Innovator 11

Post-Processing

Word Documents

How to Create Tables and Hyperlinks in a Generated Word Document
Table of Contents

Overview ....................................................................................................................... 1
System Prerequisites........................................................................................................ 1
Creating Tables ................................................................................................................ 2
  Setting Markers in the Documentation Structure ........................................................... 2
    Ino.DocTitle................................................................................................................. 2
    Ino.TableBegin............................................................................................................ 2
    Ino.TableEnd............................................................................................................... 3
    Ino.TableRow.............................................................................................................. 3
    Ino.TableCol................................................................................................................ 3
    Ino.TableCell............................................................................................................... 3
    Ino.TableCaption......................................................................................................... 3
    Ino.TableList ............................................................................................................... 4
  Example........................................................................................................................ . 4
Creating a Table List ........................................................................................................ 6
Creating Hyperlinks ........................................................................................................ 6
  Operating Modes ........................................................................................................... 6
  Markers in the Documentation Structure ....................................................................... 7
    Ino.Hyperlink.Text ....................................................................................................... 7
    Example...................................................................................................................... 7
    Ino.Hyperlink.Src......................................................................................................... 8
    Example...................................................................................................................... 8
    Ino.Hyperlink.Trg......................................................................................................... 8
    Example...................................................................................................................... 8
Creating the Table of Figures ........................................................................................... 9
Setting-Up and Using the Post-Processing Command ..................................................... 9
  Customizing Documentation Templates ........................................................................ 9
    Customizing Documentation Templates in Innovator eXcellence ............................. 10
    Customizing Documentation Templates in Innovator classiX ................................... 10
  Call Options ................................................................................................................. 11
  Documentation Generation Procedure ........................................................................ 12

Overview

Innovator documentation generation can extract a wide array of information from an Innovator model and document it. Up until now, there has never been an option for adding tables in a document in Word for Windows format which can be used for arranging information entered in the document in a compact way or for creating hyperlinks to specific chapters or sections. These features are available using a Tcl script for post-processing. This script can also be used to link a Word template file with the document and create a table of figures or a table list.

System Prerequisites

- Innovator 2008 (Version 10.0) or above
- Microsoft® Office Word 2002 or above (at least Version 10.0) on the Innovator client computer
Creating Tables

Setting Markers in the Documentation Structure

The corresponding table sections need to be selected so that part of a document can be converted into tables. This selection occurs within the documentation structure using key words (please take case sensitivity into account) in the following form:

- `Ino.DocTitle <document title>`
- `Ino.TableBegin [orientation=vertical $][format specification] $ <column1> ($ …) $ <columnN>`

Format specification can contain a string in the format "column width1:…:column widthN"; the numbers are construed as percentage values.

- `Ino.TableRow <column1> Ino.TableCell
  - <Contents of column1>`
- `Ino.TableCol <column2> Ino.TableCell
  • ...
- `Ino.TableCol <columnN> Ino.TableCell
  - <Contents of columnN>`
- `Ino.TableEnd`

Please note that the key words given must also appear in the Word document upon documentation generation. You can ensure this by correctly using the settings for logical and/or localized names in the Edit/Properties dialog box of the respective documentation elements.

`Ino.DocTitle`

`Ino.DocTitle` is the title of the document. It can be entered in a documentation structure to store the post-processed document under another name. The son nodes which contain this marker are deleted from the post-processing and do not belong to the later document text. The existing document’s file names can be defined in a variable manner by evaluating the available wildcards (such `$\{name\}$`) in the documentation structure, e.g. depends on the selection during document generation. If no `Ino.DocTitle` is found in the generated document, the generated document is replaced by the modified document.

`Ino.TableBegin`

You can use `Ino.TableBegin` to define a new table within the document to be generated. The names of the table headings exist after `Ino.TableBegin` and are separated by `$` characters. You need to do this to set how many columns there are (if they are not all available) and which section in the document belongs to which column.
If `orientation=vertical` is entered, a rotated table can be created, i.e. headings are entered as contents in the first column. The format needs to be given in this case. If you want to use an automatic format, specify it as follows:

```
<column1> ($ ... ) $ <columnN>
```

Entering a question mark ? in front of the column name means that the respective column will not be included if it doesn't contain any values.

Entering a plus + in front of the column name means that this table heading will be rotated. The height of the table heading can be given in point size in front of the column title.

Example: the entry `+60:Cardinality` creates a rotated column heading "Cardinality" with a row height of 60 points. This function is only supported for horizontal tables.

```
Ino.TableEnd
```

You can use `Ino.TableEnd` to mark the end of a table. It is important that a chapter which is to appear in the documentation is selected and that it only creates one entry; this is because only the paragraph immediately after is deleted in the Word document created. Examples of suitable chapters are "Qualified Name" or "Model". The post-processing deletes the direct son node in the documentation template which appears after this marker and is not a heading.

```
Ino.TableRow
```

Each table row starts with `Ino.TableRow` and continues until the next `Ino.TableRow`. This entry also contains the table heading of the first column to be filled and the cells contents at this point, separated from each other by `Ino.TableCell` (see below).

```
Ino.TableCol
```

Each table column starts with `Ino.TableCol` and continues until the next `Ino.TableCol`. If a son node which can have its own son node is marked in the documentation structure, this column's cell contents is used from the next sub-node shown and can be formatted using wildcards if necessary. Leaf nodes which cannot have any other son nodes write the result directly in the given cell.

```
Ino.TableCell
```

The `Ino.TableCell` marker separates the column heading from the cell contents. The text between `Ino.TableCol` and/or `Ino.TableRow` and `Ino.TableCell` needs to match one of the table headings defined with `Ino.TableBegin`, so that the contents is inserted in the right column.

Background: As it is possible for the column contents to be missing (e.g. for descriptive texts), it is not possible to tell which column this information belongs to just from the order of the information generated.

```
Ino.TableCaption
```

The `Ino.TableCaption` marker is used for entering a table caption below the table.
Ino.TableList

The **Ino.TableList** marker can be used for making a table list.

## Example

Interface operations should be documented for an interface. Certain information about a particular operation’s parameters which is stored in the model should be compactly displayed in a table:

- Name, direction and default value of a parameter;
- Name and specification of the parameter type which specifies that a class is concerned here;
- Information about whether a must or can parameter is used.

Going under the assumption that the last information can be attached to the multiplicity’s lower boundary (0 \(\rightarrow\) can, 1 \(\rightarrow\) must), the documentation structure for such a table can be organized as shown in the following illustration:

---

*Extract from a documentation structure before the key words have been expanded*
The following modifications need to be made opposite a standard documentation structure:

1. An artificial hierarchy level is inserted above the son node which delivers an operation's parameters; the hierarchy level is called parameter. This becomes the heading which marks the beginning of the table in documentation carried out at a later stage.

2. The original parameter (parameter) son node has a name which defines the table headings. In this case, this is a total of 6 columns, including the name of the parameter. The "16:14:15:6:10:40" string specifies the width of the individual columns in percent.

3. The $(name)$ node (the name of the respective parameter) is adapted in such a way that it defines a new row in the existing table. The same entry also sets the output in the Name column.

4. The Direction, Lower Boundary and Default Value leaf nodes are given the key word for a table column and given a name which corresponds to one of the declared column headings in Ino.TableBegin.

5. The original Referenced type (class) son node can similarly be renamed. If parameters can be typed by data, enumeration or primitive types as well as classes, various son nodes need to be combined using an artificial hierarchy which contains key words for column definition, whereas the combined headings themselves will be suppressed.

6. Not visible in the illustration: The node for the $(name)$ descriptive text contains a filter which only lets the specification text pass. To make the table definition independent from the filtered text, the table column is not coded using $(name)$, but is definitely coded using specification.
Post-processing of a document using the documentation structure given as an example above yields the following table for an interface operation containing an input, output and return parameter respectively.

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Data Type</th>
<th>Min.</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InputParameter</td>
<td>Input</td>
<td>Long</td>
<td>1</td>
<td>17</td>
<td>The Long class wraps a value of the primitive type long in an object. An object of type Long contains a single field whose type is long. In addition, this class provides several methods for converting a long to a string and a string to a long, as well as other constants and methods useful when dealing with a long.</td>
</tr>
<tr>
<td>Return</td>
<td>Class</td>
<td>1</td>
<td>null</td>
<td></td>
<td>Instances of the Class class represent classes and interfaces in a running Java application. An enum is a kind of class and an annotation is a kind of interface. ...</td>
</tr>
<tr>
<td>OutputParameter</td>
<td>Output</td>
<td>Data Type</td>
<td>1</td>
<td></td>
<td>The Data Type class is used to ...</td>
</tr>
</tbody>
</table>

**Creating a Table List**

You can create a table list in the document using a marker in the docu structure.

**How to proceed**

To create a chapter for the table list:

» Insert a chapter that will definitely be read in the docu structure at the topmost level, i.e. Model (UML2 Model) and rename it to "Table List".

» Rename the subchapter from $\{name\}$ to "Ino.TableList".

**Creating Hyperlinks**

**Operating Modes**

You can use new markers in the docu structure to set where bookmarks and hyperlinks should be positioned in the document for these bookmarks. A bookmark has a unique name in the document which a hyperlink can refer to.

The bookmark’s name is also set with markers in the docu structure and is generally not specified. A model element’s URI is provided here as a bookmark name must be unique.
Markers in the Documentation Structure


The following scenario is used as an example over the next page for explaining individual markers. An overview of use cases is displayed in a table in a document. This table only displays the name and a summary of the use case. To now be able to access a detailed description of a use case using the hyperlink, the name of the use case is displayed in the table as a hyperlink to a section in the document with the detailed description of the use case.

Ino.Hyperlink.Text

The position of a hyperlink is the document is set using the Ino.Hyperlink.Text marker. A text which is displayed as a hyperlink in the document and can be clicked on is necessary for a hyperlink. If you enter hyperlinks in tables and want to include more than one row in a cell, you need to add a $ in front of Ino.Hyperlink.Text.

[{$]Ino.Hyperlink.Text <Hyperlinktext>

Example

In the example shown, the Ino.TableRow use case Ino.TableCell Ino.Hyperlink.Text $(name) heading changes the name of a use case to a hyperlink and displays it as such, as can be seen in the following illustration.
HowTo Post-Processing Word Documents

Ino.Hyperlink.Src

The name of the bookmark which should be used as a hyperlink’s target is entered using the Ino.Hyperlink.Src marker; this needs to directly follow the Ino.Hyperlink.Text marker.

Example

The use case’s URI is generated directly after the use case’s name which shows the hyperlink itself. This is done using the heading or the Ino.Hyperlink.Src marker.

The hyperlink contains the bookmark as the target with the name from the use case’s URI.

Ino.Hyperlink.Trg

A bookmark’s position in the document is specified by the Ino.Hyperlink.Trg marker. The bookmark’s name is taken from the nodes chosen for this. The name is also typically taken from the model element’s URI.

The bookmark is set in the Microsoft Word document where the previous paragraph’s marker is.

Example

In this example, a bookmark with a name from the use case’s URI is set at the point of the paragraph with the use case’s name (in the illustration 2.1. $(element_type "$(name)"), as this paragraph is directly before the Ino.Hyperlink.Trg marker.
Creating the Table of Figures

If the –pi call parameter is entered, a table of figures is created. If this option is set, a caption is generated for each figure; the heading above is the inserted into the caption below the figure. If this caption is not numbered, the original heading is deleted; if it is numbered, a copy is created.

A title called Table of Figures is then searched for in the document and this chapter’s contents is then replaced by a table of figures created by Word.

How to proceed

To create a chapter for the table of figures:

» Insert a chapter that will definitely be read in the documentation structure at the topmost level, i.e. Model (UML2 Model) and rename it "Table of Figures".

Setting-Up and Using the Post-Processing Command

Customizing Documentation Templates

To create tables and hyperlinks from the markered document sections, the generated Word document needs to be reworked. You can either do this manually using the command line or automatically using a post-processing command.

The documentation template which is used for creating the Word document should be customized for automatic transformation.
Customizing Documentation Templates in Innovator eXcellence

How to proceed
To use `worddocu_table.tcl` as the post-processing command in Innovator eXcellence:

» If applicable, jump to the Documentation view in the configuration editor.

» Select the desired documentation command and select Edit>Properties…
   The dialog box of the same name appears.

» Jump to the Properties tab.

   You can specify a command and its arguments in the Post-processing Command group. The command is automatically carried out upon documentation creation, whereas the created output file will also be implicitly transferred as the first argument alongside the arguments entered in the Arguments field.

» In the Post-processing command field, enter:
   `inotcl -f worddocu_table.tcl`

» You can enter other arguments in the Post-Processing Arguments field.

» Click on [Apply] to adopt the settings.

Customizing Documentation Templates in Innovator classiX

How to proceed
To use `worddocu_table.tcl` as the post-processing command in Innovator classiX:

» Jump to the model browser and select Extras>Customize>Documentation Templates…
   The dialog box of the same name appears.

» Select the documentation template which you want to edit in the Available Documentation Templates list.
» Click on [Properties...].
   The Properties of <Documentation Template> dialog box appears.

» Jump to the Postprocessing tab.
   You can specify a command and its arguments in the Post-processing Command group. The command is automatically carried out upon documentation creation, whereas the created output file will also be implicitly transferred as the first argument alongside the arguments entered in the Arguments field.

» In the Command field: `inotcl -f worddocu_table.tcl`

» Enter the necessary call argument in the Argument field:

   » Click on [Apply] to adopt the settings.
   The post-processing command is configured.

**Call Options**

It is possible to control some formatting using call options:

- `-pi` Also creates a table of figures.
- `-dotFile <template>` Word template file specification.
- `-of <output file>` Specification of an output file for the converted file.
- `-tf <table format>` Table format as an integer value or format name (e.g. Grid1).
- `-hfs <size>` Specification of the font size for table headings.
- `-hfn <font>` Specification of the font name for table headings.
- `-bfs <size>` Specification of the font size for table cells.
- `-bfn <font>` Specification of the font name for table cells.
- `-tfhh <table format>` Format specification for headings in horizontal tables.
- `-tfhv <table format>` Format specification for headings in vertical tables.
- `-tftc <table format>` Format specification for table cells.
   Example of –tf?: `-tfhh {Display 1}`
   Formats with spaces in the name need to be entered in {}, e.g. `{Display 1}`
Documentation Generation Procedure

Documentation generation also occurs by creating a Word for Windows document in the specified output directory.

The converted document is (possibly under the name set in *Ino.DocTitle*) stored in the directory which is also used for documentation generation.