OccurrencesSearchGroup.java - ReadReferencesSearchGroup.java

```
/*******************************************************************************
 * Copyright (c) 2000, 2008 IBM Corporation and others.
 * All rights reserved. This program and the accompanying materials
 * are made available under the terms of the Eclipse Public License v1.0
 * which accompanies this distribution, and is available at
 * http://www.eclipse.org/legal/epl-v10.html
 *
 * Contributors:
 *     IBM Corporation - initial API and implementation
 *******************************************************************************/
package org.eclipse.jdt.ui.actions;



import org.eclipse.core.runtime.Assert;

import org.eclipse.jface.action.Action;
import org.eclipse.jface.action.IMenuListener;
import org.eclipse.jface.action.IMenuManager;
import org.eclipse.jface.action.MenuManager;

import org.eclipse.jface.viewers.ISelection;
import org.eclipse.jface.viewers.ISelectionChangedListener;
import org.eclipse.jface.viewers.ISelectionProvider;
import org.eclipse.jface.viewers.IStructuredSelection;

import org.eclipse.jface.text.IDocument;
import org.eclipse.jface.text.ITextSelection;

import org.eclipse.ui.IActionBars;
import org.eclipse.ui.IWorkbenchSite;
import org.eclipse.ui.PlatformUI;
import org.eclipse.ui.actions.ActionGroup;
import org.eclipse.ui.keys.IBindingService;

import org.eclipse.ui.texteditor.ITextEditorActionConstants;

import org.eclipse.jdt.core.ITypeRoot;

import org.eclipse.jdt.ui.IContextMenuConstants;
import org.eclipse.jdt.ui.JavaUI;

import org.eclipse.jdt.internal.ui.actions.SelectionConverter;
import org.eclipse.jdt.internal.ui.javaeditor.JavaEditor;
import org.eclipse.jdt.internal.ui.javaeditor.JavaTextSelection;
import org.eclipse.jdt.internal.ui.search.SearchMessages;


/**
 * Action group that adds the occurrences in file actions
 * to a context menu and the global menu bar.
 *
 * <p>
 * This class may be instantiated; it is not intended to be subclassed.
 * </p>
 *
```

```
/*******************************************************************************
 * Copyright (c) 2000, 2008 IBM Corporation and others.
 * All rights reserved. This program and the accompanying materials
 * are made available under the terms of the Eclipse Public License v1.0
 * which accompanies this distribution, and is available at
 * http://www.eclipse.org/legal/epl-v10.html
 *
 * Contributors:
 *     IBM Corporation - initial API and implementation
 *******************************************************************************/
package org.eclipse.jdt.ui.actions;

import java.util.Iterator;

import org.eclipse.core.runtime.Assert;

import org.eclipse.jface.action.IAction;

import org.eclipse.jface.action.IMenuManager;
import org.eclipse.jface.action.MenuManager;
import org.eclipse.jface.action.Separator;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.jface.viewers.ISelectionChangedListener;
import org.eclipse.jface.viewers.ISelectionProvider;
import org.eclipse.jface.viewers.IStructuredSelection;


import org.eclipse.ui.IActionBars;
import org.eclipse.ui.IWorkbenchSite;
import org.eclipse.ui.IWorkingSet;
import org.eclipse.ui.actions.ActionGroup;


import org.eclipse.ui.texteditor.ITextEditorActionConstants;



import org.eclipse.jdt.ui.IContextMenuConstants;


import org.eclipse.jdt.internal.ui.javaeditor.JavaEditor;

import org.eclipse.jdt.internal.ui.search.SearchMessages;
import org.eclipse.jdt.internal.ui.search.SearchUtil;

/**
 * Action group that adds the search for read references actions to a
 * context menu and the global menu bar.
 *
 * <p>
 * This class may be instantiated; it is not intended to be subclassed.
 * </p>
 *
```

OccurrencesSearchGroup.java - ReadReferencesSearchGroup.java

```
 * @since 3.1
 *
 * @noextend This class is not intended to be subclassed by clients.
 */
public class OccurrencesSearchGroup extends ActionGroup  {



    private IWorkbenchSite fSite;
    private JavaEditor fEditor;
    private IActionBars fActionBars;

    private String fGroupId;

    private FindOccurrencesInFileAction fOccurrencesInFileAction;
    private FindExceptionOccurrencesAction fExceptionOccurrencesAction;
    private FindImplementOccurrencesAction fFindImplementorOccurrencesAction;

    private FindBreakContinueTargetOccurrencesAction fBreakContinueTargetOccurrencesAction;
    private FindMethodExitOccurrencesAction fMethodExitOccurrencesAction;

    /**
     * Creates a new <code>ImplementorsSearchGroup</code>. The group
     * requires that the selection provided by the site's selection provider

     * is of type <code>org.eclipse.jface.viewers.IStructuredSelection</code>.
     *
     * @param site the view part that owns this action group
     */
    public OccurrencesSearchGroup(IWorkbenchSite site) {
        this(site, null);
    }

    /**
     * Creates a new <code>OccurrencesSearchGroup</code>. The group requires
     * that the selection provided by the given selection provider is of type
     * {@link IStructuredSelection}.
     *
     * @param site the site that will own the action group.
     * @param specialSelectionProvider the selection provider used instead of the
     *  sites selection provider.
     *
     * @since 3.4
     */
    public OccurrencesSearchGroup(IWorkbenchSite site, ISelectionProvider specialSelectionProvider) {
        fSite= site;
        fGroupId= IContextMenuConstants.GROUP_SEARCH;

        fOccurrencesInFileAction= new FindOccurrencesInFileAction(site);

fOccurrencesInFileAction.setActionDefinitionId(IJavaEditorActionDefinitionIds.SEARCH_OCCURRENCES_IN_FILE);
        // Need to reset the label

```

```
 * @since 2.0
 *
 * @noextend This class is not intended to be subclassed by clients.
 */
public class ReadReferencesSearchGroup extends ActionGroup  {

    private static final String MENU_TEXT= SearchMessages.group_readReferences;

    private IWorkbenchSite fSite;
    private JavaEditor fEditor;
    private IActionBars fActionBars;

    private String fGroupId;

    private FindReadReferencesAction fFindReadReferencesAction;
    private FindReadReferencesInProjectAction fFindReadReferencesInProjectAction;
    private FindReadReferencesInHierarchyAction fFindReadReferencesInHierarchyAction;
    private FindReadReferencesInWorkingSetAction fFindReadReferencesInWorkingSetAction;


    /**
     * Creates a new <code>ReadReferencesSearchGroup</code>. The group requires
     * that the selection provided by the site's selection provider is of type <code>
     * org.eclipse.jface.viewers.IStructuredSelection</code>.
     *
     * @param site the view part that owns this action group
     */
    public ReadReferencesSearchGroup(IWorkbenchSite site) {
        this(site, null);
    }

    /**
     * Creates a new <code>ReadReferencesSearchGroup</code>. The group requires
     * that the selection provided by the given selection provider is of type
     * {@link IStructuredSelection}.
     *
     * @param site the site that will own the action group.
     * @param specialSelectionProvider the selection provider used instead of the
     *  sites selection provider.
     *
     * @since 3.4
     */
    public ReadReferencesSearchGroup(IWorkbenchSite site, ISelectionProvider specialSelectionProvider) {
        fSite= site;
        fGroupId= IContextMenuConstants.GROUP_SEARCH;

        fFindReadReferencesAction= new FindReadReferencesAction(site);

fFindReadReferencesAction.setActionDefinitionId(IJavaEditorActionDefinitionIds.SEARCH_WRITE_ACCESS_IN_WORKSPACE);


```

OccurrencesSearchGroup.java - ReadReferencesSearchGroup.java

```
fOccurrencesInFileAction.setText(SearchMessages.Search_FindOccurrencesInFile_shortLabel);

        fExceptionOccurrencesAction= new FindExceptionOccurrencesAction(site);
fExceptionOccurrencesAction.setActionDefinitionId(IJavaEditorActionDefinitionIds.SEARCH_EXCEPTION_OCCURRENCES_IN_FILE);

        fFindImplementorOccurrencesAction= new FindImplementOccurrencesAction(site);
fFindImplementorOccurrencesAction.setActionDefinitionId(IJavaEditorActionDefinitionIds.SEARCH_IMPLEMENT_OCCURRENCES_IN_FILE);

        fBreakContinueTargetOccurrencesAction= new FindBreakContinueTargetOccurrencesAction(site);
fBreakContinueTargetOccurrencesAction.setActionDefinitionId(IJavaEditorActionDefinitionIds.SEARCH_BREAK_CONTINUE_TARGET_OCCURRENCES);

        fMethodExitOccurrencesAction= new FindMethodExitOccurrencesAction(site);

fMethodExitOccurrencesAction.setActionDefinitionId(IJavaEditorActionDefinitionIds.SEARCH_METHOD_EXIT_OCCURRENCES);
```

```
        // register the actions as selection listeners
        ISelectionProvider provider= specialSelectionProvider == null ? fSite.getSelectionProvider() : specialSelectionProvider;
        ISelection selection= provider.getSelection();
        registerAction(fOccurrencesInFileAction, provider, selection, specialSelectionProvider);
        registerAction(fExceptionOccurrencesAction, provider, selection, specialSelectionProvider);
        registerAction(fFindImplementorOccurrencesAction, provider, selection, specialSelectionProvider);
        registerAction(fBreakContinueTargetOccurrencesAction, provider, selection, specialSelectionProvider);
        registerAction(fMethodExitOccurrencesAction, provider, selection, specialSelectionProvider);
    }

    /**
     * Note: This constructor is for internal use only. Clients should not call this constructor.
     *
     * @param editor the Java editor
     *
     * @noreference This constructor is not intended to be referenced by clients.
     */
```

```
        fFindReadReferencesInProjectAction= new FindReadReferencesInProjectAction(site);
fFindReadReferencesInProjectAction.setActionDefinitionId(IJavaEditorActionDefinitionIds.SEARCH_READ_ACCESS_IN_PROJECT);

        fFindReadReferencesInHierarchyAction= new FindReadReferencesInHierarchyAction(site);
fFindReadReferencesInHierarchyAction.setActionDefinitionId(IJavaEditorActionDefinitionIds.SEARCH_WRITE_ACCESS_IN_HIERARCHY);

        fFindReadReferencesInWorkingSetAction= new FindReadReferencesInWorkingSetAction(site);
fFindReadReferencesInWorkingSetAction.setActionDefinitionId(IJavaEditorActionDefinitionIds.SEARCH_WRITE_ACCESS_IN_WORKING_SET);
```

```
        // register the actions as selection listeners
        ISelectionProvider provider= specialSelectionProvider == null ? fSite.getSelectionProvider() : specialSelectionProvider;
        ISelection selection= provider.getSelection();
        registerAction(fFindReadReferencesAction, provider, selection, specialSelectionProvider);
        registerAction(fFindReadReferencesInProjectAction, provider, selection, specialSelectionProvider);
        registerAction(fFindReadReferencesInHierarchyAction, provider, selection, specialSelectionProvider);
        registerAction(fFindReadReferencesInWorkingSetAction, provider, selection, specialSelectionProvider);
    }

    /**
     * Note: This constructor is for internal use only. Clients should not call this constructor.
     *
     * @param editor the Java editor
     *
     * @noreference This constructor is not intended to be referenced by clients.
     */
```

OccurrencesSearchGroup.java - ReadReferencesSearchGroup.java

```
132     public OccurrencesSearchGroup(JavaEditor editor) {
133         fEditor= editor;
134         fSite= fEditor.getSite();
135         fGroupId= ITextEditorActionConstants.GROUP_FIND;
136
137         fOccurrencesInFileAction= new FindOccurrencesInFileAction(fEditor);
138
    fOccurrencesInFileAction.setActionDefinitionId(IJavaEditorActionDefinitionIds.SEARCH_OCCURRENCES_IN_FILE);
139         // Need to reset the label
140
    fOccurrencesInFileAction.setText(SearchMessages.Search_FindOccurrencesInFile_shortLabel);
141         fEditor.setAction("SearchOccurrencesInFile", fOccurrencesInFileAction);
    //$NON-NLS-1$
142
143         fExceptionOccurrencesAction= new FindExceptionOccurrencesAction(fEditor);
144
    fExceptionOccurrencesAction.setActionDefinitionId(IJavaEditorActionDefinitionIds.SEARCH_EXCEPTION_OCCURRENCES_IN_FILE);
145         fEditor.setAction("SearchExceptionOccurrences",
    fExceptionOccurrencesAction); //$NON-NLS-1$
146
147         fFindImplementorOccurrencesAction= new
    FindImplementOccurrencesAction(fEditor);
148
    fFindImplementorOccurrencesAction.setActionDefinitionId(IJavaEditorActionDefinitionIds.SEARCH_IMPLEMENT_OCCURRENCES_IN_FILE);
149         fEditor.setAction("SearchImplementOccurrences",
    fFindImplementorOccurrencesAction); //$NON-NLS-1$
150
151         fBreakContinueTargetOccurrencesAction= new
    FindBreakContinueTargetOccurrencesAction(fEditor);
152
    fBreakContinueTargetOccurrencesAction.setActionDefinitionId(IJavaEditorActionDefinitionIds.SEARCH_BREAK_CONTINUE_TARGET_OCCURRENCES);
153         fEditor.setAction("BreakContinueTargetOccurrences",
    fBreakContinueTargetOccurrencesAction); //$NON-NLS-1$
154
155         fMethodExitOccurrencesAction= new FindMethodExitOccurrencesAction(fEditor);
156
    fMethodExitOccurrencesAction.setActionDefinitionId(IJavaEditorActionDefinitionIds.SEARCH_METHOD_EXIT_OCCURRENCES);
157         fEditor.setAction("ExitOccurrencesAction", fMethodExitOccurrencesAction);
    //$NON-NLS-1$
158
159//
160//      ISelectionProvider provider= fSite.getSelectionProvider();
```

```
119     public ReadReferencesSearchGroup(JavaEditor editor) {
120         fEditor= editor;
121         fSite= fEditor.getSite();
122         fGroupId= ITextEditorActionConstants.GROUP_FIND;
123
124         fFindReadReferencesAction= new FindReadReferencesAction(fEditor);
125
    fFindReadReferencesAction.setActionDefinitionId(IJavaEditorActionDefinitionIds.SEARCH_WRITE_ACCESS_IN_WORKSPACE);
126         fEditor.setAction("SearchReadAccessInWorkspace",
    fFindReadReferencesAction); //$NON-NLS-1$
127
128         fFindReadReferencesInProjectAction= new
    FindReadReferencesInProjectAction(fEditor);
129
    fFindReadReferencesInProjectAction.setActionDefinitionId(IJavaEditorActionDefinitionIds.SEARCH_READ_ACCESS_IN_PROJECT);
130         fEditor.setAction("SearchReadAccessInProject",
    fFindReadReferencesInProjectAction); //$NON-NLS-1$
131
132         fFindReadReferencesInHierarchyAction= new
    FindReadReferencesInHierarchyAction(fEditor);
133
    fFindReadReferencesInHierarchyAction.setActionDefinitionId(IJavaEditorActionDefinitionIds.SEARCH_WRITE_ACCESS_IN_HIERARCHY);
134         fEditor.setAction("SearchReadAccessInHierarchy",
    fFindReadReferencesInHierarchyAction); //$NON-NLS-1$
135
136         fFindReadReferencesInWorkingSetAction= new
    FindReadReferencesInWorkingSetAction(fEditor);
137
    fFindReadReferencesInWorkingSetAction.setActionDefinitionId(IJavaEditorActionDefinitionIds.SEARCH_WRITE_ACCESS_IN_WORKING_SET);
138         fEditor.setAction("SearchReadAccessInWorkingSet",
    fFindReadReferencesInWorkingSetAction); //$NON-NLS-1$
```

OccurrencesSearchGroup.java - ReadReferencesSearchGroup.java

```
161//      ISelection selection= provider.getSelection();
162//
163//      registerAction(fOccurrencesInFileAction, provider, selection);
164//      registerAction(fExceptionOccurrencesAction, provider, selection);
165//      registerAction(fFindImplementorOccurrencesAction, provider, selection);
166    }
167
168    private void registerAction(SelectionDispatchAction action, ISelectionProvider provider, ISelection selection, ISelectionProvider specialSelectionProvider) {
169        action.update(selection);
170        provider.addSelectionChangedListener(action);
171        if (specialSelectionProvider != null)
172            action.setSpecialSelectionProvider(specialSelectionProvider);
173    }
174
175    /*
176     * Method declared on ActionGroup.
177     */
178    public void fillContextMenu(IMenuManager manager) {
179        String menuText= SearchMessages.group_occurrences;
180        String shortcut= getShortcutString();
181        if (shortcut != null) {
182            menuText= menuText + '¥t' + shortcut;
183        }
184
185        MenuManager javaSearchMM= new MenuManager(menuText, IContextMenuConstants.GROUP_SEARCH);
186        javaSearchMM.add(new Action() {
187        });
188        javaSearchMM.addMenuListener(new IMenuListener() {
189            public void menuAboutToShow(IMenuManager mm) {
190                mm.removeAll();
191                updateActionsInJavaEditor();
192                addAction(fOccurrencesInFileAction, mm);
193                addAction(fFindImplementorOccurrencesAction, mm);
194                addAction(fExceptionOccurrencesAction, mm);
195                addAction(fMethodExitOccurrencesAction, mm);
196                addAction(fBreakContinueTargetOccurrencesAction, mm);
197                if (mm.isEmpty()) {
198                    mm.add(new Action(SearchMessages.group_occurrences_quickMenu_noEntriesAvailable) {
199                        public boolean isEnabled() {
200                            return false;
201                        }
202                    });
203                }
204            }
205
206            private void addAction(Action action, IMenuManager mm) {
207                if (action.isEnabled())
208                    mm.add(action);
209            }
210        });
211        manager.appendToGroup(fGroupId, javaSearchMM);
212    }
213
```

```
139    }
140
141    private void registerAction(SelectionDispatchAction action, ISelectionProvider provider, ISelection selection, ISelectionProvider specialSelectionProvider) {
142        action.update(selection);
143        provider.addSelectionChangedListener(action);
144        if (specialSelectionProvider != null)
145            action.setSpecialSelectionProvider(specialSelectionProvider);
146    }
147
148    private void addAction(IAction action, IMenuManager manager) {
149        if (action.isEnabled()) {
150            manager.add(action);
151        }
152    }
153
```

OccurrencesSearchGroup.java - ReadReferencesSearchGroup.java

```
214     private void updateActionsInJavaEditor() {
215         if (fEditor == null)
216             return;
217
218         ITypeRoot element= SelectionConverter.getInput(fEditor);
219         if (element == null)
220             return;
221
222         ITextSelection textSelection= (ITextSelection) fEditor.getSelectionProvider().getSelection();
223         IDocument document= JavaUI.getDocumentProvider().getDocument(fEditor.getEditorInput());
224         JavaTextSelection javaSelection= new JavaTextSelection(element, document, textSelection.getOffset(), textSelection.getLength());
225
226         fExceptionOccurrencesAction.update(javaSelection);
227         fOccurrencesInFileAction.update(javaSelection);
228         fFindImplementorOccurrencesAction.update(javaSelection);
229         fBreakContinueTargetOccurrencesAction.update(javaSelection);
230         fMethodExitOccurrencesAction.update(javaSelection);
231     }
232
233     private String getShortcutString() {
234         IBindingService bindingService= (IBindingService)PlatformUI.getWorkbench().getAdapter(IBindingService.class);
235         if (bindingService == null)
236             return null;
237         return bindingService.getBestActiveBindingFormattedFor(IJavaEditorActionDefinitionIds.SEARCH_OCCURRENCES_IN_FILE_QUICK_MENU);
238     }
239
240     /*
241      * Method declared on ActionGroup.
242      */
```

```
154     private void addWorkingSetAction(IWorkingSet[] workingSets, IMenuManager manager) {
155         FindAction action;
156         if (fEditor != null)
157             action= new WorkingSetFindAction(fEditor, new FindReadReferencesInWorkingSetAction(fEditor, workingSets), SearchUtil.toString(workingSets));
158         else
159             action= new WorkingSetFindAction(fSite, new FindReadReferencesInWorkingSetAction(fSite, workingSets), SearchUtil.toString(workingSets));
160         action.update(getContext().getSelection());
161         addAction(action, manager);
162     }
163
164
165     /* (non-Javadoc)
166      * Method declared on ActionGroup.
167      */
168     public void fillContextMenu(IMenuManager manager) {
169         MenuManager javaSearchMM= new MenuManager(MENU_TEXT, IContextMenuConstants.GROUP_SEARCH);
170         addAction(fFindReadReferencesAction, javaSearchMM);
171         addAction(fFindReadReferencesInProjectAction, javaSearchMM);
172         addAction(fFindReadReferencesInHierarchyAction, javaSearchMM);
173
174         javaSearchMM.add(new Separator());
175
176         Iterator iter= SearchUtil.getLRUWorkingSets().sortedIterator();
177         while (iter.hasNext()) {
178             addWorkingSetAction((IWorkingSet[]) iter.next(), javaSearchMM);
179         }
180         addAction(fFindReadReferencesInWorkingSetAction, javaSearchMM);
181
182         if (!javaSearchMM.isEmpty())
183             manager.appendToGroup(fGroupId, javaSearchMM);
184     }
185
186     /*
187      * Method declared on ActionGroup.
188      */
```

OccurrencesSearchGroup.java - ReadReferencesSearchGroup.java

```
243     public void fillActionBars(IActionBars actionBars) {
244         Assert.isNotNull(actionBars);
245         super.fillActionBars(actionBars);
246         fActionBars= actionBars;
247         updateGlobalActionHandlers();
248     }
249
250     /*
251      * Method declared on ActionGroup.
252      */
253     public void dispose() {
254         ISelectionProvider provider= fSite.getSelectionProvider();
255         if (provider != null) {
256             disposeAction(fFindImplementorOccurrencesAction, provider);
257             disposeAction(fExceptionOccurrencesAction, provider);
258             disposeAction(fOccurrencesInFileAction, provider);
259             disposeAction(fMethodExitOccurrencesAction, provider);
260             disposeAction(fBreakContinueTargetOccurrencesAction, provider);
261         }
262         super.dispose();
263         fFindImplementorOccurrencesAction= null;
264         fExceptionOccurrencesAction= null;
265         fOccurrencesInFileAction= null;
266         fMethodExitOccurrencesAction= null;
267         fBreakContinueTargetOccurrencesAction= null;
268         updateGlobalActionHandlers();
269     }
270
271     private void updateGlobalActionHandlers() {
272         if (fActionBars != null) {
273             fActionBars.setGlobalActionHandler(JdtActionConstants.FIND_OCCURRENCES_IN_FILE, fOccurrencesInFileAction);
274             fActionBars.setGlobalActionHandler(JdtActionConstants.FIND_EXCEPTION_OCCURRENCES, fExceptionOccurrencesAction);
275             fActionBars.setGlobalActionHandler(JdtActionConstants.FIND_IMPLEMENT_OCCURRENCES, fFindImplementorOccurrencesAction);
276             fActionBars.setGlobalActionHandler(JdtActionConstants.FIND_BREAK_CONTINUE_TARGET_OCCURRENCES, fBreakContinueTargetOccurrencesAction);
277             fActionBars.setGlobalActionHandler(JdtActionConstants.FIND_METHOD_EXIT_OCCURRENCES, fMethodExitOccurrencesAction);
278
279         }
280     }
281
282     private void disposeAction(ISelectionChangedListener action, ISelectionProvider provider) {
283         if (action != null)
284             provider.removeSelectionChangedListener(action);
285     }
286}
```

```
189     public void fillActionBars(IActionBars actionBars) {
190         Assert.isNotNull(actionBars);
191         super.fillActionBars(actionBars);
192         fActionBars= actionBars;
193         updateGlobalActionHandlers();
194     }
195
196     /*
197      * Method declared on ActionGroup.
198      */
199     public void dispose() {
200         ISelectionProvider provider= fSite.getSelectionProvider();
201         if (provider != null) {
202             disposeAction(fFindReadReferencesAction, provider);
203             disposeAction(fFindReadReferencesInProjectAction, provider);
204             disposeAction(fFindReadReferencesInHierarchyAction, provider);
205             disposeAction(fFindReadReferencesInWorkingSetAction, provider);
206         }
207         fFindReadReferencesAction= null;
208         fFindReadReferencesInProjectAction= null;
209         fFindReadReferencesInHierarchyAction= null;
210         fFindReadReferencesInWorkingSetAction= null;
211         updateGlobalActionHandlers();
212         super.dispose();
213     }
214
215     private void updateGlobalActionHandlers() {
216         if (fActionBars != null) {
217             fActionBars.setGlobalActionHandler(JdtActionConstants.FIND_READ_ACCESS_IN_WORKSPACE, fFindReadReferencesAction);
218             fActionBars.setGlobalActionHandler(JdtActionConstants.FIND_READ_ACCESS_IN_PROJECT, fFindReadReferencesInProjectAction);
219             fActionBars.setGlobalActionHandler(JdtActionConstants.FIND_READ_ACCESS_IN_HIERARCHY, fFindReadReferencesInHierarchyAction);
220             fActionBars.setGlobalActionHandler(JdtActionConstants.FIND_READ_ACCESS_IN_WORKING_SET, fFindReadReferencesInWorkingSetAction);
221         }
222     }
223
224     private void disposeAction(ISelectionChangedListener action, ISelectionProvider provider) {
225         if (action != null)
226             provider.removeSelectionChangedListener(action);
227     }
228}
```