Eclipse (2/3)

Deepak Dhungana
dhungana@ase.jku.at
Institute for System Engineering and Automation

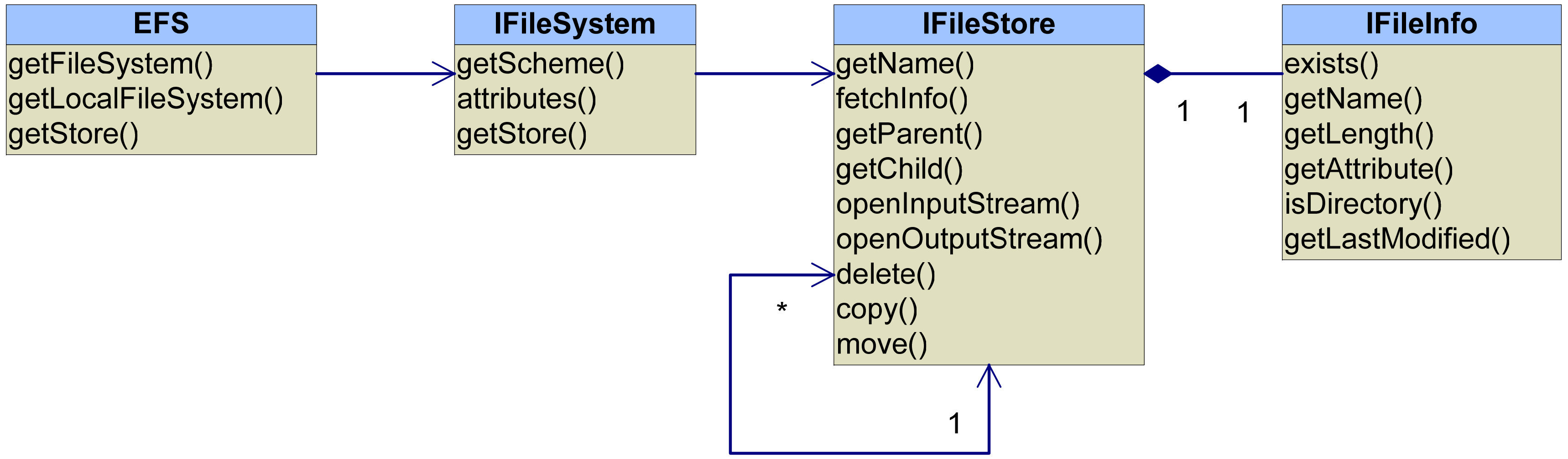
Thomas Wuerthinger
wuerthinger@ssw.jku.at
Institute for System Software

Johannes Kepler University Linz, Austria
http://www.jku.at

Eclipse File System (EFS)

- Abstraction of physical file location
  - java.io.File is limited to files on disk
  - EFS hides physical location of a resource
  - Uses URI syntax with protocols
  - No dependencies on other parts of Eclipse
- Default implementations
  - LocalFileSystem
    - The well-known “file:” protocol
    - Can fully replace java.io.File objects
  - NullFileSystem
    - Simulates an unmounted file system
    - Does not return any real data
- New implementations can be added
  - Specified via extension point

Eclipse File System (EFS)



Eclipse File System (EFS)

- Use existing file systems
  - Static methods of class EFS as entry points
  - Get the file store for a URI
    - EFS.getStore(URI)
  - Navigate using parent and children of a file store
  - Access properties via IFileInfo
  - Open input and output streams

- Define your own file system

```
<extension
    id="org.eclipse.core.filesystem.local"
    point="org.eclipse.core.filesystem.filesystems">
  <filesystem scheme="file">
    <run class=
        "org.eclipse.core.internal.filesystem.local.LocalFileSystem"/>
  </filesystem>
</extension>
```

Extension point

Name of protocol

Implementation class

Eclipse File System (EFS) – Example

```
FileDialog saveDialog = new FileDialog(..., SWT.SAVE);
saveDialog.setText("Save Drawing");
saveDialog.setFilterNames(new String[] {
    "DrawIt Drawing (*.dit)", "All Files (*.*)" });
saveDialog.setFilterExtensions(new String[] {
    "*.dit", "*.*" });

String fileName = saveDialog.open();
if (fileName == null) {
  return;
}

IPath path = new Path(fileName);
IFileStore file = EFS.getLocalFileSystem().getStore(path);

if (file.fetchInfo().exists()) {
  MessageDialog overwriteDialog = new MessageDialog(...);
  if (overwriteDialog.open() != 0) {
    return;
  }
}

OutputStream stream = file.openOutputStream(EFS.NONE, null));
```

File dialog of SWT

Configuration of file types

User canceled

Access of local file system

Check file property

User canceled

Open stream for file

Workspace

- The “IDE-specific” part of Eclipse
  - Plug-in “org.eclipse.core.resources”
- Mapping of the file system
  - All files of a project directory are mapped
    - URI of resource, accessible via EFS
  - Additional meta data
    - Arbitrary properties
    - Marker (build errors, bookmarks, TODO-tags, ...)
  - Change notification
  - Incremental build system
    - Notification when resources change
    - Deltas allow incremental compilation
- Language specific additions
  - Example: Java Development Tools
  - Higher-level data model for packages, classes, methods, ...

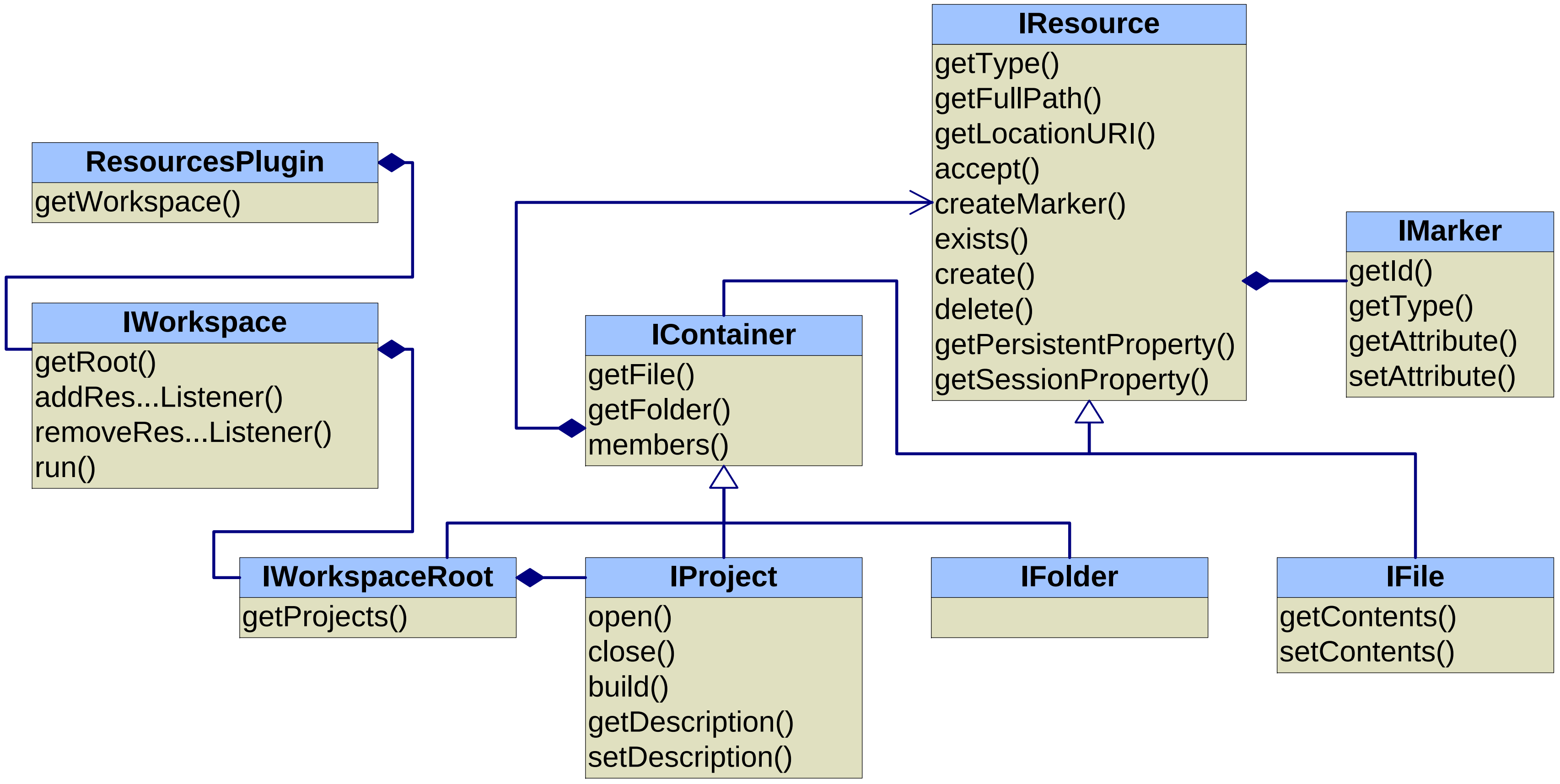
Resources

- All resources are handles
  - Small value objects that do not store the state
  - Can refer to non-existing resources
- File system
  - Mostly files and directories on disk
  - But not necessarily: transparent API in Eclipse
    - Example: Whole project in a zip-file
- Path vs. Location
  - Path: Logical position within the workspace
    - Represented by IPath
  - LocationURI: Virtual location in eclipse file system
  - Location: Physical location in disk file system
    - Not always available
- IMarker
  - Associate notes and meta-data to resources
  - Marker types are contributed by extensions
    - Defines a type system of marker types

Resources

- IResource
  - Base class for common functionality
  - Management: create, copy, move, delete
  - Property support: persistent and session properties
  - Marker support: createMarker, findMarkers
  - Accept visitor for traversal of children
- IFile: Something with a content
- IFolder: Container for other folders and files
- IProject
  - Top level folder
  - Support for builders and natures
  - Additional information stored in IProjectDescription
- IWorkspaceRoot
  - Virtual root for all projects
- IWorkspace
  - Not part of the resource hierarchy
  - Support for resource change listeners

Resources



ResourcesPlugin
getWorkspace()
IWorkspace
getRoot()
addRes...Listener()
removeRes...Listener()
run()
IWorkspaceRoot
getProjects()
IContainer
getFile()
getFolder()
members()
IResource
getType()
getFullPath()
getLocationURI()
accept()
createMarker()
exists()
create()
delete()
getPersistentProperty()
getSessionProperty()
IMarker
getId()
getType()
getAttribute()
setAttribute()
IProject
open()
close()
build()
getDescription()
setDescription()
IFolder
IFile
getContents()
setContents()

Resources

- Traversing the resource tree
  - Manually using the “members()” method
  - Use a IResourceVisitor
    - “visit()” method returns true if children should be visited
- Tracking resource changes
  - Add a IResourceChangeListener to workspace
  - Event with delta that describes all changed resources
    - Process with IResourceDeltaVisitor
  - Resource change events are expensive
    - Batch changes when possible
    - Use IWorkspaceRunnable or WorkspaceModifyOperation for atomic workspace operation
    - Only one notification after operation has finished.

Resources Example

```
IWorkspaceRoot root = ResourcesPlugin.getWorkspace().getRoot();


final IFile file = root.getFile(folderName.append(fileName));


getContainer().run(false, false, new WorkspaceModifyOperation() {

  protected void execute(IProgressMonitor monitor) {
    try {
      file.create(fileContent, false, monitor);

    } catch (CoreException ex) {
      UIUtilities.logError(ex.getMessage(), ex);
    }
  }
});
```

Get lightweight descriptor

Operation groups changes

Context that provides progress monitor

Workspace change

Error handling

Builders and Natures

- Incremental build system
  - One of the “highlights” of the Eclipse Java IDE
  - Compiles source files when they are saved
  - Class files always up-to-date
- Build system is flexible
  - No “default”-builder
  - New builders can be added
- Resource listener vs. Builder
  - Both respond to resource changes
  - Builders are added to projects, not to the workspace
  - Builders are explicitly ordered, so builders can depend on each other
  - Builders are permanently attached and stored in project configuration
  - Builders are usually long-running operations
- Natures
  - Natures are used to configure builders

Example Nature

```
<extension
    point="org.eclipse.core.resources.natures"
    id="DrawingNature">
  <runtime>
    <run class="at.ssw.drawit.internal.ide.builder.DrawingNature"/>
  </runtime>
  <builder id="at.ssw.drawit.ide.builder.DrawingBuilder"/>
</extension>
```

Extension point for natures

ID without plug-in prefix

Nature class

Builder as defined later

Property “project” is part of IProjectNature, but omitted here

```
public class DrawingNature implements IProjectNature {
  public void configure() throws CoreException {
    ...
    IProjectDescription description = getProject().getDescription();
    ICommand command = description.newCommand();
    command.setBuilderName(BuilderUtilities.BUILDER_ID);
    ICommand[] newCmds = Arrays.copyOf(cmds, cmds.length + 1);
    newCmds[cmds.length] = command;
    description.setBuildSpec(newCmds);
    getProject().setDescription(description, null);
  }

  public void deconfigure() throws CoreException {
    ...
  }
}
```

Check if builder present

Get project description

Add builder to build spec

Modify project description

Example Builder

```
<extension
    point="org.eclipse.core.resources.builders“
    id="DrawingBuilder">
  <builder hasNature="true" isConfigurable="false">
    <run class="at.ssw.drawit.internal.ide.builder.DrawingBuilder"/>
  </builder>
</extension>
```

Extension point for builders

ID without plug-in prefix

Builder class

```
public class DrawingBuilder extends IncrementalProjectBuilder {
  protected IProject[] build(int kind, ...) {
    if (kind == AUTO_BUILD || kind == INCREMENTAL_BUILD
      incrementalBuild(getDelta(getProject()));
    } else {
      fullBuild(getProject());
    }
    return null;
  }

  protected void clean(...) {
    ...
  }
}
```

Distinguish full and incremental builds

Delta specifies changes resources

List of projects the build depended on

Remove all results of previous builds

Example Marker

```
<extension
    point="org.eclipse.core.resources.markers"
    id="DrawingProblem"
    name="DrawIt Problem">
  <super type="org.eclipse.core.resources.problemmarker"/>
  <persistent value="true"/>
</extension>
```

Extension point for builders

ID without plug-in prefix

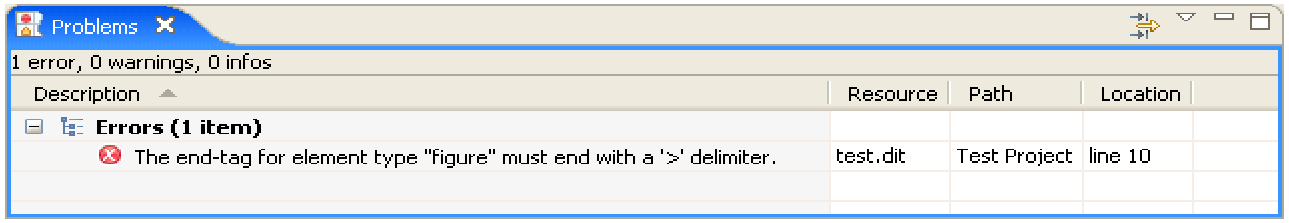
Human readable name

Marker type hierarchy

```
private void addMarker(IFile file, int line, String message) {
  IMarker marker = file.createMarker(BuilderUtilities.MARKER_ID);
  marker.setAttribute(IMarker.MESSAGE, message);
  marker.setAttribute(IMarker.SEVERITY, IMarker.SEVERITY_ERROR);
  if (line > 0) {
    marker.setAttribute(IMarker.LINE_NUMBER, line);
  }
}
```

Create new marker

Set attributes



1 error, 0 warnings, 0 infos

Description	Resource	Path	Location
Errors (1 item)			
The end-tag for element type "figure" must end with a '>' delimiter.	test.dit	Test Project	line 10

Other Features of Eclipse

- Help system
  - Content provided as html pages
  - Structure provided as xml file
  - Uses internal web server (Tomcat)
  - Servlets, Java Server Pages and other active content possible
- Internationalization
  - Localization of Java code
    - Resource bundles using class NLS
    - Or use standard Java resource bundles
  - Localization of plugin.xml
    - Label of actions, views, editors, ...
    - Localized text in file plugin.properties
    - Manifest and plugin.xml contains “%Key”
  - Compiler warning for non-localized strings

Internationalization

```
public class ContentOutlineMessages extends NLS {
  private static final String BUNDLE_NAME =
    "org.eclipse.ui.internal.views.contentoutline.messages";//$NON-NLS-1$


  public static String ContentOutline_noOutline;

  static {
    NLS.initializeMessages(BUNDLE_NAME, ContentOutlineMessages.class);
  }
}
```

Location of resources

Non-localized string

String message that is initialized with localized text

Load message values from bundle file

```
ContentOutline_noOutline = An outline is not available.
```

Content of resource bundle

Internationalization

plugin.xml

Localized label

Localized icon

```
<view
    name="%Views.ContentOutline"
    icon="$nl$/icons/full/eview16/outline_co.gif"
    category="org.eclipse.ui"
    class="org.eclipse.ui.views.contentoutline.ContentOutline"
    id="org.eclipse.ui.views.ContentOutline“/>
```

MANIFEST.MF

Localized text

```
Manifest-Version: 1.0
Bundle-Activator: org.eclipse.ui.internal.views.ViewsPlugin
Bundle-Name: %pluginName
Bundle-Vendor: %providerName
Bundle-ClassPath: .
Bundle-ManifestVersion: 2
Bundle-SymbolicName: org.eclipse.ui.views; singleton:=true
```

plugin.properties

```
pluginName= Views
providerName= Eclipse.org

Views.ContentOutline = Outline
```

Preferences

- Plug-in provides preference store
  - Create a subclass of AbstractUIPlugin
  - Method “getPreferenceStore()”
  - Preferences are key-value pairs
    - Keys are strings
    - Values are arbitrary objects, but stored as strings
    - Converter for many useful types
  - Define keys in separate interface
    - Keeps them separate
  - Preference initializer for default values
- User interface
  - Extend the standard preferences dialog
  - Uses concept of JFace dialog pages
  - Convenient support of field editors
    - Field editors for many useful types
    - Specify only the preference key and the label

Preferences

Define keys

```
public interface IPreferenceConstants {
  public static final String FIGURE_FILL_COLOR = "figureFillColor";
  public static final String FIGURE_LINE_COLOR = "figureLineColor";
  public static final String FIGURE_LINE_WIDTH = "figureLineWidth";
}
```

String constants for keys

Access preferences

```
public class UIUtilities implements IPreferenceConstants {
  public static Figure createFigure() {
    IPreferenceStore prefs = UIPlugin.getDefault().getPreferenceStore();


    Figure figure = new Figure();
    figure.setLineWidth(prefs.getInt(FIGURE_LINE_WIDTH));


    figure.setLineColor(PreferenceConverter.getColor(prefs, FIGURE_LINE_COLOR));
    figure.setFillColor(PreferenceConverter.getColor(prefs, FIGURE_FILL_COLOR));
    return figure;
  }
}
```

Get preference store

Get integer value

Get complex value

Preferences

Initialize preferences

```
public class PreferenceInitializer
    extends AbstractPreferenceInitializer implements IPreferenceConstants {

  public void initializeDefaultPreferences() {
    IPreferenceStore prefs = UIPlugin.getDefault().getPreferenceStore();

    prefs.setDefault(FIGURE_LINE_WIDTH, 2);
    PreferenceConverter.setDefault(prefs, FIGURE_LINE_COLOR, new RGB(255, 128, 0));
    PreferenceConverter.setDefault(prefs, FIGURE_FILL_COLOR, new RGB(255, 192, 128));
  }
}
```

Set integer value

Set complex value

Register initializer

```
<extension
    point="org.eclipse.core.runtime.preferences">
  <initializer
      class="at.ssw.drawit.ui.preferences.PreferenceInitializer"/>
</extension>
```

Preferences

Preference page

```
public class PreferencePage
    extends FieldEditorPreferencePage
    implements IWorkbenchPreferencePage, IPreferenceConstants {
  public PreferencePage() {
    super(GRID);
    setDescription("Default values used for new drawings and figures.");
    setPreferenceStore(UIPlugin.getDefault().getPreferenceStore());
  }

  protected void createFieldEditors() {
    addField(new IntegerFieldEditor(FIGURE_LINE_WIDTH, "Line width:",get...Parent()));
    addField(new ColorFieldEditor(FIGURE_LINE_COLOR, "Line color:",  get...Parent()));
    addField(new ColorFieldEditor(FIGURE_FILL_COLOR, "Fill color:",  get...Parent()));
  }
}
```

Page title

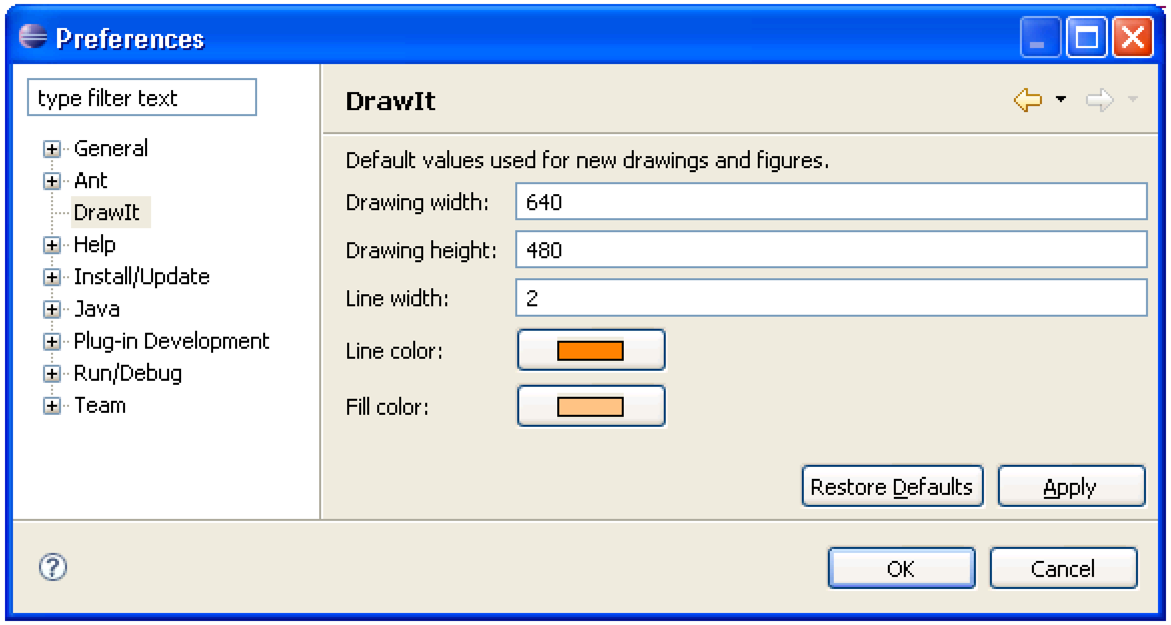
Get preference store

Add field editors

Register preference page

```
<extension
    point="org.eclipse.ui.preferencePages">
  <page
      id="at.ssw.drawit.ui.preferences.page"
      class="at.ssw.drawit.ui.preferences.PreferencePage"
      name="DrawIt"/>
</extension>
```

Preferences



Default values used for new drawings and figures.

Drawing width: 640

Drawing height: 480

Line width: 2

Line color: 🟧

Fill color: 🟧

Progress

- Long running operations
  - Report progress to user
  - Allow user to cancel operation
  - Do not block user interface
  - Run in separate thread
- Reporting progress
  - Operation reports to IProgressMonitor
  - Specify total work
  - Report when a work unit is completed
  - Check if user canceled the operation
- Jobs
  - High-level API, more flexible than threads
  - Scheduling rules
  - Locks with deadlock detection

Progress Example

```
IProgressService progressService = PlatformUI.getWorkbench().getProgressService();
try {
  progressService.busyCursorWhile(new IRunnableWithProgress() {
    public void run(IProgressMonitor monitor) throws InterruptedException {
      longRunningOperation(monitor);
    }
  });
} catch (InterruptedException ex) {
  System.out.println("user canceled");
} catch (InvocationTargetException ex) {
  ex.printStackTrace();
}
```

Method runs in another thread.
Progress dialog is shown after some time.
User can cancel.

User canceled

Unexpected exception

```
private void longRunningOperation(IProgressMonitor monitor) throws InterruptedException {
  monitor.beginTask("Long running example", 100);
  for (int i = 0; i < 100; i++) {
    sleep(100);

    monitor.worked(1);
    if (monitor.isCanceled()) {
      throw new InterruptedException();
    }
  }
  monitor.done();
}
```

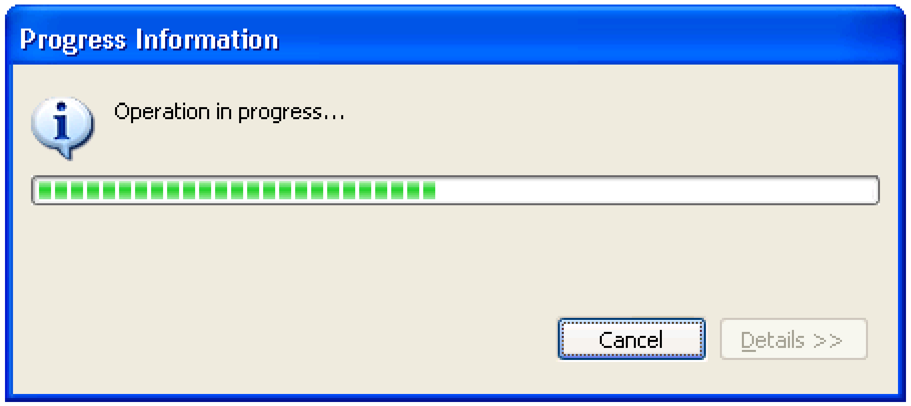
Total work

Work unit completed

Check if user canceled

Terminate execution

Progress Example

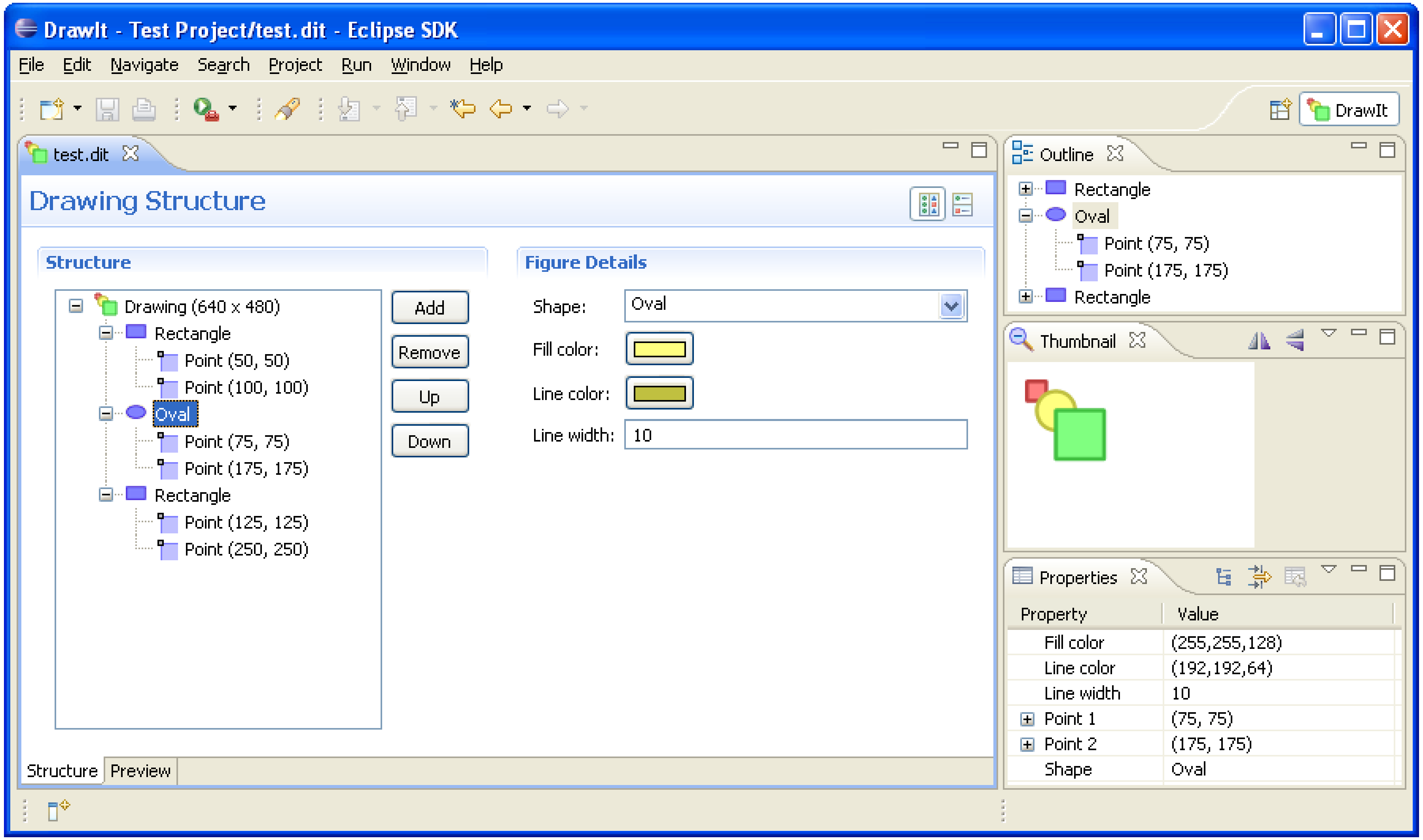


Operation in progress...

UI Forms

- Web-style user interfaces
  - Builds on SWT
  - Some new widgets, layouts and support classes
  - “Flat style” on all platforms
  - Not limited to Eclipse views or editors
- Toolkit
  - Rendering of standard widgets is changed
    - Custom borders on some platforms
    - But not on Windows XP
  - Use toolkit as a widget factory
- Managed forms
  - Adds lifecycle to a form
  - Simplifies master / details blocks
- Examples
  - Article: http://www.eclipse.org/articles/Article-Forms/article.html
  - Form-based editor for drawing files

Form Based Editor



DrawIt - Test Project/test.dit - Eclipse SDK

Drawing Structure

Structure

```
Drawing (640 x 480)
  Rectangle
    Point (50, 50)
    Point (100, 100)
  Oval
    Point (75, 75)
    Point (175, 175)
  Rectangle
    Point (125, 125)
    Point (250, 250)
```

Figure Details

Shape: Oval

Fill color:

Line color:

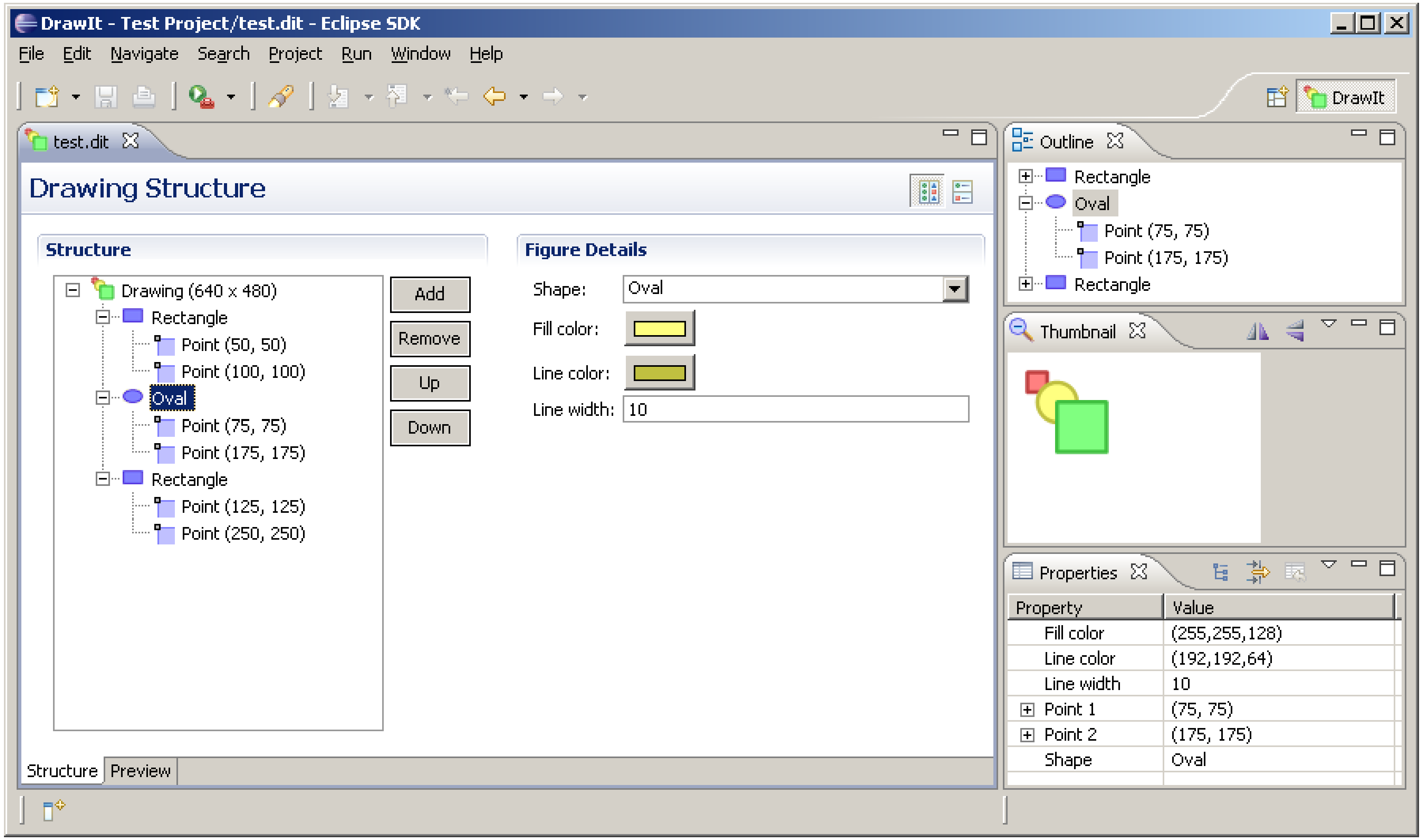
Line width: 10

Outline:

```
Rectangle
Oval
  Point (75, 75)
  Point (175, 175)
Rectangle
```

Property	Value
Fill color	(255,255,128)
Line color	(192,192,64)
Line width	10
Point 1	(75, 75)
Point 2	(175, 175)
Shape	Oval

Form Based Editor



DrawIt - Test Project/test.dit - Eclipse SDK

File Edit Navigate Search Project Run Window Help

test.dit

Drawing Structure

Structure

- Drawing (640 x 480)
  - Rectangle
    - Point (50, 50)
    - Point (100, 100)
  - Oval
    - Point (75, 75)
    - Point (175, 175)
  - Rectangle
    - Point (125, 125)
    - Point (250, 250)

Add | Remove | Up | Down

Figure Details

Shape: Oval

Fill color:

Line color:

Line width: 10

Structure | Preview

Outline

- Rectangle
- Oval
  - Point (75, 75)
  - Point (175, 175)
- Rectangle

Thumbnail

Properties

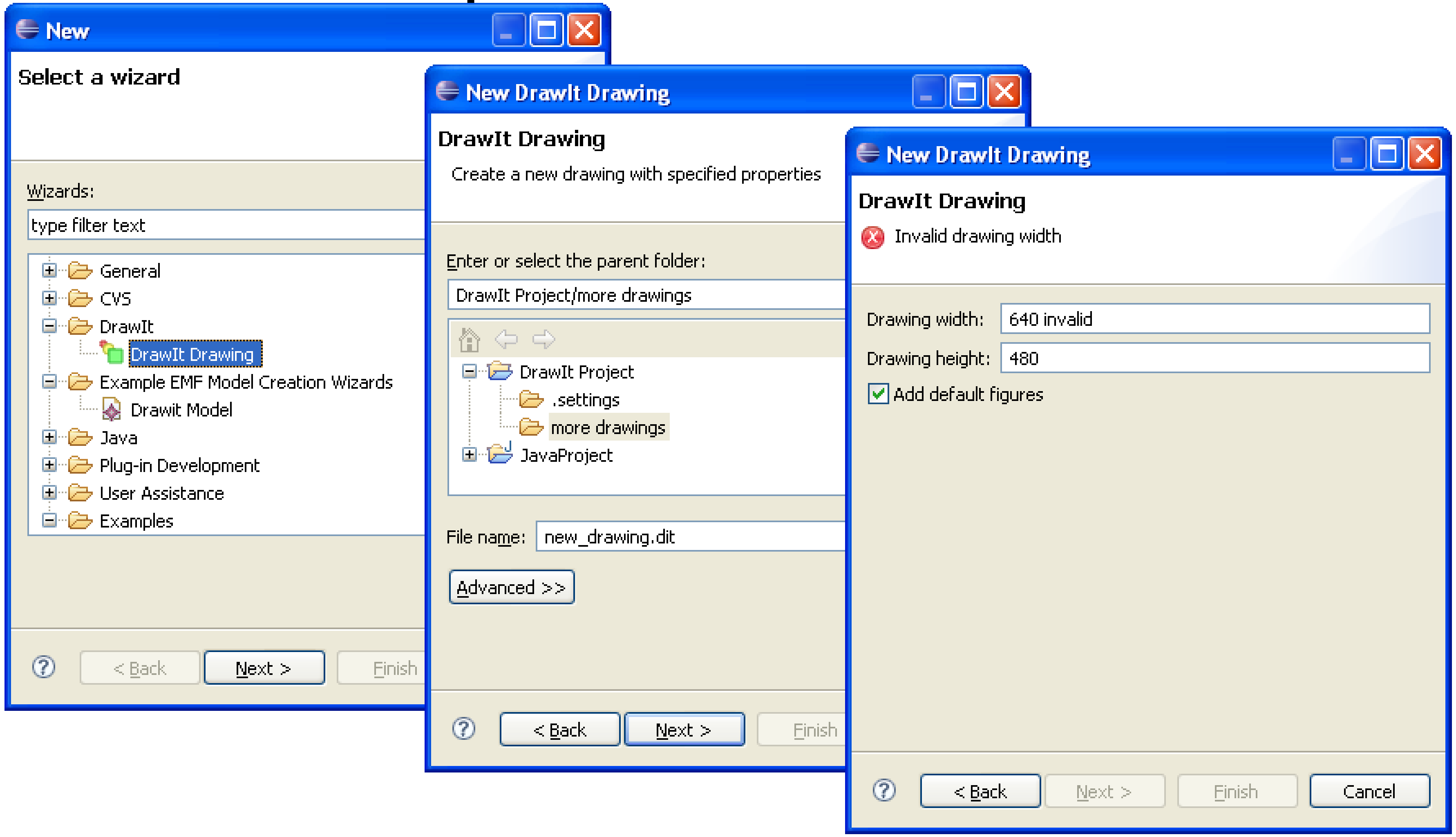
Property	Value
Fill color	(255,255,128)
Line color	(192,192,64)
Line width	10
Point 1	(75, 75)
Point 2	(175, 175)
Shape	Oval

Wizards

- JFace
  - Interface IWizard, abstract implementation class Wizard
  - Navigation between multiple pages (IWizardPage)
  - Support for input validation
    - Disable next- and finish-buttons
    - Show message for user

- Workbench
  - Extension point to contribute wizards
  - Integration with global selection
    - Pre-configure wizard based on currently selected element
  - INewWizard, IImportWizard, IExportWizard
  - Common pages can be re-used
    - Example: name for newly created file

Example: New File Wizard


New

Select a wizard

Wizards:

type filter text

- General
- CVS
- DrawIt
  - DrawIt Drawing
- Example EMF Model Creation Wizards
  - Drawit Model
- Java
- Plug-in Development
- User Assistance
- Examples

< Back | Next > | Finish

New DrawIt Drawing

DrawIt Drawing

Create a new drawing with specified properties

Enter or select the parent folder:

DrawIt Project/more drawings

- DrawIt Project
  - .settings
  - more drawings
- JavaProject

File name: new_drawing.dit

Advanced >>

< Back | Next > | Finish

New DrawIt Drawing

DrawIt Drawing

Invalid drawing width

Drawing width: 640 invalid

Drawing height: 480

✓ Add default figures

< Back | Next > | Finish | Cancel

Example: New File Wizard

```java
public class NewDrawingWizard extends Wizard implements INewWizard {

  public void init(IWorkbench newWorkbench, IStructuredSelection sel) {
    setWindowTitle("New DrawIt Drawing");

    fileNamePage = new WizardNewFileCreationPage("fileNamePage", sel);
    fileNamePage.setTitle("DrawIt Drawing");
    fileNamePage.setDescription(“...");
    fileNamePage.setFileName("new_drawing.dit");
    fileNamePage.setFileExtension("dit");
    addPage(fileNamePage);

    filePropertiesPage = new FilePropertiesPage("propertiesPage");
    filePropertiesPage.setTitle("DrawIt Drawing");
    filePropertiesPage.setDescription(“...");
    addPage(filePropertiesPage);
  }

  public boolean performFinish() {
  }
}
```

Wizard-global properties

Eclipse page for file name

Page properties

Add page to wizard

Custom page

Use the information of the pages to create new file

Example: New File Wizard

```
public class FilePropertiesPage extends WizardPage {
  public void createControl(Composite parent) {
    Composite container = new Composite(parent, SWT.NULL);
    widthText = new Text(container, SWT.BORDER | SWT.SINGLE);
    widthText.addModifyListener(modifyListener);
    setControl(container);
  }

  public int getWidth() {
    ... Integer.parseInt(widthText.getText());
  }

  private ModifyListener modifyListener = new ModifyListener() {
    public void modifyText(ModifyEvent e) {
      if (getWidth() <= 0) {
        setErrorMessage("Invalid drawing width");
        setPageComplete(false);
      } else {
        setErrorMessage(null);
        setPageComplete(true);
      }
    }
  };
}
```

Create SWT controls

Main control of the page

Conversion of user input

Respond to user input

Input invalid

Input valid

Example: New File Wizard

```
<extension
    point="org.eclipse.ui.newWizards">
  <category
      id="at.ssw.drawit.ide"
      name="DrawIt“/>
  <wizard
      id="at.ssw.drawit.ide.NewDrawingWizard"
      category="at.ssw.drawit.ide"
      class="at.ssw.drawit.internal.ide.wizard.NewDrawingWizard"
      name="DrawIt Drawing"
      icon="icons/drawing.gif"/>
</extension>
```

Name and ID of category

Implementation class

Name and icon for UI

```
<extension
    point="org.eclipse.ui.perspectiveExtensions">
  <perspectiveExtension
      targetID="at.ssw.drawit.drawingPerspective">
    <newWizardShortcut
        id="at.ssw.drawit.ide.NewDrawingWizard"/>
  </perspectiveExtension>
</extension>
```

Show in popup menu
in this perspective