Eclipse RCP Documentation Development with DITAworks

In this post, I'd like to describe an approach that we use to integrate DITA based documentation into Eclipse RCP applications and facilitate collaboration between development and documentation teams. Although we use DITAworks for authoring, but the same conceptual approach can also be used with other authoring tools.

I would not go deep into the details of Eclipse Help subsystem now. It will be a subject for another blog post which will come later. The only thing I'd like to emphasize now is that Eclipse Help subsystem expects documentation to be provided in Eclipse Help format. This documentation can be used in two ways:

- In the "book" mode or so called "Infocenter" mode, where users can browse documentation structure, search through the documentation, browse documentation index. You can explore this mode with Eclipse online documentation.
- As context-sensitive help, when appropriate help page is shown depending on the application context (window).

We use DITA Open Toolkit in order to transform DITA based documentation to Eclipse Help format. This transformation works well for the Infocenter mode out-of-the-box . DITA OT creates special Eclipse plug-in out of the DITA documentation. This plug-in should be distributed with an Eclipse RCP application.

At the same time, additional XML objects need to be created in order to provide context–sensitive help. These objects describe the association between context IDs which are declared in the RCP application source code and the list of links to related topics and optional text description of the context. The main complexity here is synchronization of IDs used in source code with context definitions, taking into account that IDs and documentation are created by different and often distributed teams. This is why we have to maximally automate Eclipse Help context creation and maintenance.

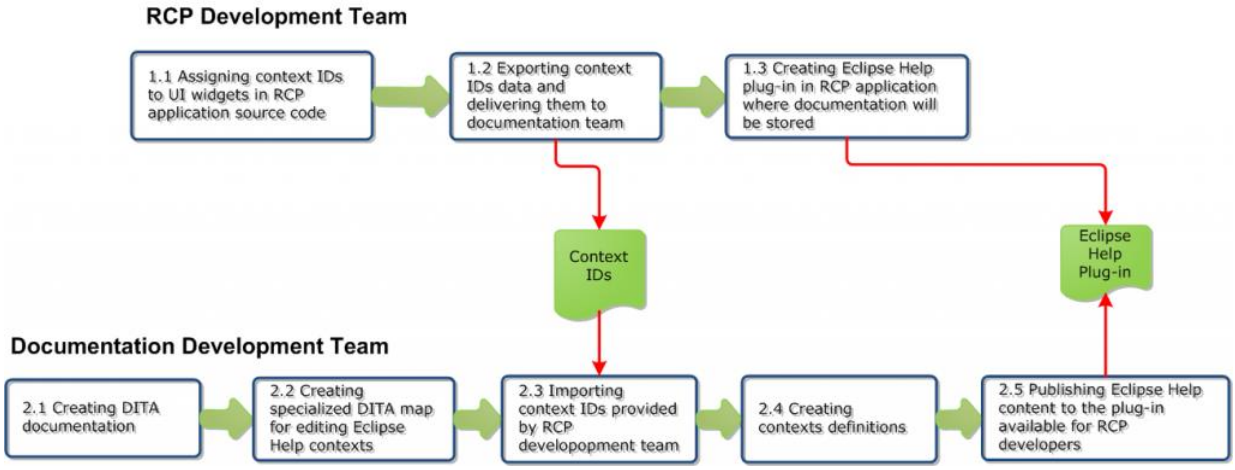
The process we are using for the RCP documentation development can be split in two parts. The first part needs to be done by RCP developers and the second part is documentation team's responsibility.

Part 1 (RCP development team)

1. Assigning context IDs to UI widgets in RCP application source code
2. Exporting context IDs data and delivering them to documentation team
3. Creating Eclipse Help plug-in in RCP application where documentation will be stored

Part 2 (documentation team)

1. Creating DITA documentation
2. Creating specialized DITA map for editing Eclipse Help contexts
3. Importing context IDs provided by RCP development team
4. Creating context definitions
5. Publishing Eclipse Help content to the plug-in provided by RCP developers



To simplify first group of activities, we have created a special Eclipse plugin that should be installed to RCP developers' environment. Second group of activities is automated by DITAworks.

Let's consider these steps in more details.

1.1 Defining Contexts IDs in Eclipse RCP Source Code

As a first step, **RCP developers** should assign unique context ID to UI controls. Each UI component can have one context ID. Later, it will be possible to link several related topics and provide a text description for each Context ID. Context ID assignments can be static or dynamic. Below, I shortly explore each of the methods.

Static contexts can be defined using ***setHelp*** method in *org.eclipse.ui.help.IWorkbenchHelpSystem* which will associate a context ID with a Control, IAction, Menu, or MenuItem. The context ID should be fully qualified with the plug-in ID. For example, the following snippet associates the id "com.example.helpexample.panic_button" with a button in the application.

```
PlatformUI.getWorkbench().getHelpSystem()
                          .setHelp(myButton,
"com.example.helpexample.panic_button");
```

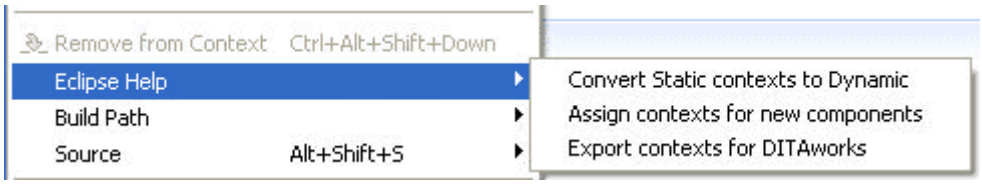
Dynamic contexts are calculated automatically during run time. This can be archieved through the implementation of *org.eclipse.help.IContextProvider* interface which ensures dynamic Context ID calculation. It is also possible to use the following code:

```
PlatformUI.getWorkbench().getHelpSystem()
              .setHelp(control,
                       HelpUtil.getContextId(IHelpContexts.MY_CONTEXT1,
                                             "my.application.main.ui"));
```

This method is recommended for dialogs but works well everywhere.

Dynamic context allow us to store all context IDs in one place which can be then delivered to the documentation team e.g. in an XML file which can also contain additional information (comments, descriptions etc.).

To simplify and speed up help related activities for RCP developers, we've implemented a **special Eclipse plug-in** which should be deployed to RCP developers' Eclipse environment. All functions provided by this plugin can be found in right-click menu item "Eclipse Help" which is available for java projects, java packages or java classes:



Let's take a look at the functionality provided by this plugin:

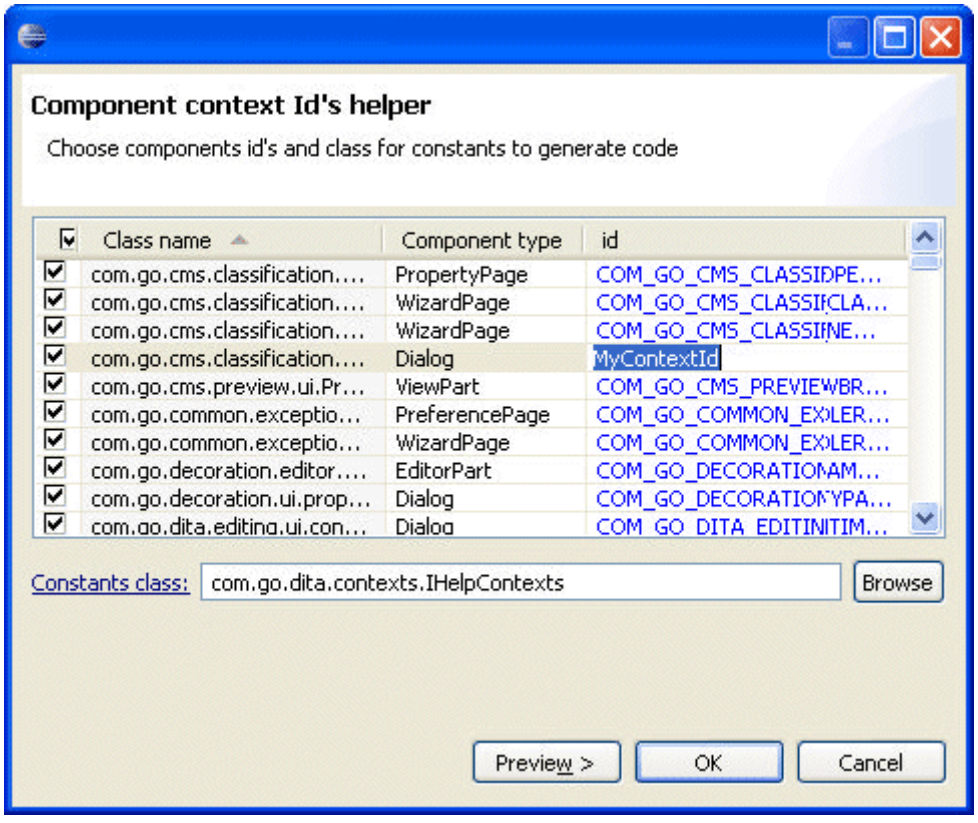
Convert Static contexts to Dynamic

This operation will find all static contexts in the selected scope (project, package or java class) and convert them to dynamic contexts. When a developer executes this function, a special wizard will be started which will search for static contexts and present them in the dialog form. There, the developer will see all static contexts which are found in the application and will be able to modify values automatically assigned by the wizard HelpKey.

The wizard will also allow the developer to specify a class where constants will be defined.

Assign contexts for new components

This operation will search for new UI components that do not have Context ID assigned to them yet. Then, developers will be able to modify autogenerated IDs and generate a Java code which will assign these IDs to the corresponding UI widgets.



1.2 Delivering Context IDs Data to the Documentation Team

In this step, RCP developers should transfer a file with Context IDs definitions to the documentation team. Eclipse plug-in described in the item 1.1 above provides **Export Contexts to DITAworks** function that should be used in order to export context IDs to the documentation team. When this function is executed, a wizard will be started which will search for contexts and will allow to edit descriptions and make comments. These should be delivered to the documentation team. After this operation is done, a directory, where context definitions will be stored, should be selected. Merging with existing context definition file is also supported by the plugin.

Exported file with Context IDs should be provided to the documentation team. We suggest to use version control in order to organize smooth sharing of this file between development and documentation teams.

1.3 Creating Eclipse Help Plug-in in RCP Application

RCP developers should create an eclipse plug-in project in Eclipse IDE and link it from the target platform of the RCP application. This plug-in project will be used to exchange information between the development and the documentation teams. More detailed description of plugin creation procedure can be found in Eclipse online help.

After creating Eclipse Help Plug-in, the plug-in project should be shared with documentation team through the version control (CVS, SVN or any other which is used in your organization). Later, documentation team will use this project as an output for Eclipse Help publishing.

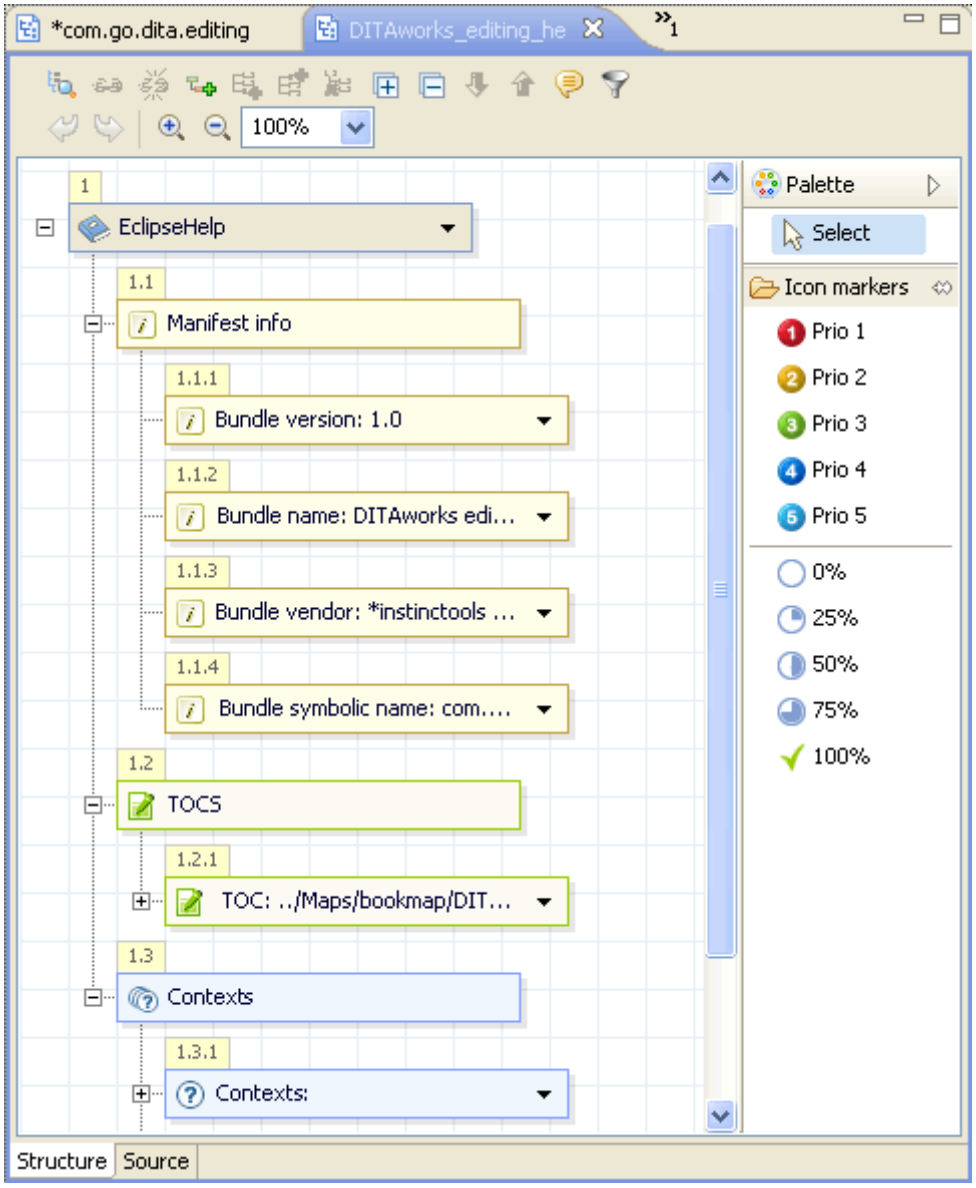
2.1 Creating DITA Documentation

This step is done by the documentation team which should create required deliverables, DITA topics and maps. It can be executed in parallel to the RCP development team activities **1.1 - 1.3** described above.

2.2 Creating Specialized DITA Map for Editing Eclipse Help Contexts

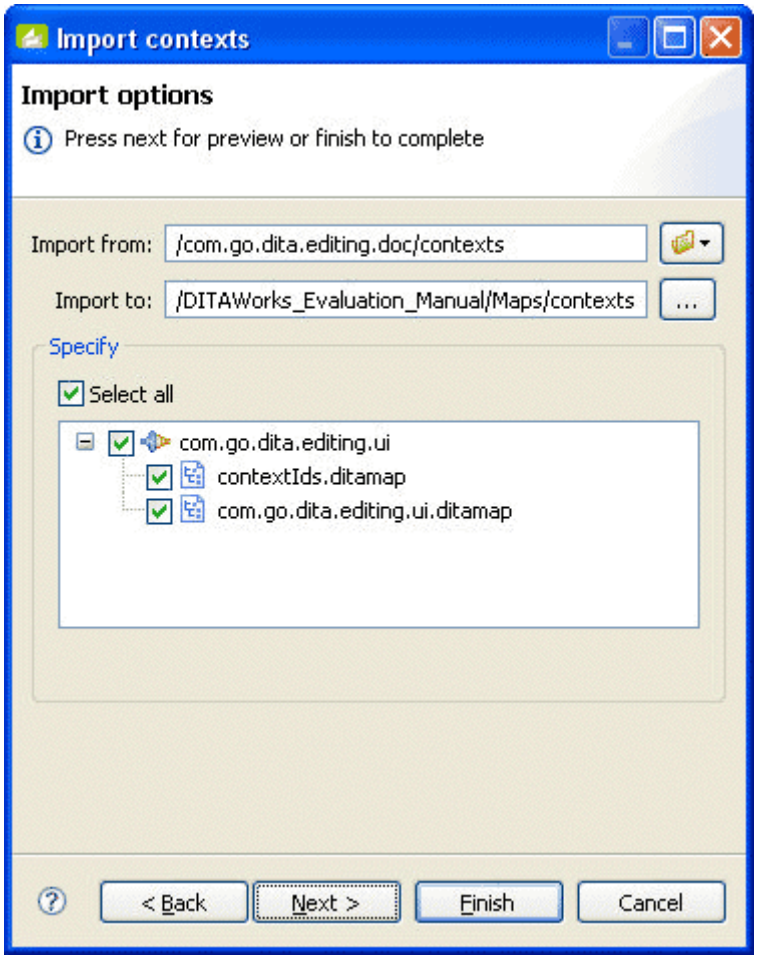
As it was described in our previous post, we use specialized DITA map object to store Eclipse Help context data. This allows us to store context data in DITA and use available tools and validation mechanisms. DITAworks provides *EclipseHelpPluginMap* specialization that should be used for these kinds of maps. To create it in DITAworks, start "**New Map**" wizard, type a title, choose document type "***eclipsehelpplugin***" and choose a folder to store new map.

As a result DITAworks will create a new map with the eclipse help plug-in structure.



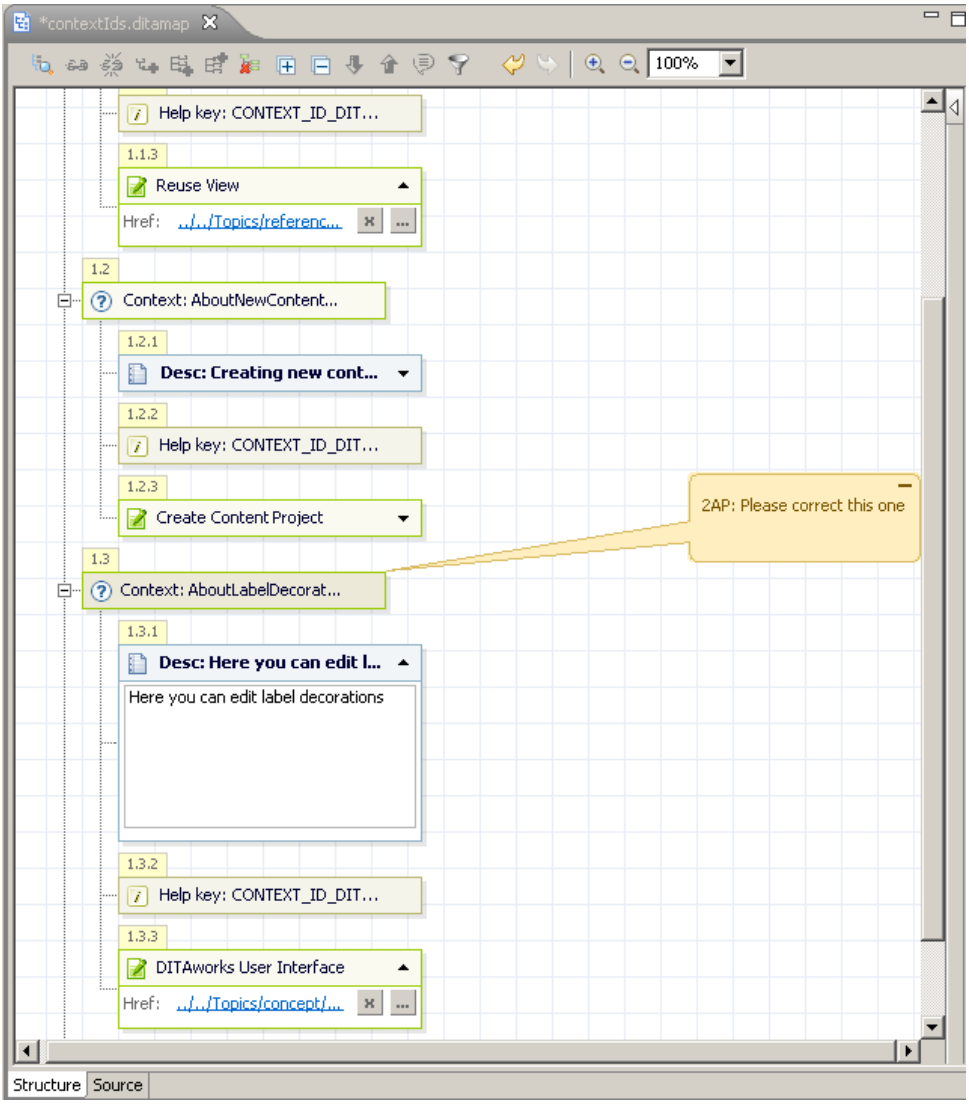
2.3 Importing Context IDs Provided by RCP Development Team

DITAworks already provides functionality that should be used for importing Context IDs data provided by the development team. It can be done by using **DITA | Import Context IDs** wizard available in the right click menu on a content project. This wizard allows to specify a file for import, define the destination folder. It also supports file merging in the case of context IDs already imported earlier.



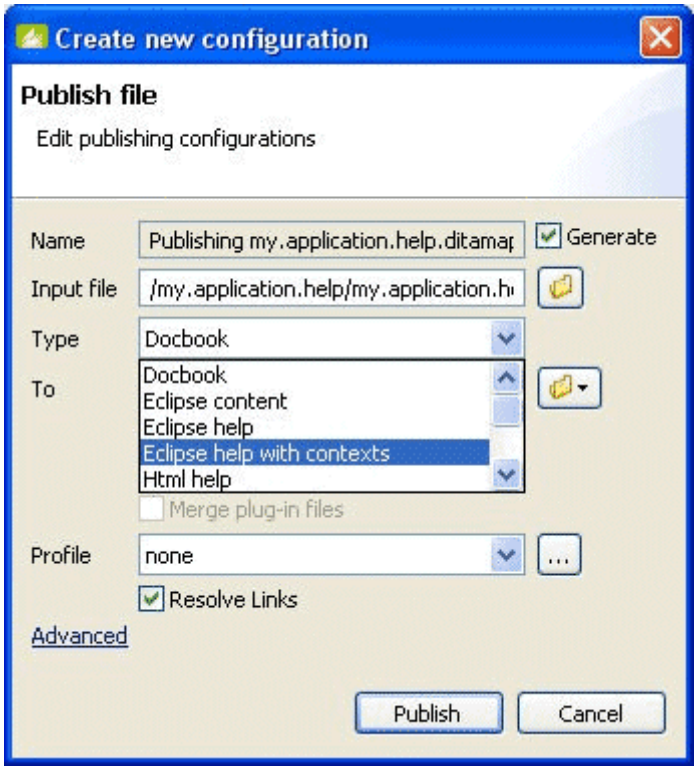
2.4 Defining Contexts

In this step, the documentation team should create context definition using DITAworks map editor.



2.5 Publishing Documentation to the Plug-in Available for RCP Developers

In this step, DITA documentation should be published to the plug-in which was created by RCP developers in the step 1.1. The publishing can be started by selecting a map and choosing **Publish** » **Publish to…** from the right-click menu. It is necessary to select ”***Eclipse help with contexts***” format as a type of deliverable and the plug-in root folder as a publishing destination.



Conclusions

This is how we facilitate collaboration during Eclipse Help documentation development. We would very much appreciate any feedback on the approach described above. We also welcome you to share any experiences which you might have regarding complex RCP documentation development.