Update Query in Marushka Design

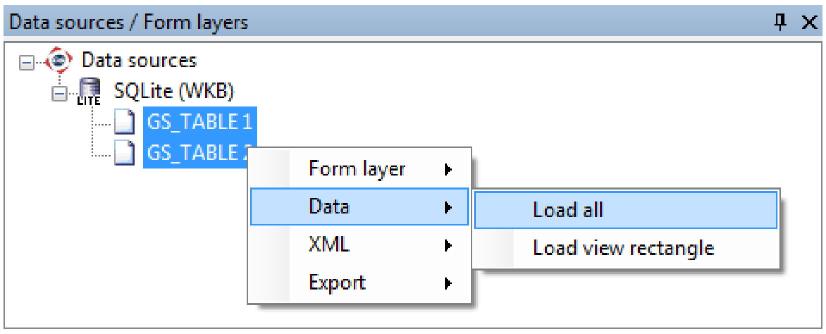
CONTENTS

1 Aim of the Example

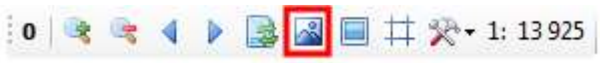
In this example we will demonstrate setting the Update query in MarushkaDesign. This example was created in version 4.0.1.0, so it does not have to be compatible with older versions

2 Working with example

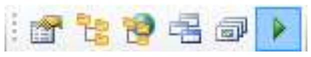
- Unzip the **Update_EN.zip** into **c:\MarushkaExamples\** folder. The target folder must be respected due to interconnection of paths with the project. In the case of placing the files in the different folder, it would not be possible to work with an example.
- Open the project **Update_EN.xml** in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



- In map window choose „Fit all“:



- Launch the local web server:



GEOVAP

3 Dialog Box Sample

Fig 1: Sample map window in the local web server

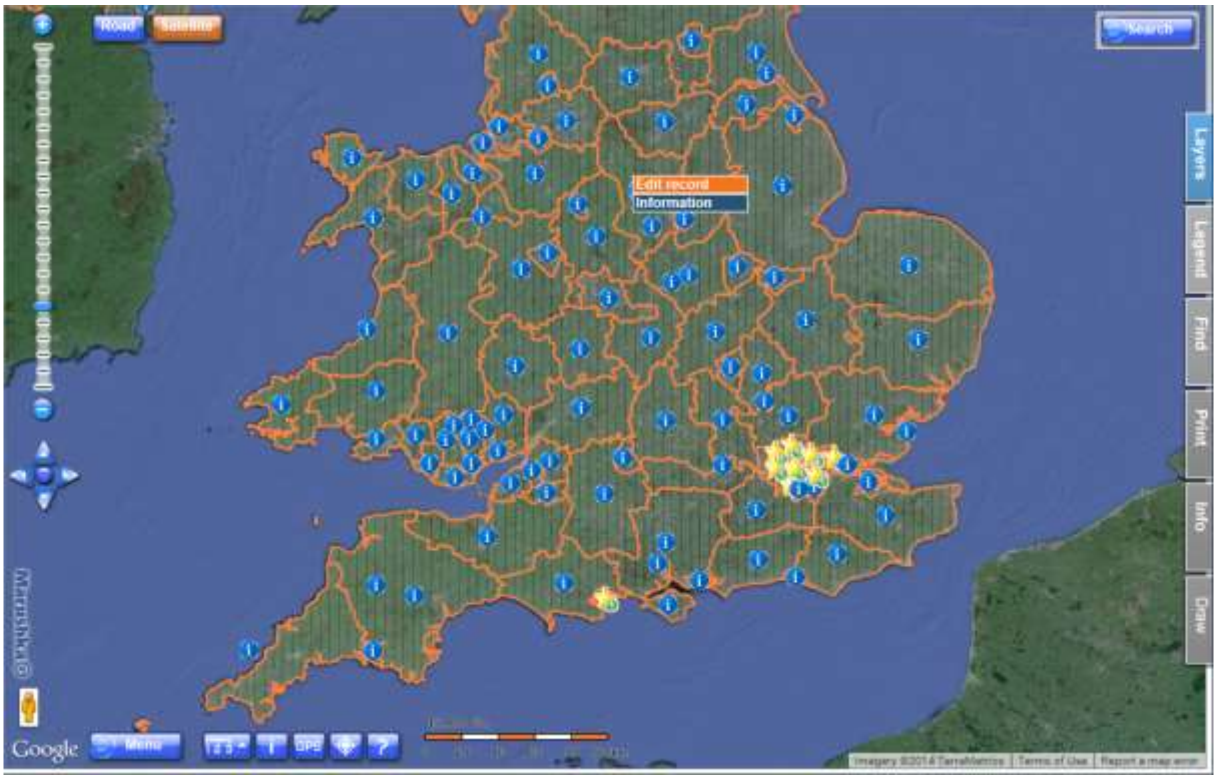
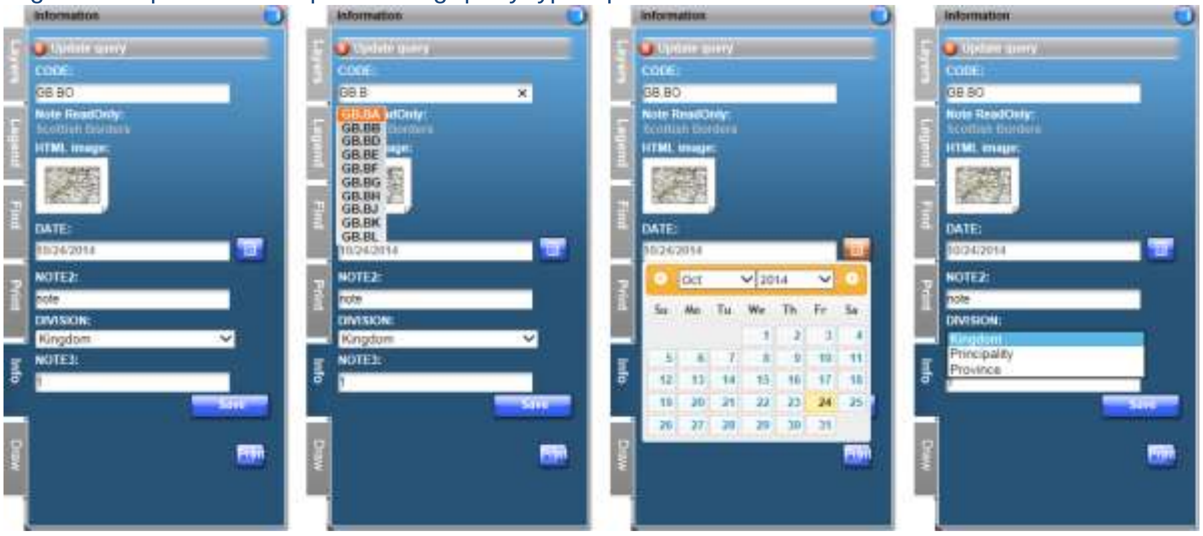


Fig 2: Examples of data update using query type Update – ***Edit record***



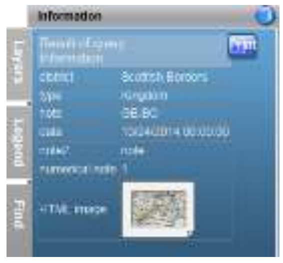
CODE: GB.BO
Note ReadOnly: Scottish Borders
HTML image:
DATE: 10/24/2014
NOTE2: note
DIVISION: Kingdom
NOTE3: 1

CODE: GB.B
GB.BA
GB.BB
GB.BD
GB.BE
GB.BF
GB.BG
GB.BH
GB.BJ
GB.BK
GB.BL

Oct 2014
Su Mo Tu We Th Fr Sa

DIVISION:
Kingdom
Principality
Province

Fig 3: Result of an information query – ***Information***



Result of Express Information

district	Scottish Borders
type	Kingdom
note	GB.BO
date	10/24/2014 00:00:00
note2	note
numerical note	1
HTML image	

4 A Brief Description of the Example in MarushkaDesign

The test example contains a database in SQLite with on publish layer. In the data source are two form layers linked to the physical layer (database table). Data bearer is the physical layer (database table) “GS_TABLE”.

Example contains a definition of Update query, which is used to update the descriptive data of geometric elements with the possibility of using code lists. It also contains one information query.

Query ***Edit record*** includes dynamic code list *“CODE”* (see note 1) with search suggestion (see note 3), one read only item “*Note ReadOnly*” item date “*DATE*”, text item “*NOTE2*”, numerical note “*NOTE3*”and static code list “*DIVISION*” (see note 2).

For each input parameter including code list (same configuration as configuration with code lists) query must have defined items in a collection of parameters “*QueryParams*”. It is very important to define the type of these parameters. HTML client based on this definition then generates visual components that control input which user entered. (*DateTime* display the calendar, for values type double or float arranges the conversion of decimal comma to dot etc.)

Configuration of query type **UPDATE** is based on configuration of two SQL queries:

A) Initial query “InitSlqTemplate”

This SQL query is used to fill the values in the *QueryParams* item. If you need to fill the parameter with index 1, the query must return a column named by constant “1”. Columns named indexes must be in the query sorted in ascending order. Further the query must initialize all the parameters in item *QueryParams*. If you want to display in a form a read only item, name the column by the text constant that must be different from “N”. Order of read only items determines order of the column in the result. Example query:

```
SELECT CODE "1", NAME2 "Note ReadOnly", '<img
src="./obrazy/PrintWidth.png"/>' "HTML image" DATE "2", NOTE2 "3",
DIVISION "4", NOTE3 "5" FROM GS_TABLE WHERE id=~(long)ID~
```

You initialize 5 parameters and the resulting form contains 5 parameters + 2 read only.

B) Update query “UpdSlqTemplate”

It is a regular update phrase. You are approaching the values via tilde and index parameter. It is very important to use type casting of parameters, if you do not use type casting, you can risk SQL injections. Example query:

```
UPDATE GS_TABLE set NOTE=~(string)1~, DATE=~(datetime)2~,
NOTE2=~(string)3~, DIVISION=~(string)4~, NOTE3=~(int)5~ where
id=~(long)ID~
```

Possible values used to type casting:
string – the text value - `~(string)1~`
int – integer value - `~(int)1~`
long – integer value - `~(long)1~`
decimal – numeric value - `~(decimal)1~`
double – numeric value - `~(double)1~`
datetime – the date type value - `~(datetime)1~`
i*dlist* – is used only when you need a condition of type ID in (id1,id2,id3), parameter in QueryParams is defined as string, and can have values of characters of english alphabet and numbers - … `where id in ~(idlist)1~`

The example includes an information query called ***Information***. This is the standard information query displaying aliases of columns of table GS_TABLE and values contained in them.

Note 1: *Dynamic code list* – is an auxiliary query for evaluation of the given values in the parameter of the query based on the query. This query returns a single column, which will be offered as a list of offered values.

Example query: `select code from gs_table where code like ~(string)1~||'%' order by code asc`

Note 2: *Static code list* - is an auxiliary query to evaluate the given values in the query parameter without database query. List of static values is firmly defined in item *ListOfValues*, individual items are separated by ENTER key.

Note 3: *Search suggestion* - is activated in an item *DynamicCodeList* in the parent query and is used for, that while typing the names of items is dynamically offered a list of items with the corresponding name.