ymin-50000 YMIN, xmax+50000 XMAX, ymax+50000 YMAX, id ID, NAME2 NAME, geom GEOM, null SET_PARS_POINT_FROM_CORG, null SET_PARS_TEXT, '255 0 128 64' SET_PARS_RGBCOLOR, '128 128 255 0' SET_PARS_RGBFCOLOR FROM GS_TABLEPARTS WHERE name2 like ~(string)1~

UNION ALL

select xmin-25000 XMIN, ymin-25000 YMIN, xmax+25000 XMAX, ymax+25000 YMAX, id ID, NAME2 LABEL, geom GEOM, '0 0 7' SET_PARS_POINT_FROM_CORG, NAME2 SET_PARS_TEXT, '255 255 0 0' SET_PARS_RGBCOLOR, '255 128 0 0' SET_PARS_RGBFCOLOR FROM GS_TABLE WHERE name2 like ~(string)1~
```

In this SQL expression is the most important command `UNION ALL`, which joins both queries. It is important to remember that associated queries should have the same columns (the resulting values, however, does not matter).

The expression includes pseudo column `SET_PARS_POINT_FROM_CORG`, which creates a text from a point element and `SET_PARS_TEXT` that sets text of the point element. In the first part of the expression (before `UNION ALL`) is for these pseudo columns set a value `NULL`. In the second part are for these elements set specific values. It is necessary to enter the same pseudo columns, albeit with the `NULL` value, otherwise, the query would not work.

The query also includes pseudocolumns `SET_PARS_RBGCOLOR` and `SET_PARS_RGBFCOLOR`, these set line color int the first part of the query, respectively polygon fill of localization query result. In the second part of query these pseudocolumns set text outline color, respectively text fill color (Fig 4).

The query includes a list of values with a search suggestion, therefore there are applied Tips for searching.

Tips for searching: using search suggestion can enter just the first letter (or multiple letters) and for this are offered occurrences in the database. Relatively large quantity of the countries begins

with the letters a, b, d, n, m, so it recommended to start with entering one of these letters. Of course it is also possible to directly enter the full name of the country, such Scottish Borders, London or Eilean Siar.