Data use and management

Data use and management

In the article Image management (In de Quaestor Knowledge Browser: Quaestor>Topics>Technical documentation) and QBinaries, it is described how to create, maintain and access a collection of images or documents that are presented when a particular parameter or value is in focus in the Knowledge Browser or Workbase. In this article we will leave these mechanisms behind and zoom in on the possibility to create (binary) databases and use them (by performing selections) during the modeling process.

A Dataset of a knowledge base may contain a set of objects with all kind and different types of data. During a modeling session, selections from these objects can be made for a variety of purposes. The mechanism is illustrated with the selection of ship hullform data.

Example 1, Use of data in the dataset with an |OPTION| script:

Assume you want to store a hull form offset table in a parameter HullForm$ because it is needed as part of an input file for software to calculate hydrostatic properties.

HullForm$ is determined by means of a relation:

HullForm$ = HULLDATA#.HullForm$

HULLDATA# is the selected input set for the calculation and is defined by an |OPTIONS| script represented as a Telitab set in its data slot of the parameter:

```
Available parent ships:
  |OPTIONS|
   8
   "CAPTION1"     "Ship type"
   "CAPTION2"     "Project number"
   "CAPTION3"     "Loading condition"
   "OPTION1"      "SHIPTYPE$"
   "OPTION2"      "PROJECT$"
   "OPTION3"      "CONDITION$"
   "HullForm$"    ""
   "DATA#"        "ParentShips"
  |
```

Let us explain what you see in more detail:

The first line you see is simply an explanation (and not required).

The script between |OPTIONS|….| describes how, what and where to get the Telitab set you will refer to as HULLDATA#.

This particular script describes three selection steps through a dataset,
"Ship type" (represented by the data connected to parameter SHIPTYPE$),
"Order number" (represented by the data connected to parameter PROJECT$) and
"Loading condition" (represented by the data connected to parameter CONDITION$).

The "CAPTION" part is the presentation of the questions, the "OPTION" part is the link to the data in the data set that is used.

Please realise that the last item, "DATA#" "ParentShips", defines the reference for HULLDATA# to the actual total data set. This is the data set for the selection as described by the script. In this case the data is available in the ParentShips object.

During calculation the system will look for this object in the current solution. If it is not found, it will look in the Dataset of the project and if it is not found in the project either, it will look in the Dataset of the knowledge base.

The line "HullForm$" "" in the script tells the system that at least a DETERMINED value of HullForm$ should be available to allow selection of a case. This means, that the result of the selection cannot be empty values for HullForm$, therefor Quaestor will pre-filter all these cases out of the total dataset. The selection of HULLDATA# will contain all other DETERMINED values available in the selected record. Also SHIPTYPE$, PROJECT$ and CONDITION$ must be DETERMINED (thus available) to allow a selection.

Optional use:

By using the QUERY# function with this syntax: *QUERY#(0, Arg_1$[:ParLab_1$],,,,Arg_n$[:ParLab_n$])* for HULLDATA#, you can make sure that already available parameters will be used to pre-fill the options.

For instance when the ship type is already defined earlier in the solution, or is available as starting data in the dataset, you can add the relation *HULLDATA# = QUERY#(0, SHIPTYPE$)* to make sure that OPTION1 already uses the value for SHIPTYPE$.

Example 2, Use data in the data slot of the parameter with |OPTIONS| script:

As alternative method to select data, in stead of making a reference to an object as done with "DATA#" "ParentShips", you can include the data in the |OPTIONS| script of the parameter. This is done in the following example

```
@SUMMARY
  @PICTURE Required options for xxxx method
  |OPTIONS|
   12
   "CAPTION1"   "Type of ship:"
   "CAPTION2"   "Type of superstructure:"
   "CAPTION3"   "Trial loading condition:"
   "CAPTION4"   "CX table:"
   "OPTION1"    "SHIPTYP"
   "OPTION2"    "SUPERSTR"
   "OPTION3"    "LOAD"
   "SUMMARY"   "Selected combination of ship type, loading condition and superstructure on trial"
   "SUMMARY1"   "Selected ship type:"
   "SUMMARY3"   "Selected loading condition on trial:"
   "SUMMARY2"   "Selected type of superstructure on trial:"
   "SUMMARY4"   "Selected wind table on trial:"
   5    "OPTION" "SHIPTYP"       "SUPERSTR"       "LOAD"   "OPTION4"
   "1"  01060134 1              6              1       "table 1:twind1"
   "2"  01070135 1              7              1       "table 2:twind2"
   "3"  01080336 1              8              3       "table 3:twind3"
   "4"  01090337 1              9              3       "table 4:twind4"
   "5"  23090945 23             5              9       "table 12:twind12"
   "6"  24090946 24             5              9       "table 13:twind13"
   "7"  25050138 25             5              1       "table 5:twind5"
   "8"  25050339 25             5              3       "table 6:twind6"
   "9"  26050340 26             5              3       "table 7:twind7"
   "10" 27050941 27              5              9       "table 8:twind8"
   "11" 28100942 28             10             9       "table 9:twind9"
   "12" 28110943 28             11             9       "table 10:twind10"
   "13" 28120944 28             12             9       "table 11:twind11"
  |
```

Let us explain what you see in more detail:

@SUMMARY and @PICTURE are attributes, they manipulate the behaviour of the knowledge based system comparable to all the Slots&Properties. Using the @SUMMARY attribute you tell the system to display an overview of the total selection made, based on the dialog through the included data. @PICTURE indicates that pictures should be displayed of which the references are given in the included dataset.

Again "CAPTION" and "OPTION" are the presentation and reference to the data. "SUMMARY" defines how the summary should be presented after the selection is made. Please note that the result corresponding with the summary number will be placed behind the text. Moreover, the order as they are presented in the scripts is the order of de lines as they are presented in the summary

By including "OPTION" in the the table, the database selection returns only a single value, being the selected case value of "OPTION" instead of all values related to the selection.

As mentioned the @PICTURE attribute defines that relevant picture information can be presented through a references to these images. For "OPTION4' this has been defined. For example, table 1:twind1 refers to an image "twind1" in QBinaries object in the Dataset (QBinaries is a reserved Quaestor object that can be used to store all knowledge base fixed binary data, see Image Management in Quaestor>topics>Technical documentation>).

Please note that, if another object is used as data source (as is done in Example 1) referring to images in the QBinaries will only work if the parameters used in the refered object have Value:Label as string values and a @PICTURE attribute in their dataslot.

Example 3, Use of binary data in the dataset with an |OPTION| script:

In the following we will show how to manage and access a set of binary objects. In this example we will enable the selection of components in a Rhinoceros configurator, a typical CAD application that can be controlled by scripts that are generated by Quaestor. The problem is related to the selection of component models on the basis of some selection criteria.

Similar to the procedures followed to create an image database in the Dataset.QBinaries object, it is possible to create any object in the dataset containing binary data using other than the reserved QBinaries, QBinary and QBinaryID parameters.

Please note that, in order to make any string parameter suitable for the storage of binary data, the only necessary action is to include the @BINARY attr ibute in the Data slot of the frame. A dataset can be created with the "Parameter to Dataset" menu option. Binary data can be included through the "Dataset>Include Binary Data" menu option after which a file dialog is invoked on the current active project directory and removed by means of the "Dataset>Remove" menu options (shortcut is Ctrl-B). Please note that only parameters of the String type can be used to store binary data.

In this example an object Dataset.RhinoComponents has been created containing the following parameters:

1. Component$: a string parameter with the @BINARY attribute. In each value cell of this parameter in the Dataset.RhinoComponents an arbitrary file can be stored (see above). If the file is double-clicked or F2 is pressed, it is saved in the currently active project working directory

and the associated program is started (in this case Rhino). If the file is changed in the associated tool and saved on the same location under the same name, Quaestor restores this modified file in the selected cell.
2. Type$: an existing parameter defining the type of component, e.g. a "Hull form", "Crane" etc. This is the first selection key.
3. Name$: the denominator of the component, e.g. Container vessel or Gantry. This is the second selection key.

Quaestor employs an internal database selection mechanism that can be defined through an |OPTION| script. In the following we will briefly explain its principles. In order to enable the selection of a hullform, the following |OPTION| script is included in the data slot of the parameter ComponentFile$:

```
@LOCAL                          'parameter is local to object
  @SUMMARY                          'generates a summary after the selection
  |OPTIONS|                       'Tag for selection script
   6
   "CAPTION1"   "Give Component Type"    'Caption of first selection list
   "CAPTION2"   "Give Component name"    'Caption of second selection list
   "OPTION1"    "Type$"                  'Type denominator of component
   "OPTION2"    "Name$"                  'parameter with first selection key
   "OPTION"     "Component$"             'parameter with second selection key
   "DATA#"      "RhinoComponents"        'Reference to object containing data
  |
```

Note: The text after ' is comment **please do not use** it in the knowledge base.

Although the object RhinoComponents contains both Name$ and Type$, it will not ask for the type if a value of Type$ is already available on the location at which a hull form file is required. So, if Type$="Hull form", the option dialog will not present the "Type$" options list and only leaves the choice for cases fulfilling Type$="Hull form".

This will also work for a larger number of selection keys of which any key with a DETERMINED value will not be presented in the selection dialog and are used as query arguments for the following sub selection. This is the way in which you can navigate and select in large collections of coherent options. An example is the QSTAP knowledge system for ship sea trial analysis in which a large number of ship- and condition dependent methods are to be selected from.

The above selection mechanism is enhanced with a mechanism that will immediately export any binary data stored in the final selection result. This implies that in the above example the binary data contained in the selected Component$ case will be exported to the currently active working (project) directory. There can be more than one binary datasets included in a selection; all will be exported assuming they will have unique names. In this way, you can ensure that particular data is available to any satellite program (in this example the CAD program Rhinoceros) in which it is addressed by a Quaestor-generated script.

The optional @INITIFNOFILE attribute…
In the event that a knowledge base in which the above mechanism with binary data is applied, is copied and a solution is restarted using data already selected in the above described manner, some required files may not be present at the moment of executing a particular script. This may lead to some confusion that can be avoided by adding the @INITIFNOFILE attribute to parameters representing binary objects (such as the above example Compon entFile$). On restarting a solution using an Input object, parameters with the @INITIFNOFILE attribute will be checked on the availability of the file (of which the name is contained in the parameter) on the current project directory. If not available, the value is intialised and the selection dialog is re-invoked. After selection, the file is exported again.

The Input object is another reserved Quaestor object you can define together with the reserved string parameters Value and Parameter. By doing so, you will have an input recorder for object-related input, a typical requirement of configurator applications.

Selection of a single binary object...

In the event that a parameter represents a single, unique binary object, it is not necessary to use the above selection mechanism. In that event, the solution is to include the object in the frame as embedded object using the "Include Binary" in frame menu option (removal through "Remove" from frame menu option). If the value given to that parameter during a session equals the file name of the embedded object (without extension), the object is immediately exported under that file name to the current project working directory. This is an elegant and simple way to the storage and making available of particular files that are unique and can therefore be associated to a parameter.