Change request log

# Team

Specify the team members working on this change request.

Angela Chen & Uma Kothuri

# Change Request

Provide the id and description of the change request.

jedit-cr-2

In the File » Recent Files main menu of jEdit, the text box on top of the recent files list allows to highlight recent files names that match with a given string (see Figure 2). The string in the text box should match all the files that contain it anywhere in their name. However, the highlight works only when the string matches the beginning of a file name. You are requested to modify this feature so that the highlight occurs for the cases when the string is contained anywhere in the file name.

# Concept Location

Use the table below to describe each step you follow when performing concept location for this change request. In your description, include the following information when appropriate:

* IDE Features used (e.g., searching tool, dependency navigator, debugging, etc.)
* Queries used when searching
* System executions and input to the system
* Interactions with the system (e.g., pages visited)
* Classes visited
* The first class found to be changed (this is when concept location ends)

When there is a major decision/step in the process, include its rationale, i.e., why that decision/step was taken.

Make sure you time yourselves when going through this process and provide the total time spent below.

The following is an example of a concept location process for the change request "Color student schedule":

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We opened the project on IntelliJ IDEA and built it. | This is where we will conduct the entire change request process. |
| 2 | We interacted with jEdit and looked at some of its features. | To familiarize ourselves with jEdit and some of its features, and identify the elements we had to change. |
| 3 | We looked through the documentation and used static text search with “Files” and found the “RecentDirectoriesProvider” class as a first result. | “Files” is a concept we identified from the change request. |
| 4 | Another search result from step 3 lead us to find a possibly related class recentFilesProvider from RecentFilesProvider.java | The first result from step 3 was unrelated to the concepts from the change request, but the second result may be related. |
| 5 | We looked through the project file system and found a file name “RecentFilesProvider.java”, then used static text search on IntelliJ (ctrl+shift+f) to find the “recentFilesProvider” class. | This class name and file seems to be related to the concept “recent files” |
| 6 | We found a statement within the update() method that determines the regex pattern used for the file name searches within the class. | This area of the code is related to matching all file names to the search string. This is the class that will be changed. |

**Time spent (in minutes):** 72

# Impact Analysis

Use the table below to describe each step you follow when performing impact analysis for this change request. Include as many details as possible, including why classes are visited or why they are discarded from the estimated impact set.

Do not take the impact analysis of your changes lightly. Remember that any small change in the code could lead to large changes in the behavior of the system. Follow the impact analysis process covered in the class. Describe in details how you followed this process in the change request log. Provide details on how and why you finished the impact analysis process.

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We inspected the code in the recentFilesProvider class and found that it called no methods from other classes. | We wanted to find any classes that the initial impact set’s class depends on. |
| 2 | We further inspected recentFilesProvider and found that it implements DynamicMenuProvider. | We determined DynamicMenuProvider was not related to the concepts and therefore will not be affected by the change. We did not add it to the estimated impact set |
| 3 | We used static text search in IntelliJ on update and updateEveryTime, which are the methods in the recentFilesProvider class. | We wanted to find any classes that depend on the initial impact set |
| 4 | We used static text search in IntelliJ to find if there are any recentFilesProvider objects declared in any other files and did not find any results. | We wanted to find any classes that depend on the initial impact set |
| 5 | We couldn’t find any classes that depend on the recentFilesProvider class nor vice versa. | We have determined that there are no additional classes within the estimated impact set. |

**Time spent (in minutes):** 7

# Actualization

Use the table below to describe each step you followed when changing the code. Include as many details as possible, including why classes/methods were modified, added, removed, renamed, etc.

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We went to the update method in the recentFilesProvider class (org/gjt/sp/jedit/menu/RecentFilesProvider.java) with the statement that determines the regex pattern used for the file name searches. | That is where we determined the required change to be. |
| 2 | We changed the statement to have “\*” in the regex pattern. | This would make the string in the text box match all the files that contain it anywhere in their name, which meets the change request’s requirements. |
| 3 |  |  |
| 4 |  |  |

**Time spent (in minutes):** 3

# Validation

Use the table below to describe any validation activity (e.g., testing, code inspections, etc.) you performed for this change request. Include the description of each test case, the result (pass/fail) and its rationale.

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | Test case defined:  Inputs: “sam”  Expected output: highlights sample.pdf | This is the regular expected behavior (from before the change request).  The test passed. |
| 2 | Test case defined:  Inputs: “.pdf”  Expected output: highlights sample.pdf | This is the expected behavior after the change request for inputs that match the end of file names.  The test passed. |
| 3 | Test case defined:  Inputs: “mpl”  Expected output: highlights sample.pdf | This is the expected behavior after the change request for inputs that match the middle of file names.  The test passed. |
| 4 | Test case defined:  Inputs: “abc”  Expected output: does not highlight sample.pdf | This is the expected behavior after the change request for inputs that don’t match any file names.  The test passed. |

**Time spent (in minutes):** 5

# Timing

Summarize the time spent on each phase.

|  |  |
| --- | --- |
| Phase Name | Time (in minutes) |
| Concept location | 72 |
| Impact Analysis | 7 |
| Actualization | 3 |
| Verification | 5 |
| Total | 87 |

# Reverse engineering

Create a UML sequence diagram (or more if needed) corresponding to the main object interactions affected by your change.

Create a partial UML class diagram of the classes visited while navigating through the code. Include the associations between classes (e.g., inheritance, aggregations, compositions, etc.), as well as the important fields and methods of each class that you learn about. The diagram may have disconnected components. Use the UML tool of your preference. When a significant fact about a class or method is learned, indicate it via annotations on the diagram. **For each change request, start with the diagram produced in the previous change request. For the first, you will start from scratch.**

# Conclusions

Perform and analysis of the change requests and the change process. List the major challenges this change request posed.

List all the classes and methods you have changed.

For example:

*For this change, concept location was relatively easy because the system is small and its architecture and code are not complicated. Concept location, impact analysis, actualization (and change propagation) was done using JRipples, which was very useful. Testing was performed using JUnit and Abbot. It took me long time to make Abbot work, mainly because I had to learn how to use the tool.*

*Classes and methods changed:*

* *org/gjt/sp/jedit/search/HyperSearchResults.java/HyperSearchResults*
  + *void foo(p:integer)*
* *org/gjt/sp/jedit/search/HyperSearchResults.java/HyperSearchFind*
* *etc.*

Concept location was relatively easy and not as challenging as we expected because the concepts we used as keywords were able to provide useful search results that helped us quickly locate the initial impact set’s class. The entire change process was done in IntelliJ IDEA, and the static search tool was very helpful. It took us the longest to set up and build the project in IntelliJ because we were not familiar with that process, and the actual change process itself took relatively much less time to complete.

Classes and methods changed:

* org/gjt/sp/jedit/menu/RecentFilesProvider.java/RecentFilesProvider
  + public void update(JMenu menu)