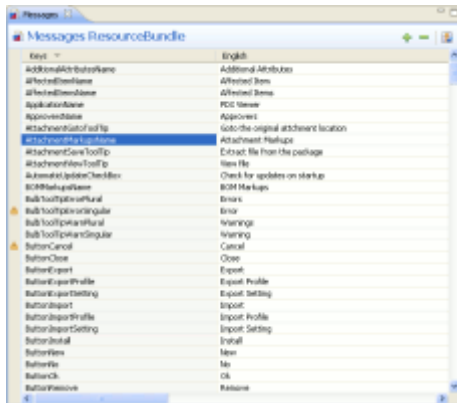


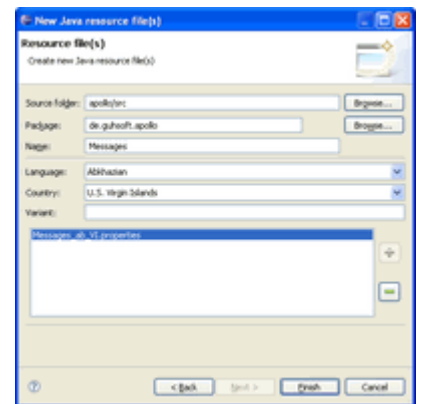
Getting Started

This Help provides a step by step walk-through of the Java Internationalization Tools.

JInto is a plugin for Eclipse which assists you in developing internationalized Java applications. You can easily create, edit and keep up-to-date your resource files.



JInto provides a resourceBundle editor for editing several .properties files at once. It collects all .properties files which define one resourceBundle and displays them in the editor. There is one column for errors and warnings, one column for the keys and one column for every .properties file.



JInto also provides a wizard for creating resourceBundle files. The wizard simplifies the language- and country-code selection and you can create more than one .properties file at once.

```

is(IStatus.ERROR, pluginID, IStatus.ERROR,
.log().log(status);
return getString(key);
};

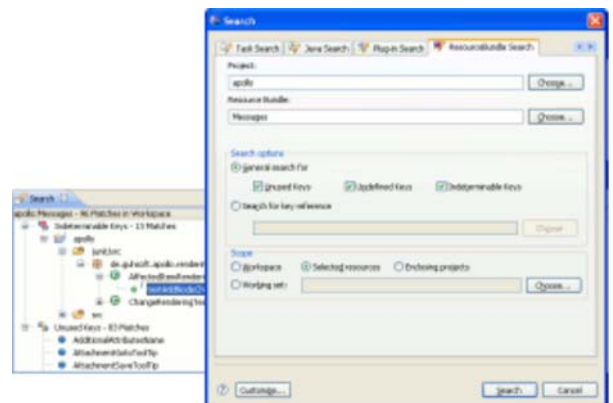
:getString
s.fEditor

itaModel()

```

JInto integrates into the Java editor to provide some cool features like content assist, quick fix/quick assist and fast navigation

Last but not least, JInto provides a powerful search functionality, which will search for unused, undefined and indeterminable keys in the .java files of a project for a resourceBundle.



We appreciate your input. For bugs-reports, feedback, feature requests or comments, please send an e-mail to jinto@guh-software.de

Using the editor




Opening an editor

To open a resourceBundle editor, simply double-click on a .properties file in the Package Explorer. If for some reason, the .properties file doesn't open in JInto, but in another editor, close that editor again, and then right-click the .properties file you want to open, and then select "Open with" and "Java ResourceBundle Editor". The editor automatically collects the data from all .properties files which are connected to the selected file (same folder; same prefix; same extension; only different language, county or variant).



The editor contains a table with the content of the files. There is an error/warning column for displaying the errors and warnings found in this resourceBundle. The next column will be the keys column. This column displays all keys found in the .properties files. Finally there will be a column for every .properties file. It contains the values for the key, for the given language. The editor also shows the name of the resourceBundle and two buttons, which can be used to add remove keys (and therefore rows in the table).

When a resourceBundle editor is open and selected, there will be a special jinto toolbar.

-  This button will assign the same width to every column, so all columns fit into the editor window. After this action, there will be no horizontal scrollbar in the table. Values that don't fit into the columns are clipped.
-  Calculates the optimal width for every column. The optimal width of the column is the width required so that the content of all cells in the column can be displayed without being clipped.
-  With this button you can add an additional language to the resourceBundle. It opens a wizard with all relevant information already filled in - you must only select language, and optionally country and variant.



To make working with the resourceBundle editor easier, the status line of eclipse shows additional information like row count and the key of the currently selected row.

If you press F2, an editor dialog which contains the text of the selected cell will be opened for making editing easier. With this editor you more easily edit long text than with the normal cell editor.

Adding and removing rows

For adding and removing rows from the resourceBundle, the editor has two buttons in the upper-right corner. With the + Button you add a row below the currently selected row, the - button removes the currently selected row from the resourceBundle.

You can also use the + and the - keys on you keyboard for doing these actions.

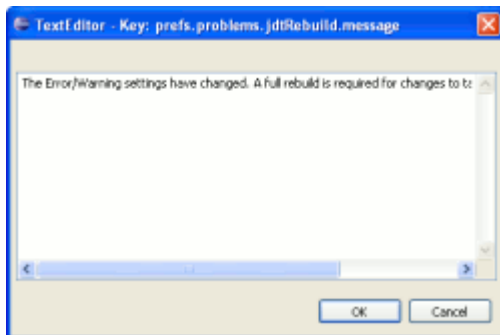
-  Add a new row to the resourceBundle.
-  Remove the selected row from the resourceBundle.

Searching in the editor



If you want to search for a string in the resourceBundle, you can do this by using the Find action (**Edit->Find/Replace... or Ctrl+F**). In this dialog you can enter text which will be searched and selected in the editor. You can search forward or backward and you have the option of doing a case sensitive search. If you want the search to continue at the beginning of the file when the search hits the end of the file, you can select wrap search.

Using the Edit Dialog



If you want a extra (multi-line) text field to enter a long message, press **Ctrl+F2** and a edit dialog opens. The dialog shows the content of the selected cell in a multi-line text field. To apply the changes click on the OK button or press **Ctrl+Enter**.

Error/warning annotations

The automatic background search of the editor runs every time the editor has been saved. It shows the errors found and warnings for the resourceBundle in the first column of the table.

There can be only one error but different kinds of warnings. An error will be shown if you enter a duplicate key into the resourceBundle.

Please see the following table of warnings:

Empty keys and values If a key or a value is empty, a warning will be shown in the first column of the table.

Possible redundancy If all values of a key are equal to the values of another key, this is a possible redundancy and a warning will be shown. You can delete one key and use only the other key in the application.

Using JInto features in the Java editor

Content Assist

The JInto content assist will make your life much easier. Everytime you need a value from the ResourceBundle, JInto will show you a list of possible completions.

Let's give an example:

You have configured `getString(String key)` of class `de.guhsoft.jinto.ui.Messages` as an accessor method for ResourceBundle `de.guhsoft.jinto.core.messages`. You've used the normal Java content assist to create the following code: `Messages.getString(key)`. Now you type **Ctrl + Space** and JInto will give you a list of possible key's from the `de.guhsoft.jinto.core.messages` ResourceBundle.

Please see the pictures below:

```
...
= new Status(IStatus.ERROR, pluginID, IStatus.ERROR, "ExecutionEx
fault().getLog().log(String key
= Messages.getString(key);
= Messages.getString(getErrorMessageKey());
penError(this.fEditor.getEditorSite().getShell(), title, message,
```

Use content assist for
`Messages.getString(key)`

```
context() {
    setResourceDataModel().getUndoContent();
}
```

```
...
= new Status(IStatus.ERROR, pluginID, IStatus.ERROR, "ExecutionEx
fault().getLog().log(status);
= Messages.getString(key);
= Messages.getString
penError(this.fEditor
```

Press **Ctrl + Space**

```
context() {
    setResourceDataModel()
```

```
...
= new Status(IStatus.ERROR, pluginID, IStatus.ERROR, "ExecutionEx
fault().getLog().log(status);
= Messages.getString(error.);
= Messages.getString
penError(this.fEditor
```

Type the beginning of the
key you want to use.

```
context() {
    setResourceDataModel()
```

The screenshot shows an IDE window with a code editor and a content assist popup. The code editor contains the following code snippet:

```
context() {
    setResourceDataModel().getUndoContent();
}

= new Status(IStatus.ERROR, pluginID, IStatus.ERROR, "ExecutionEx
fault().getLog().log(status);
= Messages.getString(key);
= Messages.getString
penError(this.fEditor
```

The content assist popup is open, showing a list of suggestions for the `getString(key)` call. The suggestions are:

- codeAssist.quickAssist.addNewKey
- codeAssist.quickAssist.addNewKey.info
- codeAssist.quickAssist.addNewKeyDialog.column.loc
- codeAssist.quickAssist.addNewKeyDialog.column.me
- codeAssist.quickAssist.addNewKeyDialog.keyEmpty
- codeAssist.quickAssist.addNewKeyDialog.keyExists
- codeAssist.quickAssist.addNewKeyDialog.labelKey
- codeAssist.quickAssist.addNewKeyDialog.message

The popup also shows the package name `de.guhsoft` and a search bar with the text "Please enter".

```

    = new Status(IStatus.ERROR, pluginID, IStatus.ERROR, "ExecutionEx
fault().getLog().log(status);
Messages.getString("error.dialog.title");
    = Messages.getString(getErrorMessageKey());
    openError(this.fEditor.getEditorSite().getShell(), title, message,

```

Press **Enter** and the selected key will be completed.

```

context() {
    getResourceDataModel().getUndoContent();

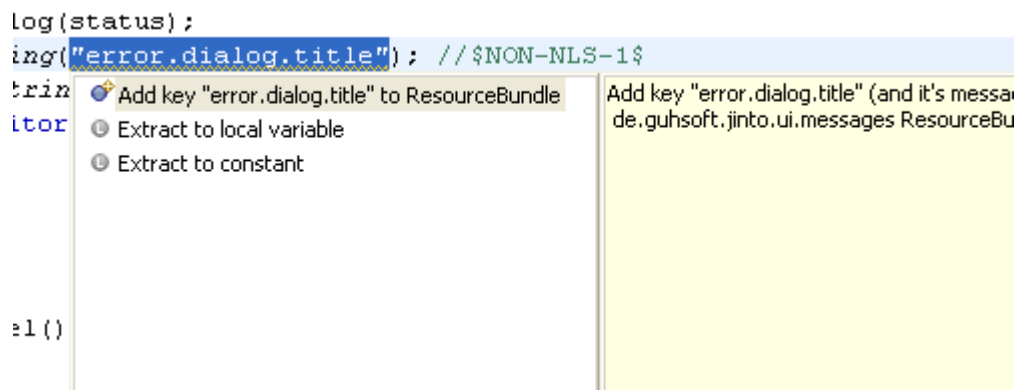
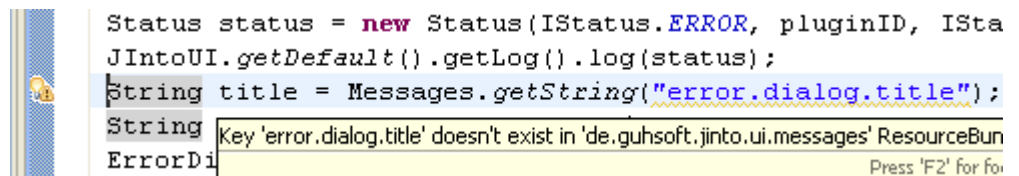
```

Quick Fix/Quick Assist

The Quick Fix/Quick Assist feature helps you to add new or non existing keys and it's messages fast and clean into the ResourceBundle without switching into the ResourceBundle editor.

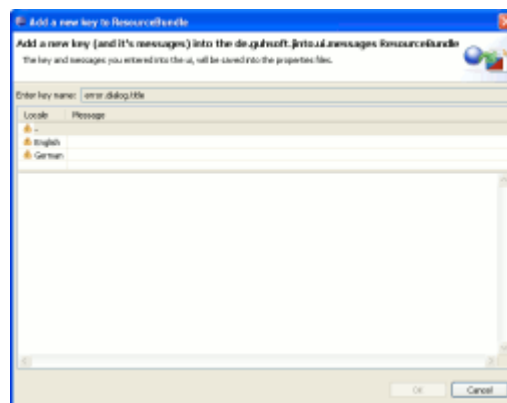
If there is a key (as string with quotes) which is not existing in the linked ResourceBundle, you can use the quick fix feature to add messages for that key into the ResourceBundle. Left click on the light bulb or invoking **Ctrl+1 (Edit > Quick Fix)** brings up the *Add key 'X' to ResourceBundle* correction proposal. If you select this correction proposal, an editor opens where you can enter the messages for this key and save them into the ResourceBundle by pressing **OK**.

Click on the light bulb or press **Ctrl+1 (Edit > Quick Fix)** to show the correction proposals.

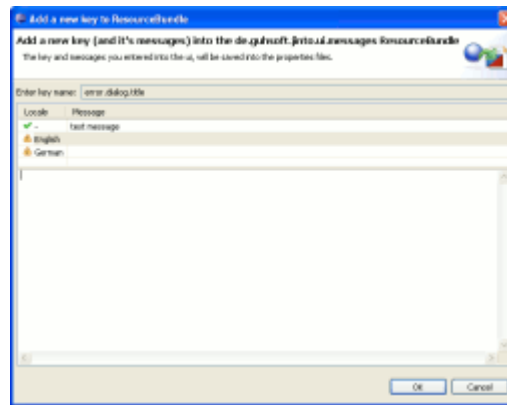


Select the *Add key 'X' to ResourceBundle* proposal

Then an editor opens where you can enter the messages for the key. After entering the message for the selected language, you can switch to the next language by clicking into the table or invoking **Ctrl+Tab**. If the last language is selected, **Ctrl+Tab** moves the focus to the OK button.

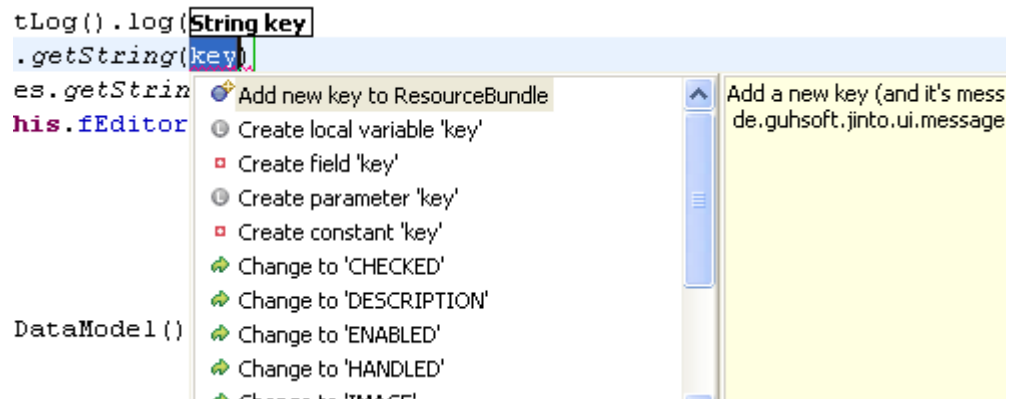


If there is a message for a displayed language, a check-icon is shown before the language.
 After pressing *OK*, the messages are saved under the given key.

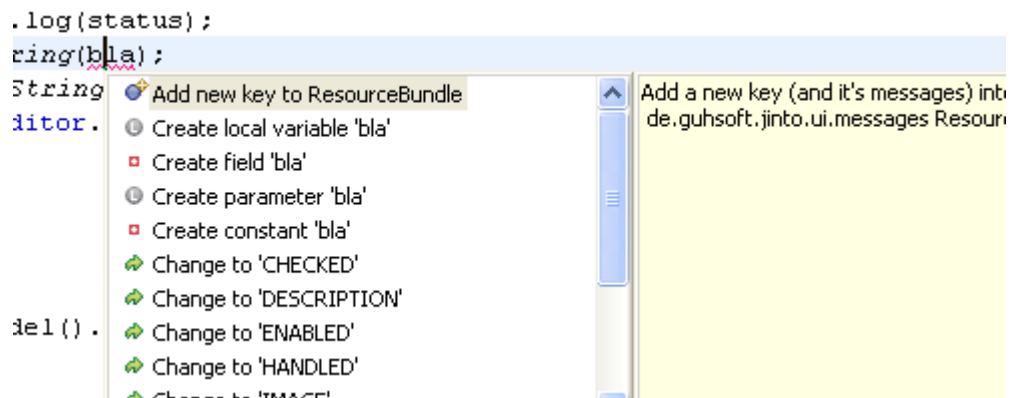


If you want to add a new key into the ResourceBundle without switching into the ResourceBundle editor, you can use the quick assist. The JInto quick assist is invoked on a selection of the right argument in a accessor method invocation and uses the same shortcut as quick fixes (**Ctrl+1**).
 For example if you type a method invocation by using code assist:

Press **Ctrl+1** to show the proposals. Select *Add new key to ResourceBundle* to invoke the "Add new Key"-Dialog.



Press **Ctrl+1** to show the proposals. Select *Add new key to ResourceBundle* to invoke the "Add new Key"-Dialog.



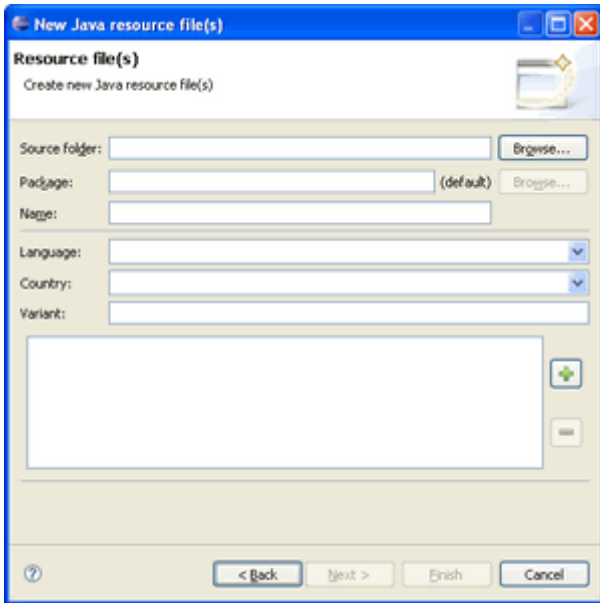
Fast Navigation

```
try {
    ...
} catch (Exception e) {
    ...
    oUI.getID();
    status(IStatus.ERROR, pluginID, IStatus.ERROR,
    ...
    getLog().log(status);
    s.getString("error.dialog.title"); //$NON-NLS-1$
    ges.getString(getErrorMessageKey());
    this.fEditor.getEditorSite().getShell(), tit
}
```

If you want to get from the Java editor into the ResourceBundle editor with one click, you can use the JInto fast navigation.

You only have to hover a key and press **Ctrl**. If the key is found in the linked ResourceBundle, a hyperlink is shown. When you click on that hyperlink, it opens the ResourceBundle editor and selects the clicked key.

ResourceFile Wizard



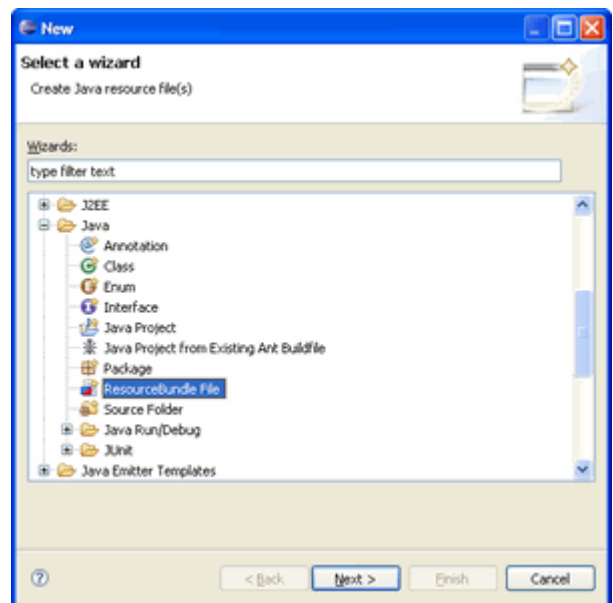
In this section, you will learn how to create new resourceBundle files with the JInto resourceFile wizard. The wizard simplifies the language- and country-code selection and you can create several .properties files in one step.

The following table contains a tutorial on how to generate 4 .properties files at once. The generated files are:

Language	Country	Variant
German	Germany	Unix
German	Germany	Windows
English	United Kingdom	Unix
English	United Kingdom	Windows

You must open the wizard selection dialog with **CTRL+N**, **File > New > Other** or **New > Other** in the toolbar.

Select the **ResourceBundle File** under folder **Java** and click Next.



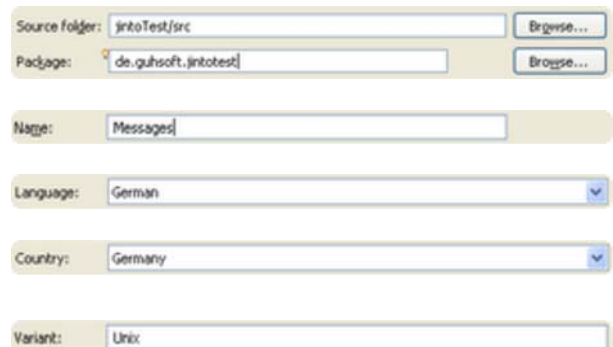
Fill in the **Source Folder** and the **Package** Fields.

Now enter the **Name** of the resourceBundle files.

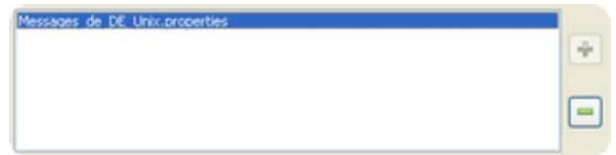
As next step we must select the **Language** of our first file.

Then we select the **Country** of the first file.

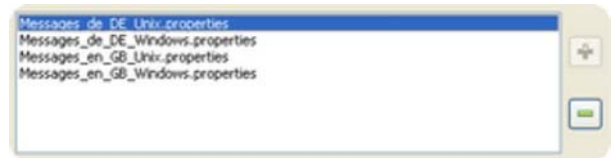
The next step is to write additional text into the **Variant** field if needed.



As last action for our first file we press the **add** button for adding the file into the list.



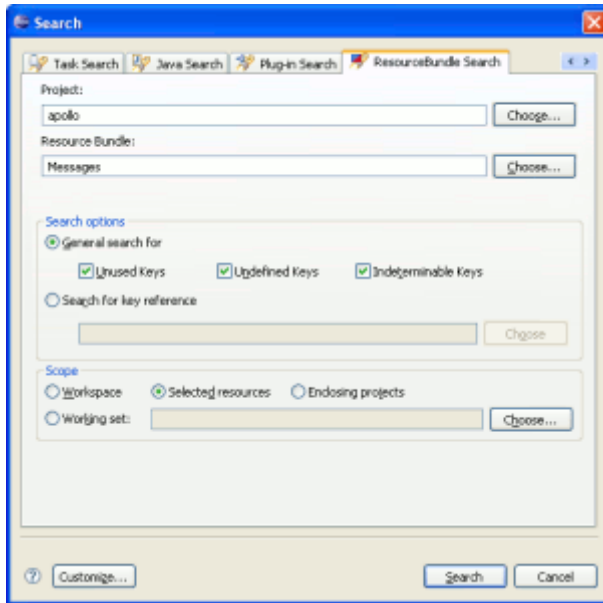
Now we repeat the last four steps until we have all files in the list (changing the country, language and variant settings appropriately).



As last step we press the **Finish** button and the files will be generated.



ResourceBundle Search



When you work on big projects with complex resource files, it may quickly happen that you forget to remove or add a key into the resourceBundle. For this reason JInto offers a search function with which you can search for unused or undefined keys in a resourceBundle.

You can also search for references of a key in Java files by clicking right on a key in the ResourceBundle editor and select 'Search by Reference'.

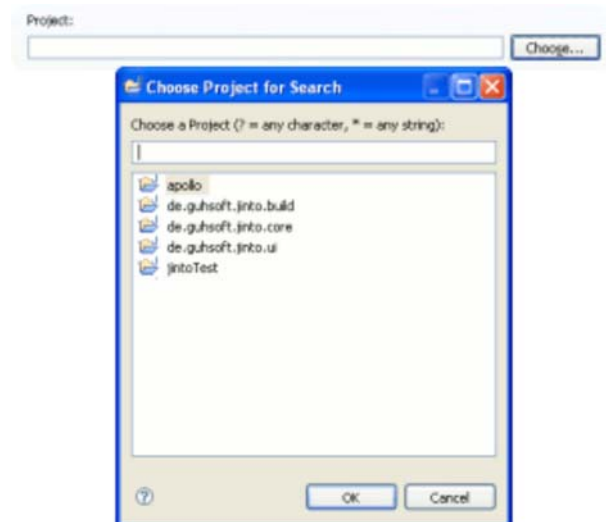
The following tutorial shows a sample ResourceBundle search.

You must first open the ResourceBundle search. You can do this by clicking **CTRL+H** in a ResourceBundle editor, or select **Search > ResourceBundle...**

If the currently selected item in eclipse is a ResourceBundle editor or a resource file, the ResourceBundle search automatically selects the ResourceBundle in the search dialog.

You can also select the ResourceBundle manually in the dialog. You first have to select the project where the ResourceBundle is located. Click on **Choose...** next to the project text field. Here you can select one of the project's which contains resource bundles.

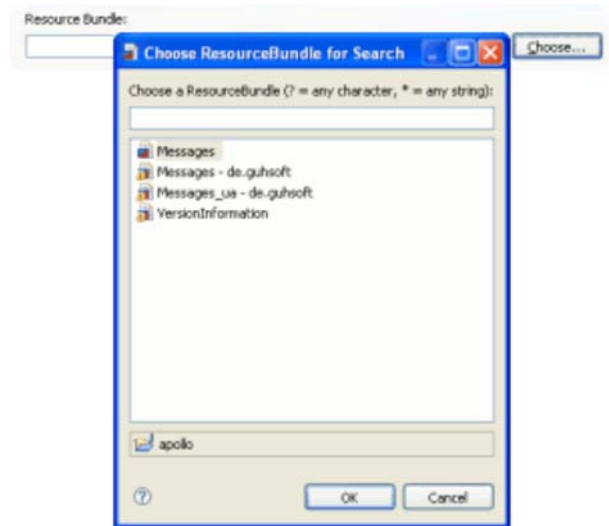
Tipp: use *CTRL+Space* to choose a project with auto completion.



After selecting a project you have to select the resourceBundle. Please open the resourceBundle dialog by clicking on **Choose...** next the resourceBundle text field. Here you can select one of the resource bundle in the selected project.

If the resourceBundle is not configured in the project (see [JInto Property Page](#)), an error is shown under the text field's.

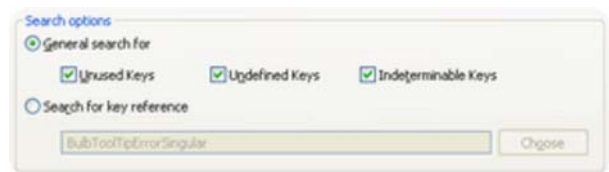
Tipp: use *CTRL+Space* to choose a resourceBundle with auto completion.



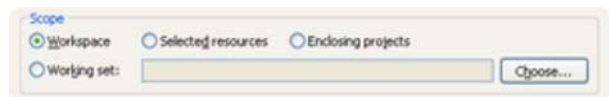
The next step is the search options selection. You can choose between **General search** or **Reference Search**. In the General Search you can search for unused, undefined and indeterminable keys.

Unused Keys are keys in the resourceBundle which are not used in the .java files searched, **Undefined Keys** are referenced in the .java files but don't exist in the resourceBundle. And **Indeterminable Keys** are references in .java file which the program cannot resolve (variables, anything that will only be known at runtime of the application).

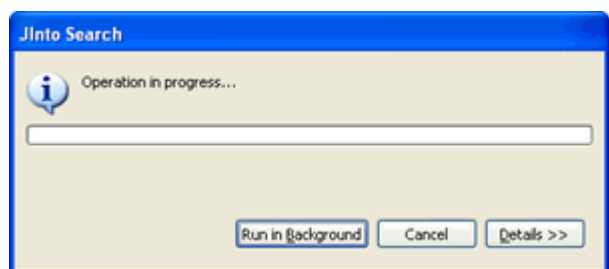
A Reference Search searches for all references of one key in the selected resource bundle. You can choose the key from a dialog or enter the text manually.



The last step of the search is the selection of the search scope. Here you can chose between Workspace, Selected Resources, Enclosing Projects and one or more Working Sets.

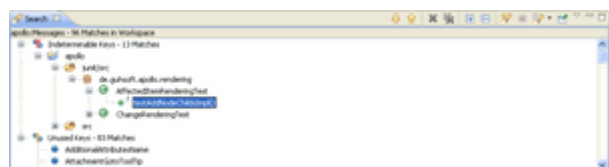


Now you can start the search. The result will be shown in the Eclipse Search view. The search can also be done in background.

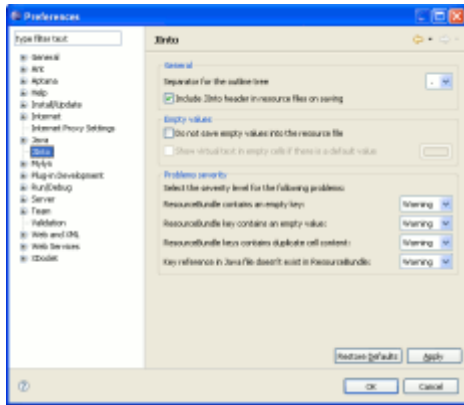


The result is structured in three groups: Unused Keys, Undefined Keys and Indeterminable Keys. If there is no result for one group or you haven't selected the group in the search dialog, there is no node in the Search view.

A **Reference Search** shows the java elements which references the key.



Preferences



The preferences page of JInto is separated into three categories. The first category is for general settings. The next category contains the settings for empty values and the third category is for problems settings.

General

Select the separator for the outline. The selected separator is used to separate the key's into a tree for better and faster navigation.

If you select **Include JInto header in resource files on saving**, a header comment is written in every *.properties file you're saving with JInto.

Empty values

If you don't want to save empty values into the *.properties file, you can select the **Do not save empty values into the resource file** option.

If you select the **Show virtual text in empty cells if there is a default value**, every empty cell displays the text from the default resource file if there is a default resource file.

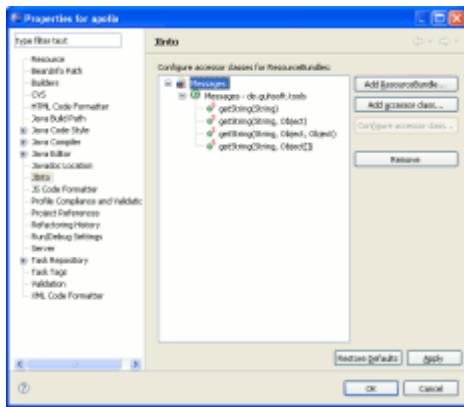
With the color button you can select the color of the virtual text.

Problems severity

In this category of the preferences dialog you can select the severity of the JInto problems. You can select **Ignore**, **Warning** or **Error** for every problem JInto checks in the background.

The first three problems are marked in the ResourceBundle Editor and the last problem will be marked in the Compilation Unit (only if JInto is [configured right](#)). JInto will also provide a quick fix for this problem.

Property Page

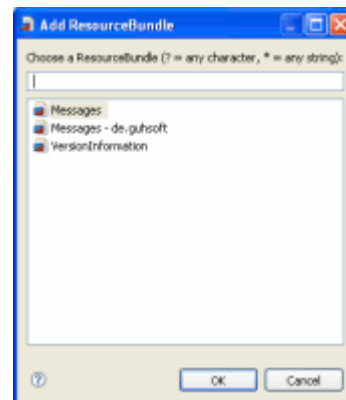


JInto also has a property page in the project properties. Here you can configure accessor methods for resourceBundles. This is needed to use the search or all the cool features in the [Java Editor](#).

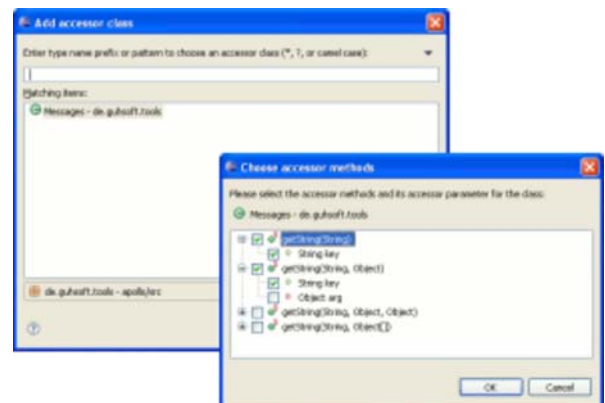
You first have to open the **Project Properties** and select the **JInto** property page.

Please click on the **Add ResourceBundle...** button to add a new resourceBundle configuration.

A Dialog where you can choose one of the resourceBundle's in the project is opened. Select the resourceBundle you want to configure.

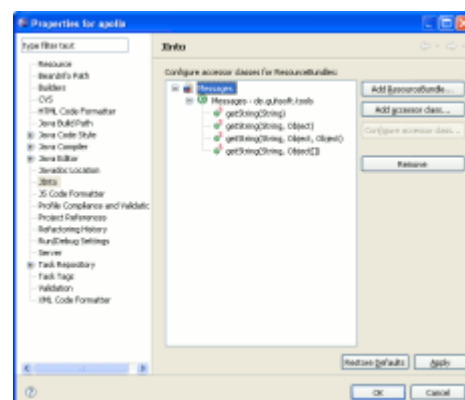


After selecting the resourceBundle, a dialog where you select the accessor class is shown. Please choose the class which contains the accessor method's for the selected resourceBundle. Then you see a method selection dialog, where you have to select the method and the parameter that will hold the key.



If you have configured a resourceBundle correct, you can see the configuration in the property page. With the button's in the right, you can add multiple accessor classes and/or methods to one resourceBundle...

Note: The resourceBundle and its accessor class(es) have to be in the same project.



Known issues

- Repaint bug in the table header in very wide tables under windows. Please see [Eclipse Bugzilla #79980](#)

Eclipse Public License - v 1.0

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS ECLIPSE PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. DEFINITIONS

"Contribution" means:

- a) in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and
- b) in the case of each subsequent Contributor:
 - i) changes to the Program, and
 - ii) additions to the Program;where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

"Contributor" means any person or entity that distributes the Program.

"Licensed Patents " mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

"Program" means the Contributions distributed in accordance with this Agreement.

"Recipient" means anyone who receives the Program under this Agreement, including all Contributors.

2. GRANT OF RIGHTS

- a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.
- b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.
- c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.
- d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

3. REQUIREMENTS

A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:

- a) it complies with the terms and conditions of this Agreement; and
- b) its license agreement:
 - i) effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;
 - ii) effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;
 - iii) states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and
 - iv) states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.

When the Program is made available in source code form:

- a) it must be made available under this Agreement; and
- b) a copy of this Agreement must be included with each copy of the Program.

Contributors may not remove or alter any copyright notices contained within the Program.

Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution.

4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. The Eclipse Foundation is the initial Agreement Steward. The Eclipse Foundation may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.