HTML Template in MarushkaDesign

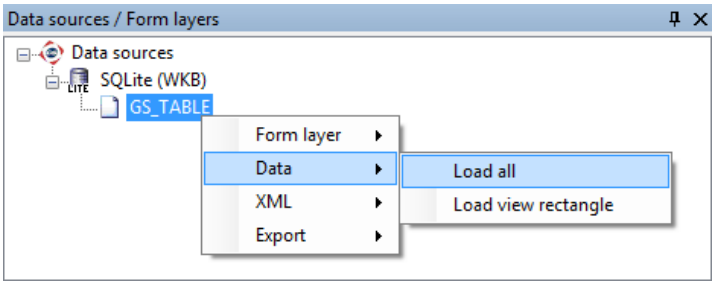
CONTENTS

1 Aim of the Example

In this example, we will demonstrate setting of the information query with HTML template 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 contents of **HTMLTemplate_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 **HTMLTemplate_EN.xml** in MarushkaDesign environment.
- Select the form layer **GS_TABLE**, in context menu choose Data – Load all:



- In map window choose "Fit all":

- Launch the local web server:

3 Dialog Box Sample

Fig 1: Information query "Information" result

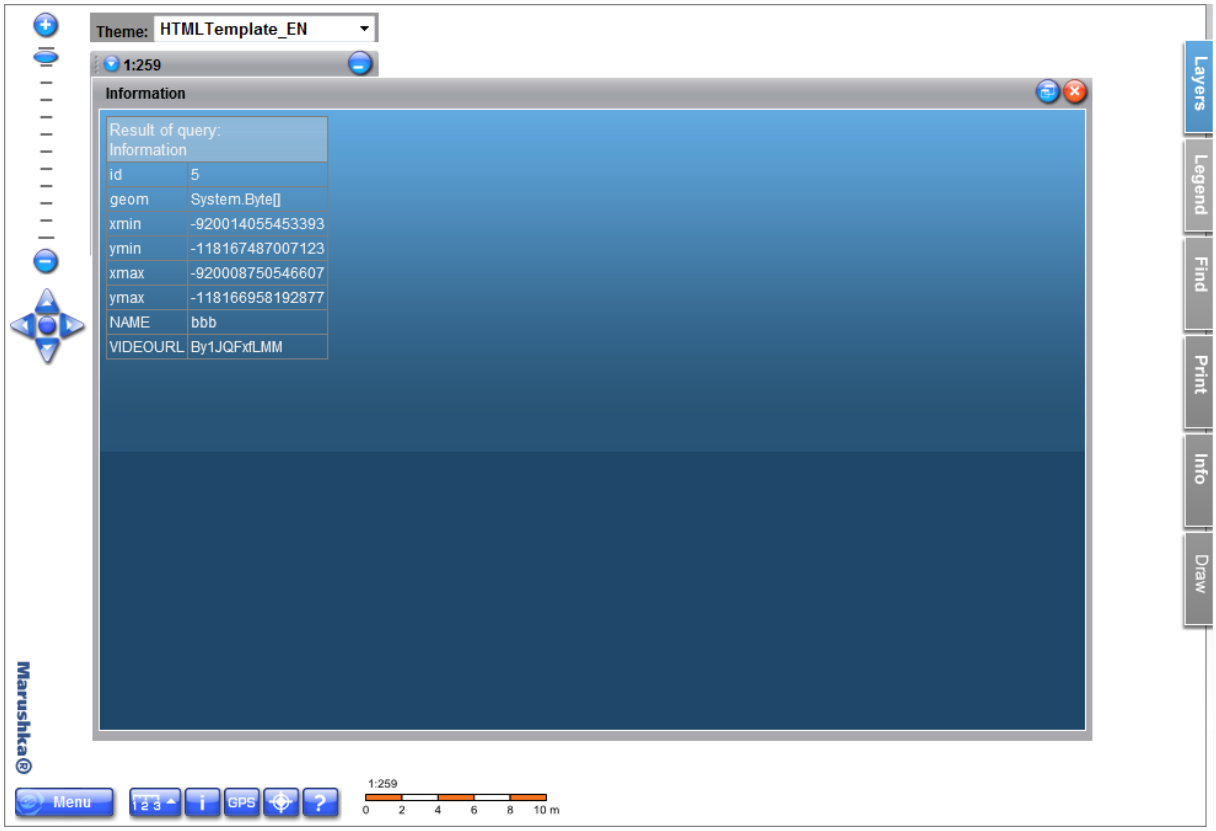
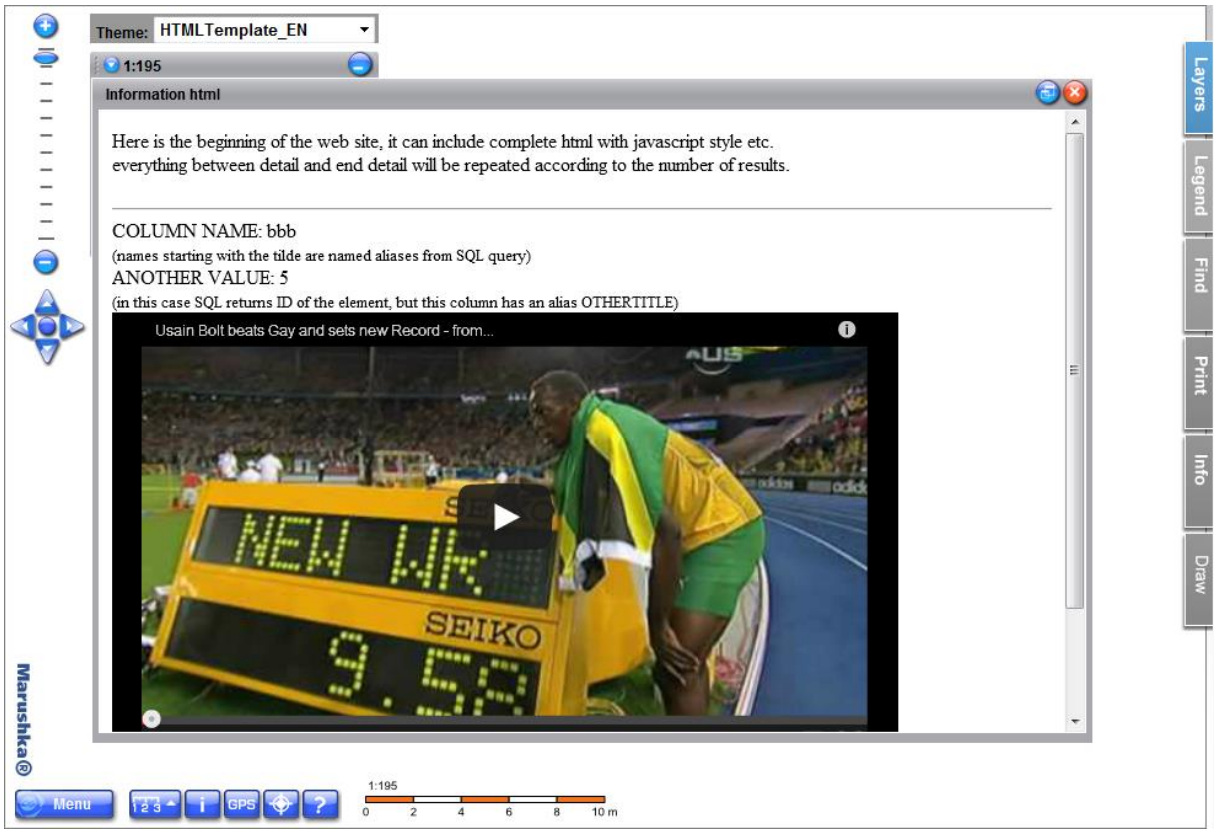


Fig 2: Information query "Information HTML" result



4 A Brief Description of the Example in Marushka Design

This test example contains a database in SQLite with one publish layer. The data source includes one form layer referring to the physical layer (database table). Data bearer is the physical layer (database table) "GS_TABLE".

The physical layer GS_TABLE has in *DBColumnsToClient* using command *SET_INFO_ICON_TEXT* set text of information icon. In this example it is specifically parameter `NAME`. Using the command *SET_INFO_ICON_LABEL* is set an additional label of the information icon, which will display after hover on the cursor, here it is specifically "*Label on hover*".

This project includes a pair of information queries.

The first query is named "***Information***", it selects and displays all the records of the element with the ID corresponding to the ID of the selected element and lists them in a popup bubble. The definition of this query is specified in the query property *SlqStmtTemplate*, its value is: `SELECT * FROM GS_TABLE WHERE ID=~(long)ID~`.

The second query is called "***Information html***", its result is displayed in the popup bubble (type of placement is defined in property *ViewStyle*).

SqlStmtTemplate property (which is a SQL query template) defines the actual information query. SQL command is in form: `SELECT <list of columns> where ID = ~ID~` where ID is the identifier of the graphic element.

In our particular case is *SqlStmtTemplate* defined by the following phrase: `SELECT name "TITLE",id "OTHERTITLE",VIDEOURL "URL" FROM GS_TABLE WHERE ID=~(long)ID~`.

Thus, from a graphical table GS_TABLE for selected graphic element choose a column NAME under alias "TITLE", column ID under alias "OTHERTITLE" and column VIDEOURL under alias "URL". If we did test this query in this form, as a result we would get the table, where on the right side are the names of the columns (aliases) and on the left side are values of the selected row.

In *ResultTemplate* property, we however have defined so called HTML template, that we can use to reformat the result of sql query, so that from the ordinary text table (in the previous query) we are able to define (typeset/text wrapping) either complex HTML document, or even generate interactive web page with functional JavaScript and CSS style definitions. The result of the information query after substitution of SQL query into a HTML template is complete and standard HTML website, so that the limits of the appearance and functionality are limited only by restrictions of HTML, JavaScript and CSS.

HTML template as already mentioned is a standard HTML website with two special tags ~DETAIL~ and ~DETAIL_END~. These tags are dividing document into three parts, namely:

a) The header is a part of the document from the beginning to tag ~DETAIL~
b) Body of the document is the part from the tag ~DETAIL~ until ~DETAIL_END~, if the SQL phrase returns more than one record, so the body of the document is repeated for each record. Body of the document can also contain special tags for alias substitution of SQL query result, these marks are in the form ~ALIAS_NAME~
c) The footer is a part from ~DETAIL_END~ until the last character of the template.

The actual substitution is performed so that the resulting HTML site is defined by the composition of the header of the document.

Furthermore, for each row of the result of SQL query is joined one body of the document (i.e. the part between tags detail and end_detail). For each column, with the alias "COLUMN_ALIAS" the SQL result document is then performed substitution. That is, that in the body of the document will be replaced string ~COLUMN_ALIAS~ for the current value of the processed row. If text of the string ~COLUMN_ALIAS~ cannot be found, nothing happens and the substitution is not performed for the column.

Subsequently, the document footer is attached. In the document header can be also defined tags ~COLUMN_ALIAS~, in this case is done substitution of the values of the first row of the result of SQL query.

In our current case, the body of the document contains the following tags ~TITLE~, ~OTHERTITLE~ and ~URL~, these tags are replaced with the values of the result. The resulting HTML displays to the appropriate element following records: COLUMN NAME: in this is the value of alias ~NAME~ (specifically the value bbb), OTHERVALUE: It is the value of alias ~OTHERTITLE~ (specifically the value 5) and iframe with URL to video (specifically value By1JQFxfLMM, which is the key part of the code of URL site).