JFace – Content Assist – Two Traces
Use Full Trace (No Slicing)

Supporting Applications : [Example Java Editor, JSP Editor]

```
import org.eclipse.jface.text.contentassist.IContentAssistant;
import org.eclipse.jface.text.DefaultPositionUpdater;
import org.eclipse.jface.action.IAction;
import org.eclipse.swt.graphics.Image;
import org.eclipse.jface.text.source.IAnnotationHover;
import org.eclipse.jface.text.contentassist.IContentAssistProcessor;
import org.eclipse.jface.viewers.ISelectionChangedListener;
import org.eclipse.jface.text.IDocumentExtension3;
import org.eclipse.swt.graphics.Color;
import org.eclipse.jface.viewers.ISelectionProvider;
import org.eclipse.jface.text.ITextHover;
import org.eclipse.jface.text.DocumentCommand;
import org.eclipse.jface.action.IMenuManager;
import org.eclipse.jface.text.contentassist.IContextInformation;
import org.eclipse.jface.text.TextAttribute;
import org.eclipse.jface.text.contentassist.CompletionProposal;
import org.eclipse.jface.text.contentassist.ContentAssistant;
import org.eclipse.jface.text.IDocument;
import org.eclipse.jface.action.Separator;
import org.eclipse.jface.action.Action;
import org.eclipse.jface.text.ITextViewer;
import org.eclipse.jface.text.IAutoEditStrategy;
import org.eclipse.jface.text.source.ISourceViewer;
import org.eclipse.jface.text.source.SourceViewerConfiguration;
import org.eclipse.jface.text.contentassist.ICompletionProposal;
import org.eclipse.jface.text.IDocumentPartitioner;

public class AppAutoEditStrategy
implements IAutoEditStrategy  {

      public void customizeDocumentCommand(IDocument,DocumentCommand) {
      }
}

public class AppContentAssistProcessor
implements IContentAssistProcessor  {

      public ICompletionProposal[] computeCompletionProposals(ITextViewer,int) {
            CompletionProposal completionProposal = new CompletionProposal(String,int,int,int)||
                                                      (String,int,int,int,Image,String,IContextInformation,String);
      }

      public char[] getCompletionProposalAutoActivationCharacters() {
      }

      public String getErrorMessage() {
      }

      public char[] getContextInformationAutoActivationCharacters() {
      }
}

public class AppSourceViewerConfiguration
extends SourceViewerConfiguration {

      public void getTextHover((ISourceViewer,String):(ITextHover)||(ISourceViewer,String,int):(ITextHover)) {
            AppTextHover appTextHover = new AppTextHover();
      }

      public String getConfiguredDocumentPartitioning(ISourceViewer) {
      }

      public int getTabWidth(ISourceViewer) {
      }

      public IAnnotationHover getAnnotationHover(ISourceViewer) {
            AppAnnotationHover appAnnotationHover = new AppAnnotationHover();
      }

      public IContentAssistant getContentAssistant(ISourceViewer) {
            AppContentAssistProcessor appContentAssistProcessor = new AppContentAssistProcessor();
            String string = appSourceViewerConfiguration.getConfiguredDocumentPartitioning(ISourceViewer);
            ContentAssistant contentAssistant = new ContentAssistant();
            contentAssistant.setDocumentPartitioning(string);
            contentAssistant.setContentAssistProcessor(appContentAssistProcessor);  // MAY REPEAT!
            contentAssistant.enableAutoActivation(boolean);
            contentAssistant.setAutoActivationDelay(int);
            contentAssistant.setProposalPopupOrientation(int);
            contentAssistant.setContextInformationPopupOrientation(int);
      }

      public String[] getIndentPrefixes(ISourceViewer,String) {
      }

      public IAutoEditStrategy[] getAutoEditStrategies(ISourceViewer,String) {
      }
}

public class AppAction
extends Action {

      public  AppAction() {
            appAction.setEnabled(boolean);
      }
}

public class AppTextHover
implements ITextHover  {
}

public class AppAnnotationHover
implements IAnnotationHover  {
}

public class AppSelectionChangedListener
```

```
implements ISelectionChangedListener  {
}

public class SomeClass {

      public void someMethod() {
            IDocumentPartitioner.connect(IDocument);
            TextAttribute textAttribute = new TextAttribute(Color)||(Color,Color,int);
            AppSelectionChangedListener appSelectionChangedListener = new AppSelectionChangedListener();
            ISelectionProvider.addSelectionChangedListener(appSelectionChangedListener);  // MAY REPEAT!
            Separator separator = new Separator(String)||();      // REPEATED!
            AppSourceViewerConfiguration appSourceViewerConfiguration = new AppSourceViewerConfiguration();
            IMenuManager imenuManager = IMenuManager.findMenuUsingPath(String);
            imenuManager.add(separator);  // REPEATED!
            DefaultPositionUpdater defaultPositionUpdater = new DefaultPositionUpdater(String);
            int app_int1 = IDocument.getLineOffset(int);
            IAction.setActionDefinitionId(String);    // REPEATED!
            AppAction appAction = new AppAction();
            IDocumentExtension3.setDocumentPartitioner(String,IDocumentPartitioner);
      }
}
```

Description of False Negatives:

We have one false negative because of not calling the following instruction since it is in org.eclipse.ui, not in JFace:
`TextEditor.setSourceViewerConfiguration(appSourceViewerConfiguration);`