Image management

Image Management in Quaestor

Quaestor has the ability to deal with- and present binary information, such as pictures, in the knowledge browser and Workbase using the QBinaries obj ect. In the following, we briefly describe the current possibilities for using binary data in the knowledge base and in solutions focussing on the use of images (Click on QBinaries for a detailed description how to create and fill the QBinaries object).

Please note that by default all Quaestor constants are given the @HIDE attribute. So please select the "Show hidden data" option in the "Modeller" tab of the Options window.

To show pictures based on selections in the Knowledge Browser or Workbase, binaries can be coupled to a frame or parameter value in the following ways:

Method 1, direct reference to a file:

The first method is to refer to image files accompanying a knowledge base.

Any knowledge base has its own file structure under the knowledge base directory (by default, this will be under My Knowledge\Kbs_KnowledebaseName in which KnowledebaseName is the name of the current knowledge base, without extension.
The directory tree under this level will contain a directory Html\images in which the standard set from the Html\images directory present on the Quaestor program directory will be present.

In the My Knowledge\Kbs_KnowledebaseName\Html\images directory one can store a collection of images related to this knowledge base which can be referred to by including in the parameters' data slot or (object related) reference an attribute:
@PICTURE:FigureX.gif

FigureX.gif needs to be present in the above mentioned directory. If not, the standard background picture will be shown in the manner described above.

No use is made of the QBinaries picture database described above. The knowledge engineer is solely responsible for ensuring the availability of the images in the ._KnowledebaseName directory.

So, by renaming the knowledge base or by copying the knowledge base to another location, the connection with the original ._KnowledebaseName\Html\images directory is lost and the images will not be automatically available to the copied/renamed knowledge base, unless the knowledge engineer has copied them to the new (correct) location.

Method 2, reference to the binary included in the frame:

It is also possible to include an image (object) in a frame by the menu option Include Binary in Frame.

So if you include FigureX.gif and you write in the data slot: @PICTURE:FigureX.gif Quaestor will first save the binary and than show it as the picture.

This method can be followed if the images unique purpose is to be presented with that single parameter. As soon as you need that image on more than one location, it will introduce (undesired) redundant data in the knowledge base.

Method 3, reference to a QBinaryID in the QBinaries object:

If we use the QBinaries database, we ensure the availability of a collection of images independent from the name or location of a knowledge base. In this case we have several ways in which we can make an image appear. By introducing the attribute:
@PICTURE::5

in a frame, the image Figure 5.gif will be presented as soon you reach this frame in the knowledge browser (in the knowledge base presented in fig.1). Please note the double semicolon in@PICTURE::5 which is used to distinguish the value 5 from a file name. in this way, it is possible to show a single image on multiple locations without redundancy. Based on the value behind the double semicolon, the QBinary image value in the first record in Dataset .QBinaries with a QBinaryID value of 5 will be presented.

Method 4, reference to a parameter containing a binary or a reference to a binary:

Instead of immediately referring to a file one can refer to another parameter:
@PICTURE:Lpp

If the parameter Lpp contains a picture reference like the methods described above, this reference is inherited by the frame containing the @PICTURE: Lpp attribute.

Copyright © 2022, MARIN

This method will work in combination with all methods 1-3.

Method 5, reference to the QBinaries object based on a parameter value equal to QBinaryID:

Instead of a Knowledge Browser-centered approach to presenting images/objects, one can connect images to parameter values.

By including only the @PICTURE attribute, so without file name, parameter name or value, the current value of this parameter will be used as the QBinaryID search key.

Please note that in the event of a VALUE parameter the formatted value of the parameter will be used as search key. Thus, a value of 2 will be provided as 2.00 if a Fixed Format with 2 decimals (FF2) is used. **Note that a QBinaryID string of 2 will not fulfil this argument value.** The used value will be the parameter value of the current solution/dataset/object/case. Parameters not having the single @PICTURE attribute will not be linked to images in this fashion.

Connecting values to images as described under Method 5 is a useful concept with two limitations, however:

1. if you want to present the same image for multiple values, you need to introduce multiple instances of an image in Dataset.QBinaries;

2. Generic parameters can have object dependent references, each containing sets of option lists that are solely related to particular objects. This implies that one wishes to relate images not only to parameter values but also to the object in which the value exists.

Method 5a, reference using a Boolean expression for QBinaryID:

The first limitation implies that if you want to present the same image for multiple values you need to introduce multiple instances of an image in

Dataset.QBinaries, one of each value that may occur. In order to get around this limitation, it is possible to provide a Boolean expression for QBinaryID instead of a single value. The expression need to be preceded by @. Examples are:

@ParX=1 OR ParX=2 OR ParX2=6

@ParX=>1 AND ParX<6

In fig.1 the constraint @a>3 AND a<9 is connected to Figure 6.gif. For any value >3 and <9, the image Figure 6.gif will be shown. In the constraint all intrinsic Quaestor functions can be used including brackets. However, functions using objects are not evaluated.

Method 5b, reference using a script to refer to object dependent images:

The second limitation of Method 5 is related to generic parameters. Generic parameters have object dependent references, each containing e.g. sets of option lists that are solely related to particular objects. This implies that one wishes to relate images not only to parameter values but also to the object in which the value exists, i.e. it may become necessary to relate ImageA to value X of Par_i in object Obj_j and !ImageB to value X of Par_i in object Obj_k .The way to do this is to use the lists in the object-related references as in the following example from a parameter Switch.

```
><
=misc= Switch mode
0Switched in parallel
=Diesel.LTCooler=
-1Switched in series cold:12
1Switched in series warm:13
@PICTURE
=Diesel.HTCooler=
-1Switched in series cold:14
1Switched in series warm:15
```

Copyright © 2022, MARIN

@PICTURE

=InjCooler=

-1Switched in series cold

0Switched in parallel

1Switched in series warm

In this example, in any instance of the object Diesel.LTCooler the reference

-1Switched in series cold:12

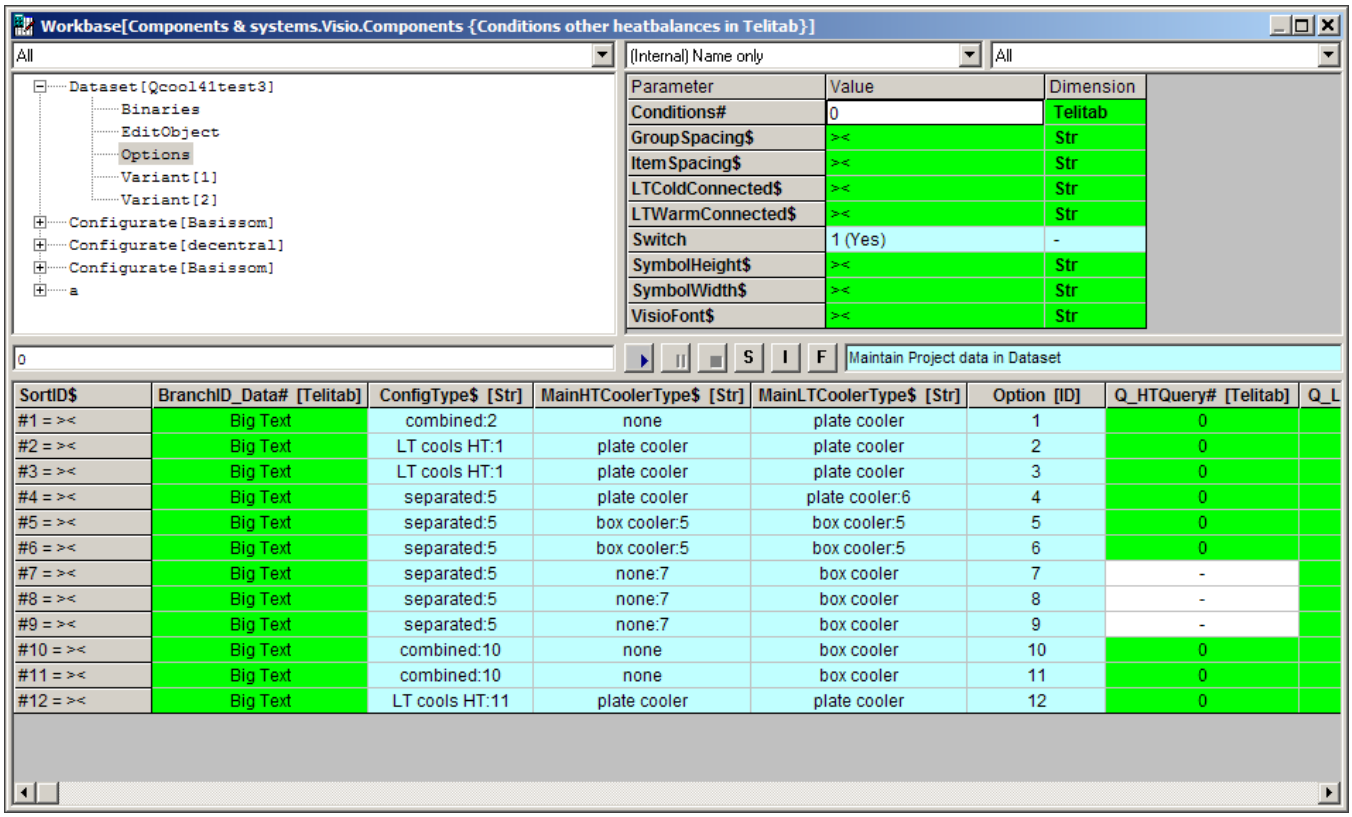
1Switched in series warm:13

@PICTURE

will be available. In this object, for a Switch value of -1 the Dataset.QBinaries.qBinary with QBinaryID value of 12 will be shown in the Illustration form. For a Switch value of 1 theDataset.QBinaries.QBinary with QBinaryID value of 13 will be shown. The attribute @PICTURE is included in order to tell Quaestor that it should search for an image/object inDataset.Binaries .

In any instance of the object Diesel.HTCooler this will be picture 14 and 15 respectively. This example shows how to relate different images to the same parameter value in different object instances.

Method 6, reference using an |OPTIONS| script



In the above picture, some parameters like ConfigType$ and some values of MainLTCoolerType$ image references are included.

For instance, the fifth case value of ConfigType$ is separated:5 in which 5 is considered to be a Dataset.QBinaries.QBinaryID reference to an image. In this example, the Dataset.Options object is data source to an |OPTIONS| script for parameter OPTIONS# , a TeLiTab set in which all elements of a case of Dataset.Options will be its value. The |OPTIONS| script is discussed in a separate article Data use and management in Quaestor.

The |OPTIONS| script results in invoking a list view including a browser component that will present the referred to image per parameter that is proposed in the list view. This allows a stepwise selection of a database record supported by images that are context sensitive, i.e. for each step in the selection process, it now possible to present relevant graphical images(, spreadsheet, pfd documents, etc).

Copyright © 2022, MARIN