

# Questionnaire

## *Introduction*

Thank you for agreeing to participate to the validation study of Mismar. This questionnaire should not take more than 10 minutes to complete and will help us validate the precision and usability of Mismar.

The questions can be answered at any time and in multiple sessions. You may use any tool or document you need, except Mismar itself. If you have any questions while completing the questionnaire, do not hesitate to contact us at [bdagenai@ca.ibm.com](mailto:bdagenai@ca.ibm.com) .

Please note that sections 3 and 4 are optional. They would greatly help us in our research on framework documentation, but might extend the completion time beyond 10 minutes. Feel free to simply skip either or both of these sections.

Please return the completed questionnaire by December 21<sup>st</sup> 2007 to: [bdagenai@ca.ibm.com](mailto:bdagenai@ca.ibm.com).

## **Confidentiality**

When presenting the results, we would like to discuss the results of each editor separately, giving detailed, specific examples. We won't mention your name explicitly, but, since few people have worked on each editor, this would come close to identifying you implicitly. If this is a problem for you, we would report results from your questionnaire in aggregate form only.

I agree to detailed reporting of results on my editor (yes or no): \_\_\_\_\_

## **Section 1 – About you**

In this section, we ask general questions about your experience as an Eclipse developer.

### **Questions:**

1. Approximately how many years of experience do you have in Java development?
2. Approximately how many years of experience do you have in Eclipse development?
3. On which Eclipse component(s) did/do you work? E.g., jdt ui, platform debug, SWT.

## Section 2 – Eclipse Text Editor Implementations

In this section of the study, you will be presented with a short tutorial on how to create a generic Eclipse Text Editor. Then, you will be asked to associate each step in the tutorial with a program element of the editor you are reviewing: **Ant Editor**.

### Eclipse Text Editor Tutorial

To create an Eclipse Text Editor, a developer needs to follow these steps:

1. Create a plug-in project.
2. Make the plug-in project depend on **org.eclipse.ui.workbench.texteditor** plug-in.
3. Make the plug-in project depend on **org.eclipse.jface.text** plug-in.
4. Create an extension **org.eclipse.ui.editors**.
5. Create an element editor in the extension **org.eclipse.ui.editors**.
6. Create a class implementing **ITextEditor**.
7. Create a class extending **SourceViewerConfiguration**.
8. Call method **AbstractTextEditor.setSourceViewerConfiguration**.
9. Create a class extending **RuleBasedScanner**.
10. Create a class extending **RuleBasedPartitionScanner**.
11. Create a class implementing **ITextHover**.
12. Create a class implementing **IContentAssistProcessor**.
13. Create a class implementing **IContentOutlinePage**.

For each of these steps, you will be presented with a list of elements potentially fulfilling the step. Each element is of the right type for the step. For example, if a step asks to implement the interface **IHelloWorld**, only classes that implement the **IHelloWorld** interface (directly or indirectly) are listed.

Please select the element that best fulfills each step for this editor: **Ant Editor**. If more than one element is relevant, simply rank them from the most relevant (1) to the least relevant (10). If an element is not relevant at all, do not put anything beside it. If you are not sure, do not put anything beside it. If you think a relevant element is not in the list, add it beside the “Other” option. If no element is relevant for this step in the context of your editor, choose the “No relevant element” option by putting a 1 beside it.

The **Ant Editor** is represented by this editor element in the ant.ui plugin.xml file:

```
<extension
  point="org.eclipse.ui.editors">
  <editor
    name="%AntEditor.name"
    icon="$nl$/icons/full/obj16/ant_buildfile.gif"
    class="org.eclipse.ant.internal.ui.editor.AntEditor"
    contributorClass="org.eclipse.ant.internal.ui.editor.AntEditorActionContributor"
    id="org.eclipse.ant.ui.internal.editor.AntEditor">
    <contentTypeBinding
      contentTypeId="org.eclipse.ant.core.antBuildFile"/>
  </editor>
  ...
```

### Steps:

1. Create a class implementing **ITextEditor**.

The appropriate implementation(s) of ITextEditor for **Ant Editor** is(are):

- JSEditor
- StructuredTextEditor
- DebugTextEditor
- JavaSnippetEditor
- PropertiesFileEditor
- AbstractTextEditor
- StatusTextEditor
- AbstractDecoratedTextEditor
- TextEditor
- CompilationUnitEditor
- JavaEditor
- ClassFileEditor
- AntEditor
- Other:
- No relevant element.

2. Create a class extending **SourceViewerConfiguration**.

The appropriate implementation(s) of SourceViewerConfiguration for **Ant Editor** is(are):

- JSEditorSourceViewerConfiguration
- JSSourceViewerConfiguration
- NewHTMLTempaltesWizardPage\$6
- TextSourceViewerConfiguration
- StructuredTextViewerConfigurationHTML
- AntSourceViewerConfiguration
- StructuredTextViewerConfigurationWSDL
- AntEditorSourceViewerCondiguration
- JavaSourceViewerConfiguration
- AntTemplateViewerConfiguration
- Other:
- No relevant element.

3. Call method **AbstractTextEditor.setSourceViewerConfiguration**.

The method `AbstractTextEditor.setSourceViewerConfiguration` for **Ant Editor** is called in:

- `PropertiesFileEditor()` - `PropertiesFileEditor`
- `initializeEditor()` - `JavaEditor`
- `setPreferenceStore(...)` - `JavaEditor`
- `JavaSnippedEditor()` - `JavaSnippedEditor`
- `TextEditor()` – `TextEditor`
- `createPartControl(Composite)` – `StructuredTextEditor`
- `setSourceViewerConfiguration()` – `JSEditor`
- `PDEProjectionSourcePage(PDEFormEditor, String, String)` – `PDEProjectionSourcePage`
- `AntEditor()` – `AntEditor`
- `setSourceViewerConfiguration(SourceViewerConfiguration)` – `StructuredTextEditor`
- `updateSourceViewerConfiguration()` - `StructuredTextEditor`
- Other:
- No relevant element.

4. Create a class extending **RuleBasedScanner**.

The appropriate implementation(s) of `RuleBasedScanner` for **Ant Editor** is(are):

- `SingleTokenJavaScanner`
- `JavaCommentScanner`
- `JavaCodeScanner`
- `JavaPartitionScanner`
- `PropertyValueScanner`
- `PropertiesFilePartitionScanner`
- `BufferedRuleBasedScanner`
- `JavaDocScanner`
- `JavaCodeScanner`
- `JSPELCodeScanner`
- `RuleBasedPartitionScanner`
- `AbstractJavaScanner`
- `XMLPartitionScanner`
- `AbstractAntEditorScanner`
- `AntEditorTagScanner`
- `AntEditorProcInstrScanner`
- `AntEditorPartitionScanner`
- Other:
- No relevant element.

5. Create a class extending **RuleBasedPartitionScanner**.

The appropriate implementation(s) of RuleBasedPartitionScanner for **Ant Editor** is(are):

- JavaPartitionScanner
- PropertiesFilePartitionScanner
- AntEditorPartitionScanner
- ManifestPartitionScanner
- XMLPartitionScanner
- Other:
- No relevant element.

6. Create a class implementing **ITextHover**.

The appropriate implementation(s) of ITextHover for **Ant Editor** is(are):

- JavaScriptTagInfoHoverProcessor
- BestMatchHover
- AnnotationHoverProcessor
- ProblemAnnotationHoverProcessor
- XMLTagInfoHoverProcessor
- AbstractHoverProcessor
- AntTemplateVariableTextHover
- ProblemHover
- BestMatchHover
- JavaEditorTextHoverProxy
- JavadocHover
- NLSStringHover
- JavaTypeHover
- JavaSourceHover
- AbstractJavaEditorTextHover
- JavaDebugHover
- AnnotationHover
- XMLTextHover
- ManifestTextHover
- Other:
- No relevant element.

7. Create a class implementing **IContentAssistProcessor**.

The appropriate implementation(s) of IContentAssistProcessor for **Ant Editor** is(are):

- JavaScriptContentAssistProcessor
- AbstractContentAssistProcessor
- JavaCompletionProcessor
- XMLContentAssistProcessor
- JavaCompletionProcessor
- TemplateCompletionProcessor
- AntEditorCompletionProcessor
- TypeContentProposalProvider
- VariableNamesProcessor

- JavaSourcePackageFragmentRootCompletionProcessor
- JavaDebugContentAssistProcessor
- FieldNameProcessor
- JavaPackageCompletionProcessor
- JavaTypeCompletionProcessor
- JavadocCompletionProcessor
- Other:
- No relevant element.

8. Create a class implementing **IContentOutlinePage**.

The appropriate implementation(s) of IContentOutlinePage for **Ant Editor** is(are):

- JSContentOutlinePage
- ConfigurableContentOutlinePage
- ASDContentOutlinePage
- ADTContentOutlinePage
- OthersContentOutlinePage
- ComponentsContentOutlinePage
- ContentOutlinePage
- ManagedBeanContentOutlinePage
- JavaOutlinePage
- AntEditorContentOutlinePage
- SourceOutlinePage
- Other:
- No relevant element.

This completes the mandatory part of the questionnaire. Thank you very much for taking the time to respond to it. If you have extra time, we would appreciate your completing the next two optional sections about the quality of the tutorial above and your preferences and habits when using and authoring framework documentation.

## Section 3 – Eclipse Text Editor Documentation (OPTIONAL)

In this section, please evaluate the quality of the short tutorial introduced in Section 2.

As a reminder, here is the complete tutorial again:

1. Create a plug-in project.
2. Make the plug-in project depend on **org.eclipse.ui.workbench.texteditor** plug-in.
3. Make the plug-in project depend on **org.eclipse.jface.text** plug-in.
4. Create an extension **org.eclipse.ui.editors**.
5. Create an element editor in the extension **org.eclipse.ui.editors**.
6. Create a class implementing **ITextEditor**.
7. Create a class extending **SourceViewerConfiguration**.
8. Call method **AbstractTextEditor.setSourceViewerConfiguration**.
9. Create a class extending **RuleBasedScanner**.
10. Create a class extending **RuleBasedPartitionScanner**.
11. Create a class implementing **ITextHover**.
12. Create a class implementing **IContentAssistProcessor**.
13. Create a class implementing **IContentOutlinePage**.

### Questions

1. Based on your experience, are there steps in the tutorial that should be marked as **optional** for a complex text editor (e.g., like an XML editor)? List the numbers and reasons here:
2. Based on your experience, are there steps in the tutorial that should not be part of it? Why? List the numbers and reasons here:
3. Based on your experience, are there important steps missing? List them here:
4. Is the step ordering adequate? If not, can you propose a new ordering?
5. Based on your experience, if you had to explain to someone how to create a text editor in **three (3)** steps, which steps would you choose? List the numbers and reasons here:
6. If you needed to write a new Eclipse text editor for the Perl programming language, what kind of documentation would you use? Rank the following choices from 1 (most likely to use) to 8 (least likely to use):
  - The code of an editor you wrote. \_\_\_\_
  - The code of an editor in Eclipse. \_\_\_\_
  - The code of an open source Perl editor outside of Eclipse. \_\_\_\_
  - The documentation in Eclipse help system (e.g., The “Text editors and platform text” page in the Platform Plug-in Developer guide). \_\_\_\_

- A web tutorial. \_\_\_\_ Which one:
- A book. \_\_\_\_ Which one:
- A cheatsheet. \_\_\_\_ Which one:
- Other: \_\_\_\_

## Section 4 – General Framework documentation (OPTIONAL)

In this section of the study, we ask questions about your preferences and habits when using and authoring framework documentation.

1. Did you ever write documentation (except javadoc) like tutorials or cheatsheets on Eclipse? What kind of documentation?
2. Did you ever write documentation (except javadoc) like tutorials or cheatsheets on Eclipse text editors? What kind of documentation?
3. Suppose that you must use a new open source framework, SuperCoolWidget, and you are provided with a tutorial similar to the Eclipse text editor one. If there were other sources of documentation attached to each step, what kind of documentation would be most helpful? Rank the following choices from 1 (most helpful) to 8 (least helpful). For the sake of this question, imagine that the step is “Create a class implementing IHotWidget”.
  - A UML class diagram showing how your implementation class fits with the rest of the framework and the other steps. \_\_\_\_
  - A UML sequence diagram showing how your implementation class will be used by the framework. \_\_\_\_
  - Example code implementing this step. \_\_\_\_
  - JUnit class testing the contract of IHotWidget.
  - The classes (and their source code) in the framework that use an instance of IHotWidget. \_\_\_\_
  - A small description explaining **why** this step is important and how it fits with the rest of the tutorial. \_\_\_\_
  - A small description explaining **how** to fulfill this step, possibly by describing the main methods to implement. \_\_\_\_
  - Other: \_\_\_\_
4. Suppose that you create another Eclipse text editor and, as a reminder, you are provided with the Eclipse text editor tutorial introduced in Section 2. If there were other sources of documentation attached to each step, what kind of documentation would be most helpful **considering that you already have experience in developing Eclipse text editors**? Rank the following choices from 1 (most helpful) to 9 (least helpful).
  - A UML class diagram showing how your implementation class fits with the rest of the framework and the other steps. \_\_\_\_
  - A UML sequence diagram showing how your implementation class will be used by the framework. \_\_\_\_
  - Example code implementing this step. \_\_\_\_
  - JUnit class testing the contract of IHotWidget. \_\_\_\_
  - The classes (and their source code) in the framework that use an instance of IHotWidget. \_\_\_\_
  - A small description explaining **why** this step is important and how it fits with the rest of the tutorial. \_\_\_\_

- A small description explaining **how** to fulfill this step, possibly by describing the main methods to implement. \_\_\_\_
  - Other: \_\_\_\_
5. When you need to use a new framework, what kind of documentation do you usually look for? Rank the following choices from 1 (least likely to use) to 7 (most likely to use).
- A book. \_\_\_\_
  - A web tutorial guiding the reader in accomplishing one specific task with the framework. \_\_\_\_
  - API documentation (e.g., javadoc). \_\_\_\_
  - Reference documentation covering most features of the framework. \_\_\_\_
  - Code examples. \_\_\_\_
  - Help from a knowledgeable colleague: \_\_\_\_
  - Other: \_\_\_\_
6. Did you ever produce documentation, excluding javadoc, intended for developers? If your answer is no, skip to question 7:
7. What kind(s) of developer documentation did you produce? Please, indicate the support (e.g., Microsoft Word, cheatsheet, html):
8. When producing developer documentation, what are the main difficulties that you usually encounter? Rank the following choices from 1 (least problematic) to 8 (most problematic). Do not rank anything that is not a problem.
- It takes a long time to write. \_\_\_\_
  - It is hard to determine the ordering of the steps (when writing tutorials). \_\_\_\_
  - It takes a long time to maintain. \_\_\_\_
  - You need to constantly switch from the development environment to the documentation creation tool (e.g., Microsoft Word). \_\_\_\_
  - It is error-prone (spelling error, code example error, incorrect class name, etc.): \_\_\_\_
  - It is hard to find the right level of detail that should go in the document. \_\_\_\_
  - It is hard to determine where to start. \_\_\_\_
  - Other: \_\_\_\_
9. When using (reading) developer documentation, what are the main difficulties that you usually encounter? Rank the following choices from 1 (most problematic) to 7 (least problematic). Do not rank anything that is not a problem.
- You need to constantly switch from the developer documentation to your development environment. \_\_\_\_
  - The detail is overwhelming. \_\_\_\_
  - There is not enough explanation or detail. \_\_\_\_
  - It is hard to understand how it fits with the big picture. \_\_\_\_
  - It is hard to quickly find the important elements from the text. \_\_\_\_
  - There are errors in the documentation. \_\_\_\_

Other: \_\_\_\_

10. Do you have any other comments or suggestions regarding developer documentation?

This completes the optional part of the questionnaire. Thank you very much for taking the time to respond to it.