

## Eclipse 3.3

### - org.eclipse.ui.internal.ide.actions.OpenLocalFileAction.java

```
public void run() {
    FileDialog dialog = new FileDialog(window.getShell(), SWT.OPEN | SWT.MULTI);
    dialog.setText(IDEWorkbenchMessages.OpenLocalFileAction_title);
    dialog.setFilterPath(filterPath);
    dialog.open();
    String[] names = dialog.getFileNames();

    if (names != null) {
        filterPath = dialog.getFilterPath();

        int numberOfFilesNotFound = 0;
        StringBuffer notFound = new StringBuffer();
        for (int i = 0; i < names.length; i++) {
            IFileStore fileStore = EFS.getLocalFileSystem().getStore(new Path(filterPath));
            fileStore = fileStore.getChild(names[i]);
            if (!fileStore.fetchInfo().isDirectory() && fileStore.fetchInfo().exists()) {
                IWorkbenchPage page = window.getActivePage();
                try {
                    IDE.openEditorOnFileStore(page, fileStore);
                } catch (PartInitException e) {
                    String msg = NLS.bind(IDEWorkbenchMessages.OpenLocalFileAction_message_e
                    -----
                }
            }
        }
    }
}
```

### - org.eclipse.ui.ide.IDE.java

```
/* @see org.eclipse.ui.IWorkbenchPage#openEditor(IEditorInput, String)
 * @since 3.3
 */
public static IEditorPart openEditorOnFileStore(IWorkbenchPage page, IFileStore fileStore)
    //sanity checks
    if (page == null) {
        throw new IllegalArgumentException();
    }

    IEditorInput input = getEditorInput(fileStore);
    String editorId = getEditorId(fileStore);

    // open the editor on the file
    return page.openEditor(input, editorId);
}
```

### - org.eclipse.ui.ide.IDE.java

```

    * @return the id of the appropriate editor
    * @since 3.3
    */
    private static String getEditorId(IFileStore fileStore) {
        IEditorDescriptor descriptor;
        try {
            descriptor = IDE.getEditorDescriptor(fileStore.getName());
        } catch (PartInitException e) {
            return null;
        }
        if (descriptor != null)
            return descriptor.getId();
        return null;
    }

```

- org.eclipse.ui.ide.IDE.java

```

    * @throws PartInitException
    *         if no editor can be found
    * @since 3.1
    */
    public static IEditorDescriptor getEditorDescriptor(String name)
        throws PartInitException {
        return getEditorDescriptor(name, true);
    }

```

/\*\*

- org.eclipse.ui.ide.IDE.java

```

    *         if no editor can be found
    * @since 3.1
    */
    public static IEditorDescriptor getEditorDescriptor(String name,
        boolean inferContentType) throws PartInitException {

        if (name == null) {
            throw new IllegalArgumentException();
        }

        IContentType contentType = inferContentType ? Platform
            .getContentTypeManager().findContentTypeFor(name) : null;
        IEditorRegistry editorReg = PlatformUI.getWorkbench()
            .getEditorRegistry();

        return getEditorDescriptor(name, editorReg, editorReg.getDefaultEditor(
            name, contentType));
    }

```

- org.eclipse.core.internal.content.ContentTypeMatcher

```

public IContentType findContentTypeFor(String fileName) {
    // basic implementation just gets all content types
    ContentTypeCatalog currentCatalog = getCatalog();
    IContentType[] associated = currentCatalog.findContentTypesFor(this, fileName);
    return associated.length == 0 ? null : new ContentTypeHandler((ContentType) associated[0],
}

```

### - org.eclipse.core.internal.content.ContentTypeCatalog

```

IContentType[] findContentTypesFor(ContentTypeMatcher matcher, final String fileName) {
    IContentType[] selected = concat(internalFindContentTypesFor(matcher, fileName, policyConstantGeneralIsBetter));
    // give the policy a chance to change the results
    ISelectionPolicy policy = matcher.getPolicy();
    if (policy != null)
        selected = applyPolicy(policy, selected, true, false);
    return selected;
}

```

## Eclipse 3.2

### - org.eclipse.ui.internal.editors.text.OpenExternalFileAction.java

```

public void run() {
    FileDialog dialog= new FileDialog(fWindow.getShell(), SWT.OPEN | SWT.MULTI);
    dialog.setText(TextEditorMessages.OpenExternalFileAction_title);
    dialog.setFilterPath(fFilterPath);
    dialog.open();
    String[] names= dialog.getFileNames();

    if (names != null) {
        fFilterPath= dialog.getFilterPath();

        int numberOfFilesNotFound= 0;
        StringBuffer notFound= new StringBuffer();
        for (int i= 0; i < names.length; i++) {
            IFileStore fileStore= EFS.getLocalFileSystem().getStore(new Path(fFilterPath)
            fileStore= fileStore.getChild(names[i]);
            if (!fileStore.fetchInfo().isDirectory() && fileStore.fetchInfo().exists()) {
                IEditorInput input= createEditorInput(fileStore);
                String editorId= getEditorId(fileStore);
                IWorkbenchPage page= fWindow.getActivePage();
                try {
                    page.openEditor(input, editorId);
                } catch (PartInitException e) {
                    EditorsPlugin.log(e.getStatus());
                    String msg= NLSUtility.format(TextEditorMessages.OpenExternalFileActi
                    MessageDialog.openError(fWindow.getShell(), TextEditorMessages.OpenEx
                }
            } else {

```

### - org.eclipse.ui.internal.editors.text.OpenExternalFileAction.java

```

private String getEditorId(IFileStore file) {
    IWorkbench workbench= fWindow.getWorkbench();
    IEditorRegistry editorRegistry= workbench.getEditorRegistry();
    IEditorDescriptor descriptor= editorRegistry.getDefaultEditor(file.getName(), getContentType(file));

    // check the OS for in-place editor (OLE on Win32)
    if (descriptor == null && editorRegistry.isSystemInPlaceEditorAvailable(file.getName()))
        descriptor= editorRegistry.findEditor(IEditorRegistry.SYSTEM_INPLACE_EDITOR_ID);

    // check the OS for external editor
    if (descriptor == null && editorRegistry.isSystemExternalEditorAvailable(file.getName()))
        descriptor= editorRegistry.findEditor(IEditorRegistry.SYSTEM_EXTERNAL_EDITOR_ID);

    if (descriptor != null)
        return descriptor.getId();

    return EditorsUI.DEFAULT_TEXT_EDITOR_ID;
}

```

- org.eclipse.ui.internal.editors.text.OpenExternalFileAction.java

```

private IContentType getContentType (IFileStore fileStore) {
    if (fileStore == null)
        return null;

    InputStream stream= null;
    try {
        stream= fileStore.openInputStream(EFS.NONE, null);
        return Platform.getContentTypeManager().findContentTypeFor(stream, fileStore.getName());
    } catch (IOException x) {
        EditorsPlugin.log(x);
        return null;
    } catch (CoreException x) {
        // Do not log FileNotFoundException (no access)
        if (!(x.getStatus().getException() instanceof FileNotFoundException))
            EditorsPlugin.log(x);

        return null;
    } finally {
        try {
            if (stream != null)
                stream.close();
        } catch (IOException x) {
            EditorsPlugin.log(x);
        }
    }
}

```

- org.eclipse.core.internal.content.ContentTypeMatcher

```

/**
 * @see IContentTypeMatcher
 */
public IContentType findContentTypeFor(InputStream contents, String fileName) throws IOException {
    ContentTypeCatalog currentCatalog = getCatalog();
    IContentType[] all = currentCatalog.findContentTypesFor(this, contents, fileName);
    return all.length > 0 ? new ContentTypeHandler((ContentType) all[0], currentCatalog.getGeneration()
}

```



## - org.eclipse.core.internal.content.ContentTypeCatalog

```
IContentType[] findContentTypesFor(ContentTypeMatcher matcher, InputStream contents, String fileName)
    final ILazySource buffer = ContentTypeManager.readBuffer(contents);
    IContentType[] selected = internalFindContentTypesFor(matcher, buffer, fileName, true);
    // give the policy a chance to change the results
    ISelectionPolicy policy = matcher.getPolicy();
    if (policy != null)
        selected = applyPolicy(policy, selected, fileName != null, true);
    return selected;
}
```

## - org.eclipse.core.internal.content.ContentTypeCatalog

```
private IContentType[] internalFindContentTypesFor(ContentTypeMatcher matcher, ILazySource buffer, String fileName,
    final IContentType[][] subset;
    final Comparator validPolicy;
    Comparator indeterminatePolicy;
    if (fileName == null) {
        // we only have a single array, by need to provide a two-dimensional, 2-element array
        subset = new IContentType[][] {getAllContentTypes(), NO_CONTENT_TYPES};
        indeterminatePolicy = policyConstantGeneralIsBetter;
        validPolicy = policyConstantSpecificIsBetter;
    } else {
        subset = internalFindContentTypesFor(matcher, fileName, policyLexicographical);
        indeterminatePolicy = policyGeneralIsBetter;
        validPolicy = policySpecificIsBetter;
    }
    int total = subset[0].length + subset[1].length;
    if (total == 0)
        // don't do further work if subset is empty
        return NO_CONTENT_TYPES;
    if (!forceValidation && total == 1) {
        // do not do validation if not forced and only one was found (caller will validate later)
        IContentType[] found = subset[0].length == 1 ? subset[0] : subset[1];
        // bug 100032 - ignore binary content type if contents are text
        if (!buffer.isText())
            // binary buffer, caller can call the describer with no risk
            return found;
        // text buffer, need to check describer
        IContentDescriber describer = ((ContentType) found[0]).getDescriber();
        if (describer == null || describer instanceof ITextContentDescriber)
            // no describer or text describer, that is fine
            return found;
        // only eligible content type is binary and contents are text, ignore it
        return NO_CONTENT_TYPES;
    }
    return internalFindContentTypesFor(buffer, subset, validPolicy, indeterminatePolicy);
}
```