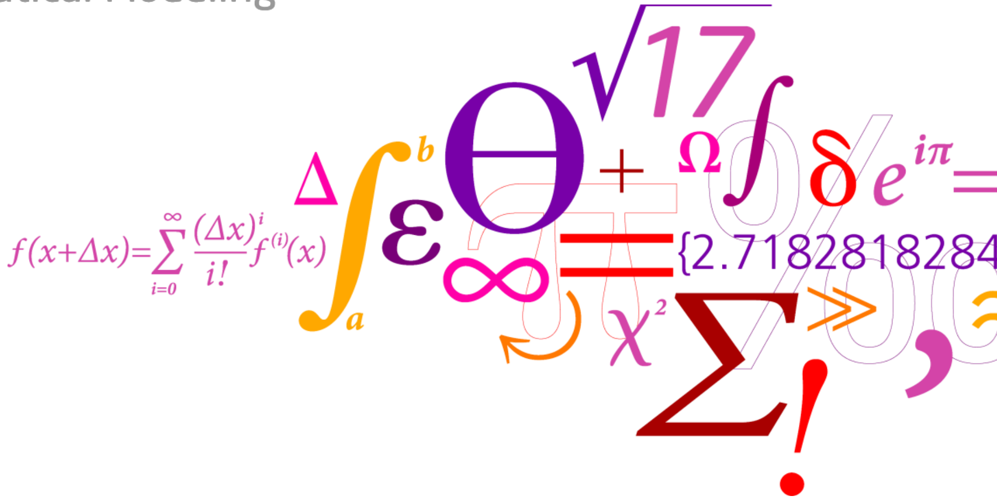
Software Engineering 2
A practical course in software engineering

Ekkart Kindler

DTU Informatics
Department of Informatics and Mathematical Modeling

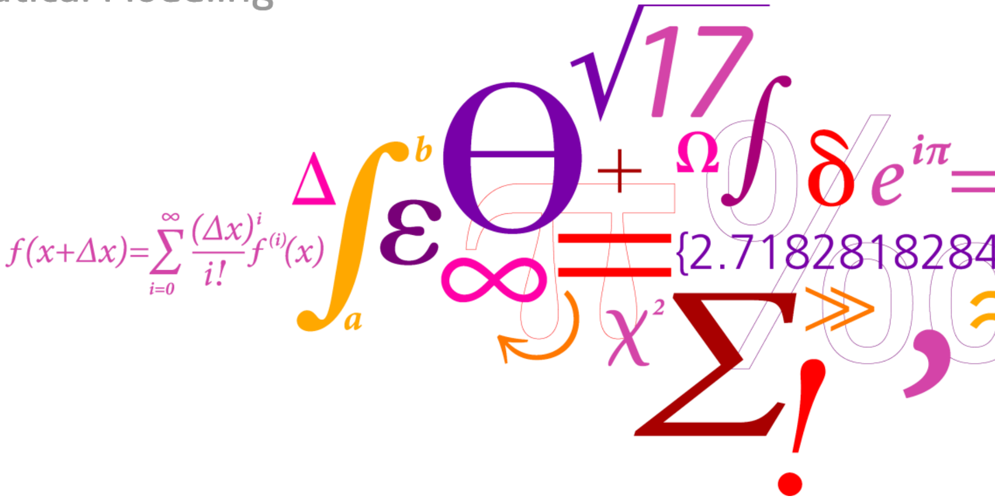
Tutorial



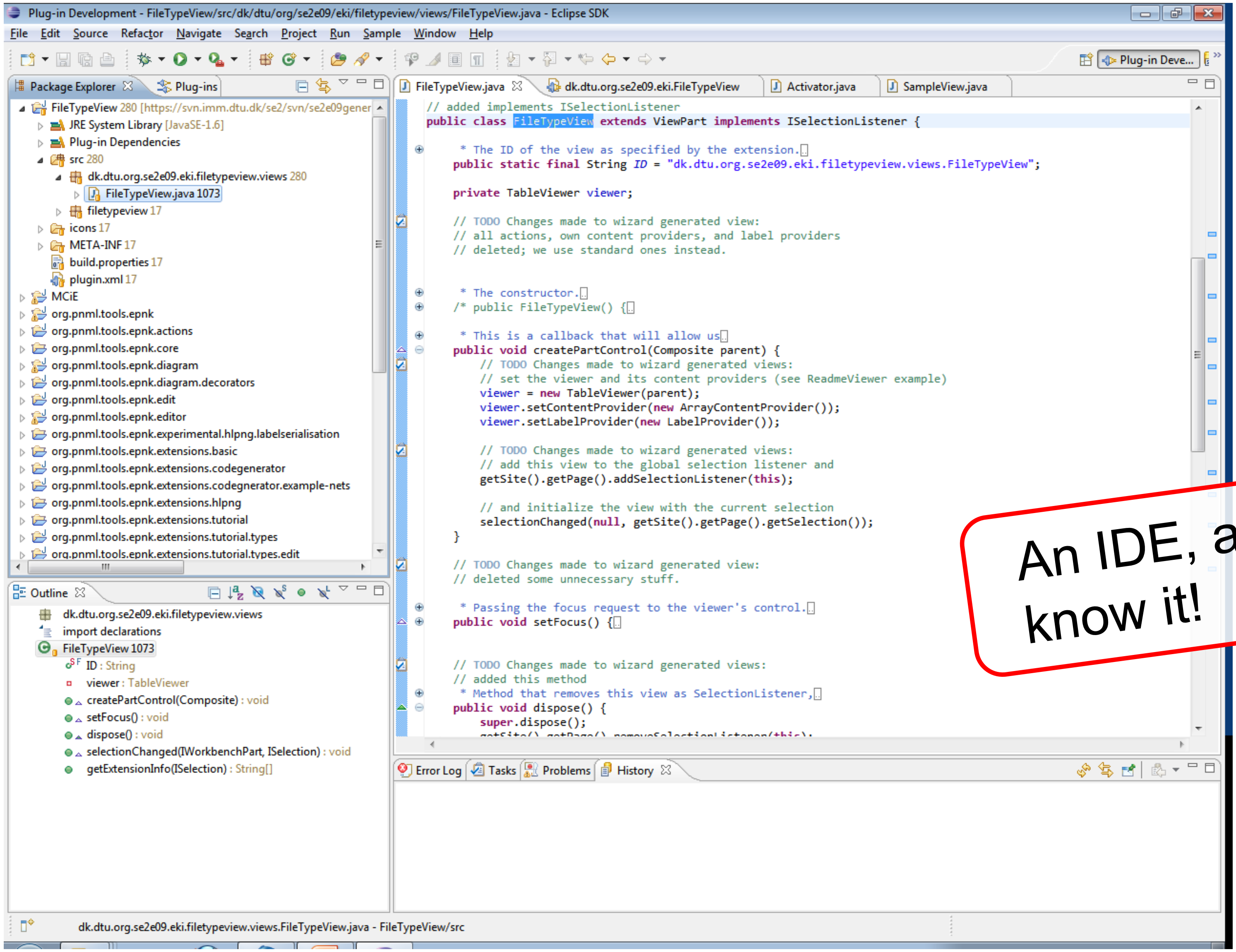
I. Introduction to Eclipse

Ekkart Kindler

DTU Informatics
Department of Informatics and Mathematical Modeling



1. Eclipse as an IDE



Plug-in Development - FileTypeView/src/dk/dtu/org/se2e09/eki/filetypeview/views/FileTypeView.java - Eclipse SDK

```
FileTypeView 280 [https://svn.imm.dtu.dk/se2/svn/se2e09gener
  JRE System Library [JavaSE-1.6]
  Plug-in Dependencies
  src 280
    dk.dtu.org.se2e09.eki.filetypeview.views 280
      FileTypeView.java 1073
    filetypeview 17
  icons 17
  META-INF 17
  build.properties 17
  plugin.xml 17
MCiE
org.pnml.tools.epnk
org.pnml.tools.epnk.actions
org.pnml.tools.epnk.core
org.pnml.tools.epnk.diagram
org.pnml.tools.epnk.diagram.decorators
org.pnml.tools.epnk.edit
org.pnml.tools.epnk.editor
org.pnml.tools.epnk.experimental.hlpng.labelserialisation
org.pnml.tools.epnk.extensions.basic
org.pnml.tools.epnk.extensions.codegenerator
org.pnml.tools.epnk.extensions.codegnerator.example-nets
org.pnml.tools.epnk.extensions.hlpng
org.pnml.tools.epnk.extensions.tutorial
org.pnml.tools.epnk.extensions.tutorial.types
org.pnml.tools.epnk.extensions.tutorial.types.edit
```

```java
// added implements ISelectionListener
public class FileTypeView extends ViewPart implements ISelectionListener {

     * The ID of the view as specified by the extension.
    public static final String ID = "dk.dtu.org.se2e09.eki.filetypeview.views.FileTypeView";

    private TableViewer viewer;

    // TODO Changes made to wizard generated view:
    // all actions, own content providers, and label providers
    // deleted; we use standard ones instead.


     * The constructor.
    /* public FileTypeView() {

     * This is a callback that will allow us
    public void createPartControl(Composite parent) {
        // TODO Changes made to wizard generated views:
        // set the viewer and its content providers (see ReadmeViewer example)
        viewer = new TableViewer(parent);
        viewer.setContentProvider(new ArrayContentProvider());
        viewer.setLabelProvider(new LabelProvider());

        // TODO Changes made to wizard generated views:
        // add this view to the global selection listener and
        getSite().getPage().addSelectionListener(this);

        // and initialize the view with the current selection
        selectionChanged(null, getSite().getPage().getSelection());
    }

    // TODO Changes made to wizard generated view:
    // deleted some unnecessary stuff.

     * Passing the focus request to the viewer's control.
    public void setFocus() {

    // TODO Changes made to wizard generated views:
    // added this method
     * Method that removes this view as SelectionListener,
    public void dispose() {
        super.dispose();
```

```
dk.dtu.org.se2e09.eki.filetypeview.views
import declarations
FileTypeView 1073
  ID : String
  viewer : TableViewer
  createPartControl(Composite) : void
  setFocus() : void
  dispose() : void
  selectionChanged(IWorkbenchPart, ISelection) : void
  getExtensionInfo(ISelection) : String[]
```

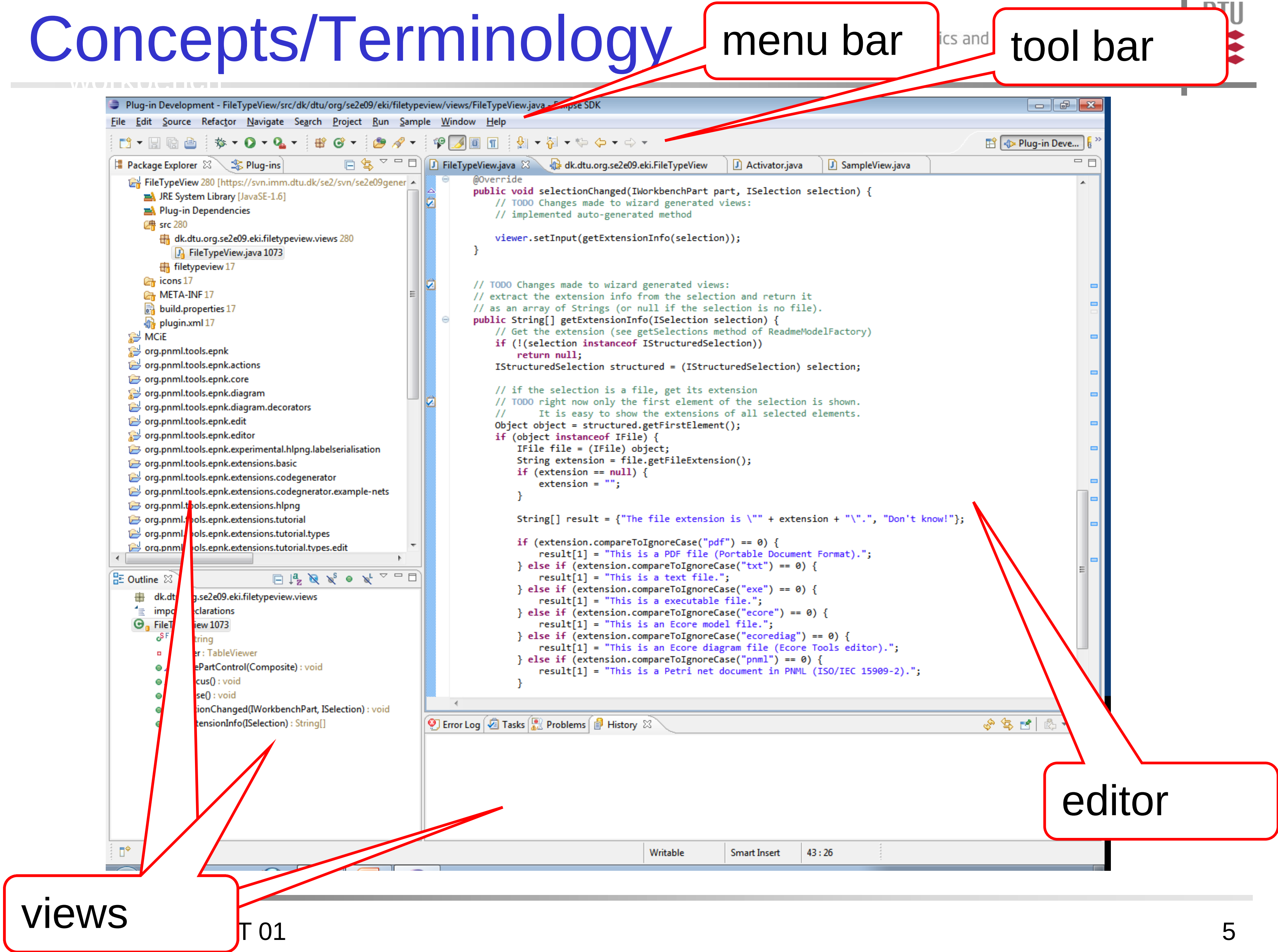
dk.dtu.org.se2e09.eki.filetypeview.views.FileTypeView.java - FileTypeView/src

An IDE, as we know it!

IDE

- Project and resource browser
- Nice and powerful (structural) editors (different programming languages)
- Error highlighting (and correction support)
- Build process behind the scenes
- Powerful debugging
- ...

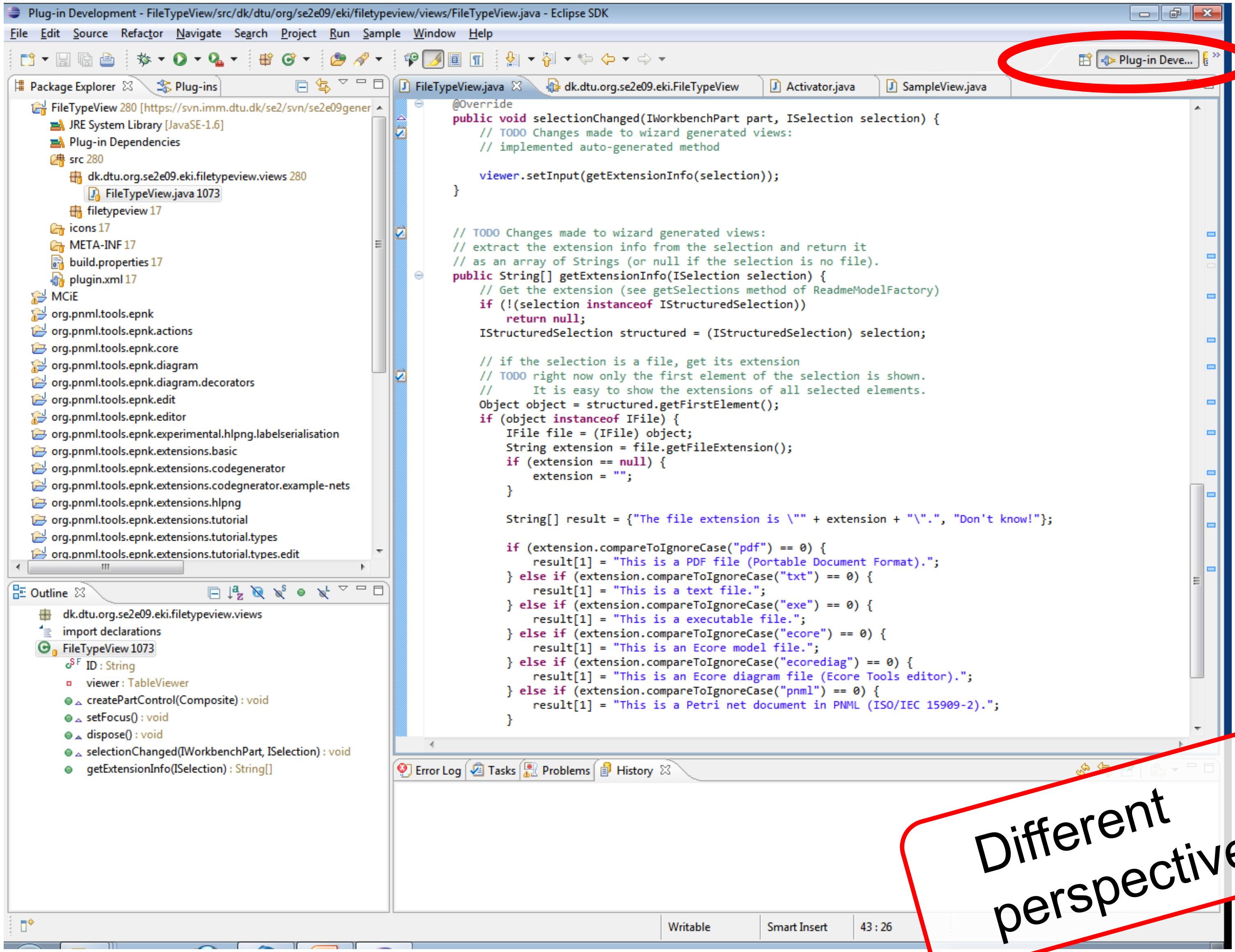
Concepts/Terminology


menu bar
tool bar
editor
views
Plug-in Development - FileTypeView/src/dk/dtu/org/se2e09/eki/filetypeview/views/FileTypeView.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Sample Window Help
Plug-in Deve...
Package Explorer
Plug-ins
FileTypeView 280 [https://svn.imm.dtu.dk/se2/svn/se2e09gener
JRE System Library [JavaSE-1.6]
Plug-in Dependencies
src 280
dk.dtu.org.se2e09.eki.filetypeview.views 280
FileTypeView.java 1073
filetypeview 17
icons 17
META-INF 17
build.properties 17
plugin.xml 17
MCiE
org.pnml.tools.epnk
org.pnml.tools.epnk.actions
org.pnml.tools.epnk.core
org.pnml.tools.epnk.diagram
org.pnml.tools.epnk.diagram.decorators
org.pnml.tools.epnk.edit
org.pnml.tools.epnk.editor
org.pnml.tools.epnk.experimental.hlpng.labelserialisation
org.pnml.tools.epnk.extensions.basic
org.pnml.tools.epnk.extensions.codegenerator
org.pnml.tools.epnk.extensions.codegnerator.example-nets
org.pnml.tools.epnk.extensions.hlpng
org.pnml.tools.epnk.extensions.tutorial
org.pnml.tools.epnk.extensions.tutorial.types
org.pnml.tools.epnk.extensions.tutorial.types.edit
Outline
FileTypeView.java
dk.dtu.org.se2e09.eki.FileTypeView
Activator.java
SampleView.java
Error Log
Tasks
Problems
History
Writable
Smart Insert
43 : 26

```
@Override
public void selectionChanged(IWorkbenchPart part, ISelection selection) {
    // TODO Changes made to wizard generated views:
    // implemented auto-generated method

    viewer.setInput(getExtensionInfo(selection));
}


// TODO Changes made to wizard generated views:
// extract the extension info from the selection and return it
// as an array of Strings (or null if the selection is no file).
public String[] getExtensionInfo(ISelection selection) {
    // Get the extension (see getSelections method of ReadmeModelFactory)
    if (!(selection instanceof IStructuredSelection))
        return null;
    IStructuredSelection structured = (IStructuredSelection) selection;

    // if the selection is a file, get its extension
    // TODO right now only the first element of the selection is shown.
    //      It is easy to show the extensions of all selected elements.
    Object object = structured.getFirstElement();
    if (object instanceof IFile) {
        IFile file = (IFile) object;
        String extension = file.getFileExtension();
        if (extension == null) {
            extension = "";
        }

        String[] result = {"The file extension is \"" + extension + "\".", "Don't know!"};

        if (extension.compareToIgnoreCase("pdf") == 0) {
            result[1] = "This is a PDF file (Portable Document Format).";
        } else if (extension.compareToIgnoreCase("txt") == 0) {
            result[1] = "This is a text file.";
        } else if (extension.compareToIgnoreCase("exe") == 0) {
            result[1] = "This is a executable file.";
        } else if (extension.compareToIgnoreCase("ecore") == 0) {
            result[1] = "This is an Ecore model file.";
        } else if (extension.compareToIgnoreCase("ecorediag") == 0) {
            result[1] = "This is an Ecore diagram file (Ecore Tools editor).";
        } else if (extension.compareToIgnoreCase("pnml") == 0) {
            result[1] = "This is a Petri net document in PNML (ISO/IEC 15909-2).";
        }
```

Concepts/Terminology

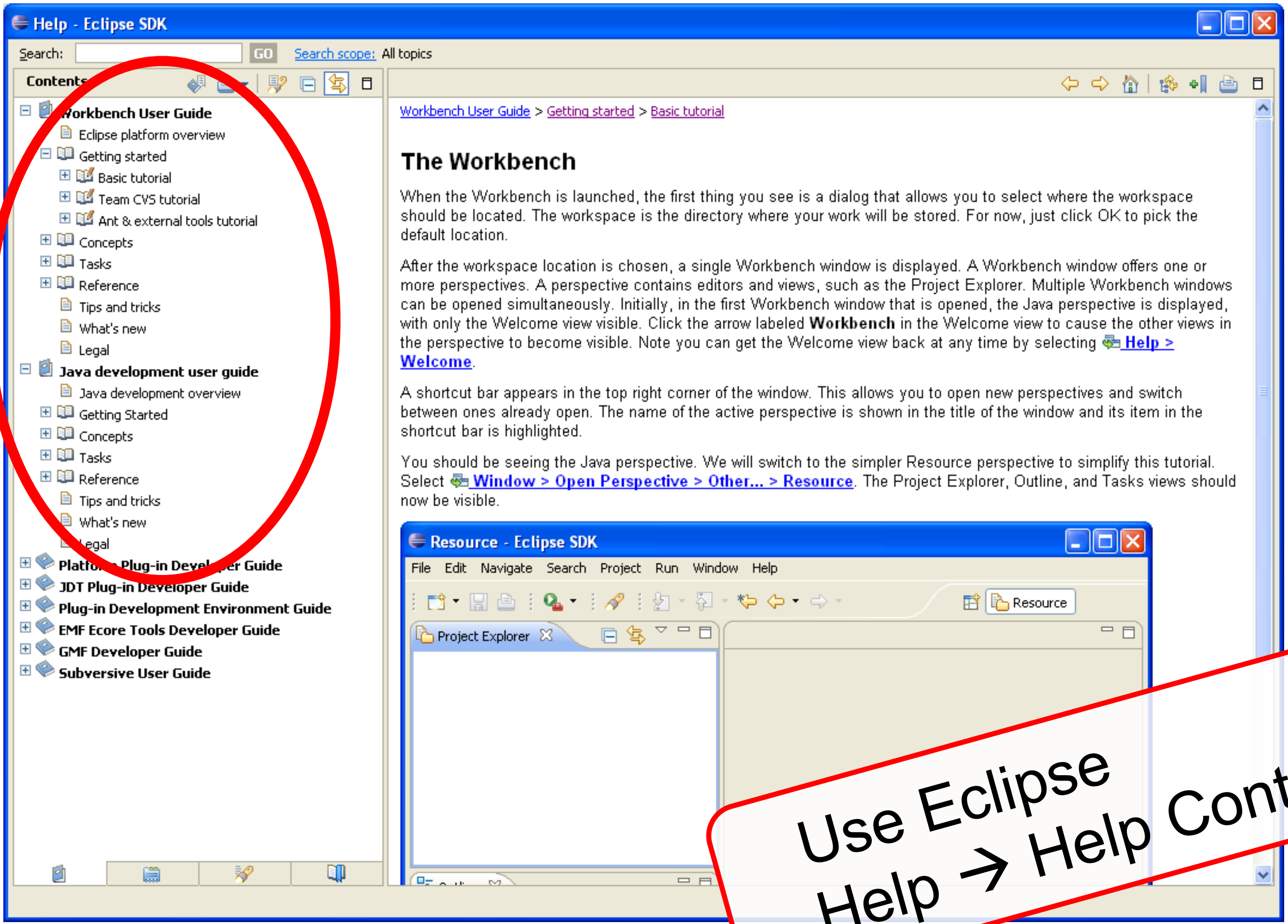
DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler


Plug-in Development - FileTypeView/src/dk/dtu/org/se2e09/eki/filetypeview/views/FileTypeView.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Sample Window Help
Plug-in Deve...
Package Explorer
Plug-ins
FileTypeView 280 [https://svn.imm.dtu.dk/se2/svn/se2e09gener
JRE System Library [JavaSE-1.6]
Plug-in Dependencies
src 280
dk.dtu.org.se2e09.eki.filetypeview.views 280
FileTypeView.java 1073
filetypeview 17
icons 17
META-INF 17
build.properties 17
plugin.xml 17
MCiE
org.pnml.tools.epnk
org.pnml.tools.epnk.actions
org.pnml.tools.epnk.core
org.pnml.tools.epnk.diagram
org.pnml.tools.epnk.diagram.decorators
org.pnml.tools.epnk.edit
org.pnml.tools.epnk.editor
org.pnml.tools.epnk.experimental.hlpng.labelserialisation
org.pnml.tools.epnk.extensions.basic
org.pnml.tools.epnk.extensions.codegenerator
org.pnml.tools.epnk.extensions.codegnerator.example-nets
org.pnml.tools.epnk.extensions.hlpng
org.pnml.tools.epnk.extensions.tutorial
org.pnml.tools.epnk.extensions.tutorial.types
org.pnml.tools.epnk.extensions.tutorial.types.edit
Outline
dk.dtu.org.se2e09.eki.filetypeview.views
import declarations
FileTypeView 1073
ID : String
viewer : TableViewer
createPartControl(Composite) : void
setFocus() : void
dispose() : void
selectionChanged(IWorkbenchPart, ISelection) : void
getExtensionInfo(ISelection) : String[]
FileTypeView.java
dk.dtu.org.se2e09.eki.FileTypeView
Activator.java
SampleView.java
Error Log
Tasks
Problems
History
Writable
Smart Insert
43 : 26

```
    @Override
    public void selectionChanged(IWorkbenchPart part, ISelection selection) {
        // TODO Changes made to wizard generated views:
        // implemented auto-generated method

        viewer.setInput(getExtensionInfo(selection));
    }


    // TODO Changes made to wizard generated views:
    // extract the extension info from the selection and return it
    // as an array of Strings (or null if the selection is no file).
    public String[] getExtensionInfo(ISelection selection) {
        // Get the extension (see getSelections method of ReadmeModelFactory)
        if (!(selection instanceof IStructuredSelection))
            return null;
        IStructuredSelection structured = (IStructuredSelection) selection;

        // if the selection is a file, get its extension
        // TODO right now only the first element of the selection is shown.
        //      It is easy to show the extensions of all selected elements.
        Object object = structured.getFirstElement();
        if (object instanceof IFile) {
            IFile file = (IFile) object;
            String extension = file.getFileExtension();
            if (extension == null) {
                extension = "";
            }

            String[] result = {"The file extension is \"" + extension + "\".", "Don't know!"};

            if (extension.compareToIgnoreCase("pdf") == 0) {
                result[1] = "This is a PDF file (Portable Document Format).";
            } else if (extension.compareToIgnoreCase("txt") == 0) {
                result[1] = "This is a text file.";
            } else if (extension.compareToIgnoreCase("exe") == 0) {
                result[1] = "This is a executable file.";
            } else if (extension.compareToIgnoreCase("ecore") == 0) {
                result[1] = "This is an Ecore model file.";
            } else if (extension.compareToIgnoreCase("ecorediag") == 0) {
                result[1] = "This is an Ecore diagram file (Ecore Tools editor).";
            } else if (extension.compareToIgnoreCase("pnml") == 0) {
                result[1] = "This is a Petri net document in PNML (ISO/IEC 15909-2).";
            }
```

Different perspectives

More info on IDE


Help - Eclipse SDK
Search:
GO
Search scope: All topics
Contents
Workbench User Guide
Eclipse platform overview
Getting started
Basic tutorial
Team CVS tutorial
Ant & external tools tutorial
Concepts
Tasks
Reference
Tips and tricks
What's new
Legal
Java development user guide
Java development overview
Getting Started
Concepts
Tasks
Reference
Tips and tricks
What's new
Legal
Platform Plug-in Developer Guide
JDT Plug-in Developer Guide
Plug-in Development Environment Guide
EMF Ecore Tools Developer Guide
GMF Developer Guide
Subversive User Guide
Workbench User Guide > Getting started > Basic tutorial
Resource - Eclipse SDK
File Edit Navigate Search Project Run Window Help
Resource
Project Explorer

The Workbench

When the Workbench is launched, the first thing you see is a dialog that allows you to select where the workspace should be located. The workspace is the directory where your work will be stored. For now, just click OK to pick the default location.

After the workspace location is chosen, a single Workbench window is displayed. A Workbench window offers one or more perspectives. A perspective contains editors and views, such as the Project Explorer. Multiple Workbench windows can be opened simultaneously. Initially, in the first Workbench window that is opened, the Java perspective is displayed, with only the Welcome view visible. Click the arrow labeled **Workbench** in the Welcome view to cause the other views in the perspective to become visible. Note you can get the Welcome view back at any time by selecting **Help > Welcome**.

A shortcut bar appears in the top right corner of the window. This allows you to open new perspectives and switch between ones already open. The name of the active perspective is shown in the title of the window and its item in the shortcut bar is highlighted.

You should be seeing the Java perspective. We will switch to the simpler Resource perspective to simplify this tutorial. Select **Window > Open Perspective > Other... > Resource**. The Project Explorer, Outline, and Tasks views should now be visible.

Use Eclipse
Help → Help Contents

Installation

http://www2.imm.dtu.dk/courses/02162/e13/project/eclipse-installation.html

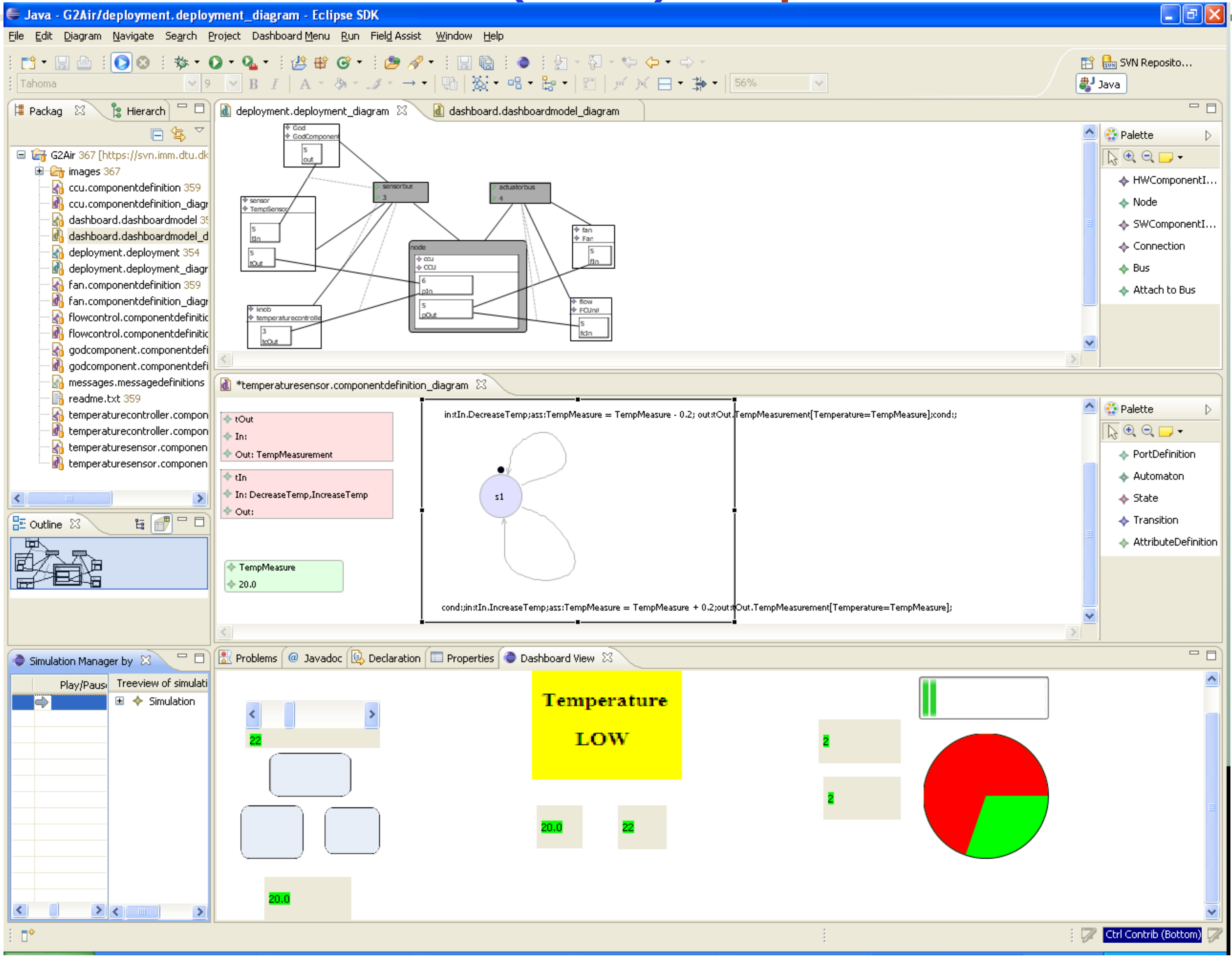
2. Eclipse as a Platform

- Eclipse is not only an IDE you can develop software (programs?) with
- You can also develop software for Eclipse and this way extend Eclipse by your own functionality (e.g. CASE Tool of the last years, the ePNK, or your “PNVis” tool of this year)
- If you do not like the full Eclipse, you can also restrict it for your application: Rich Client Platform (RCP)

Not in this course!

CASE Tool SE2 (e09)

DTU Informatics
Department of Informatics and Mathematical Modelling
Ekkart Kindler


Java - G2Air/deployment.deployment_diagram - Eclipse SDK
File Edit Diagram Navigate Search Project Dashboard Menu Run Field Assist Window Help
SVN Reposito...
Java
Packag
Hierarch
G2Air 367 [https://svn.imm.dtu.dk
images 367
ccu.componentdefinition 359
ccu.componentdefinition_diagr
dashboard.dashboardmodel 35
dashboard.dashboardmodel_d
deployment.deployment 354
deployment.deployment_diagr
fan.componentdefinition 359
fan.componentdefinition_diagr
flowcontrol.componentdefinitic
flowcontrol.componentdefinitic
godcomponent.componentdefi
godcomponent.componentdefi
messages.messagedefinitions
readme.txt 359
temperaturecontroller.compon
temperaturecontroller.compon
temperaturesensor.componen
temperaturesensor.componen
deployment.deployment_diagram
dashboard.dashboardmodel_diagram
Palette
HWComponentI...
Node
SWComponentI...
Connection
Bus
Attach to Bus
*temperaturesensor.componentdefinition_diagram
tOut
In:
Out: TempMeasurement
tIn
In: DecreaseTemp,IncreaseTemp
Out:
TempMeasure
20.0
s1
PortDefinition
Automaton
State
Transition
AttributeDefinition
Outline
Simulation Manager by
Play/Paus
Treeview of simulati
Simulation
Problems
Javadoc
Declaration
Properties
Dashboard View
Temperature
LOW
22
20.0
2
Ctrl Contrib (Bottom)

in:tIn.DecreaseTemp;ass:TempMeasure = TempMeasure - 0.2; out:tOut.TempMeasurement[Temperature=TempMeasure];cond:;

cond:;in:tIn.IncreaseTemp;ass:TempMeasure = TempMeasure + 0.2;out:tOut.TempMeasurement[Temperature=TempMeasure];

Concept/Terminology

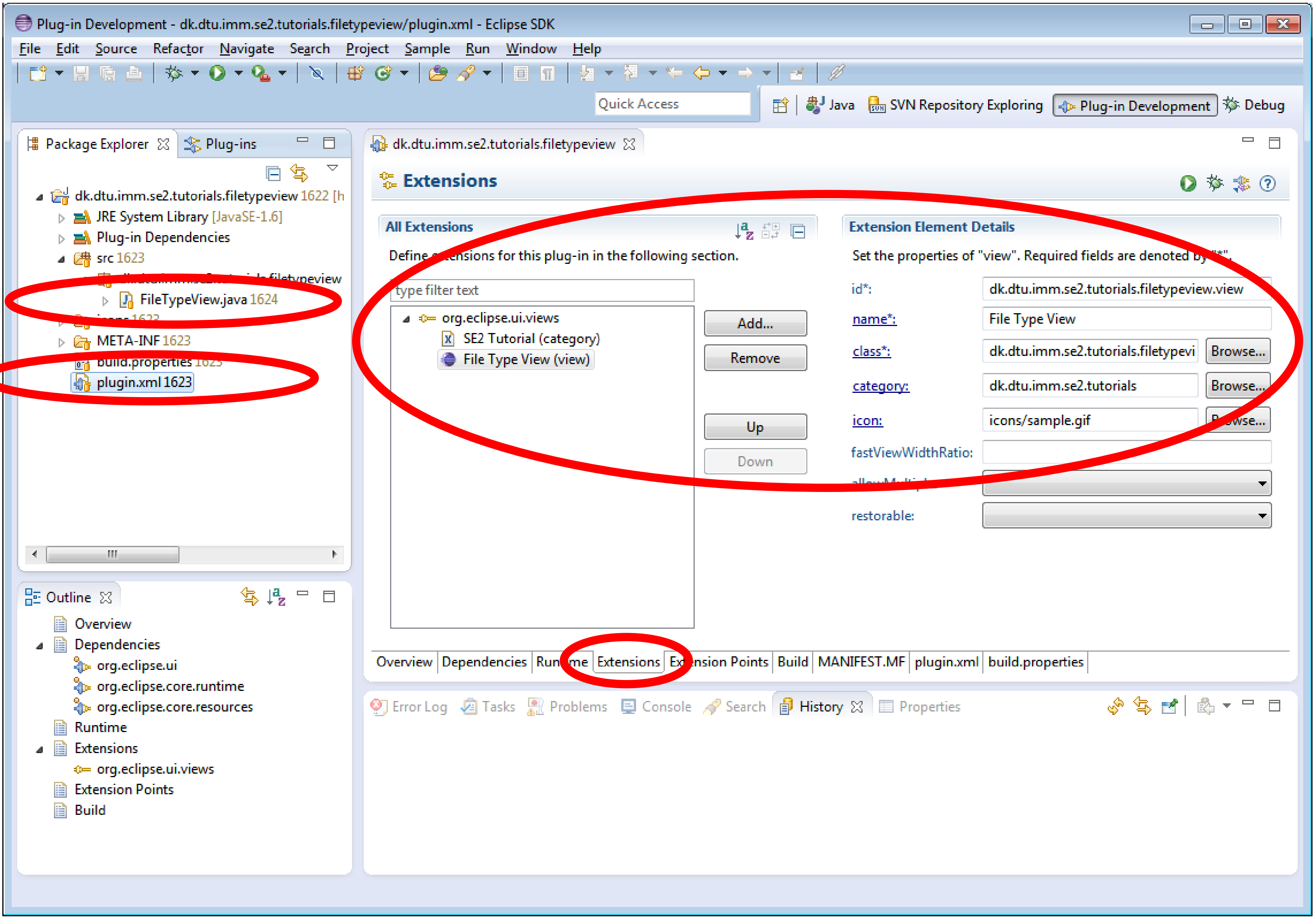
- The basic concept for extending Eclipse is **plugins**
- Almost everything in Eclipse is a plugIn

- **Extension point**: defines a possibility for others to extend the functionality

- **Extension**: defines the actual extension with the information required by the resp. extension point

Extensions are often called **plugins**

Example: File Type View



Plug-in Development - dk.dtu.imm.se2.tutorials.filetypeview/plugin.xml - Eclipse SDK

Package Explorer:
- dk.dtu.imm.se2.tutorials.filetypeview 1622
  - JRE System Library [JavaSE-1.6]
  - Plug-in Dependencies
  - src 1623
    - dk.dtu.imm.se2.tutorials.filetypeview
      - FileTypeView.java 1624
  - icons 1623
  - META-INF 1623
  - build.properties 1623
  - plugin.xml 1623

Extensions

All Extensions

Define extensions for this plug-in in the following section.

- org.eclipse.ui.views
  - SE2 Tutorial (category)
  - File Type View (view)

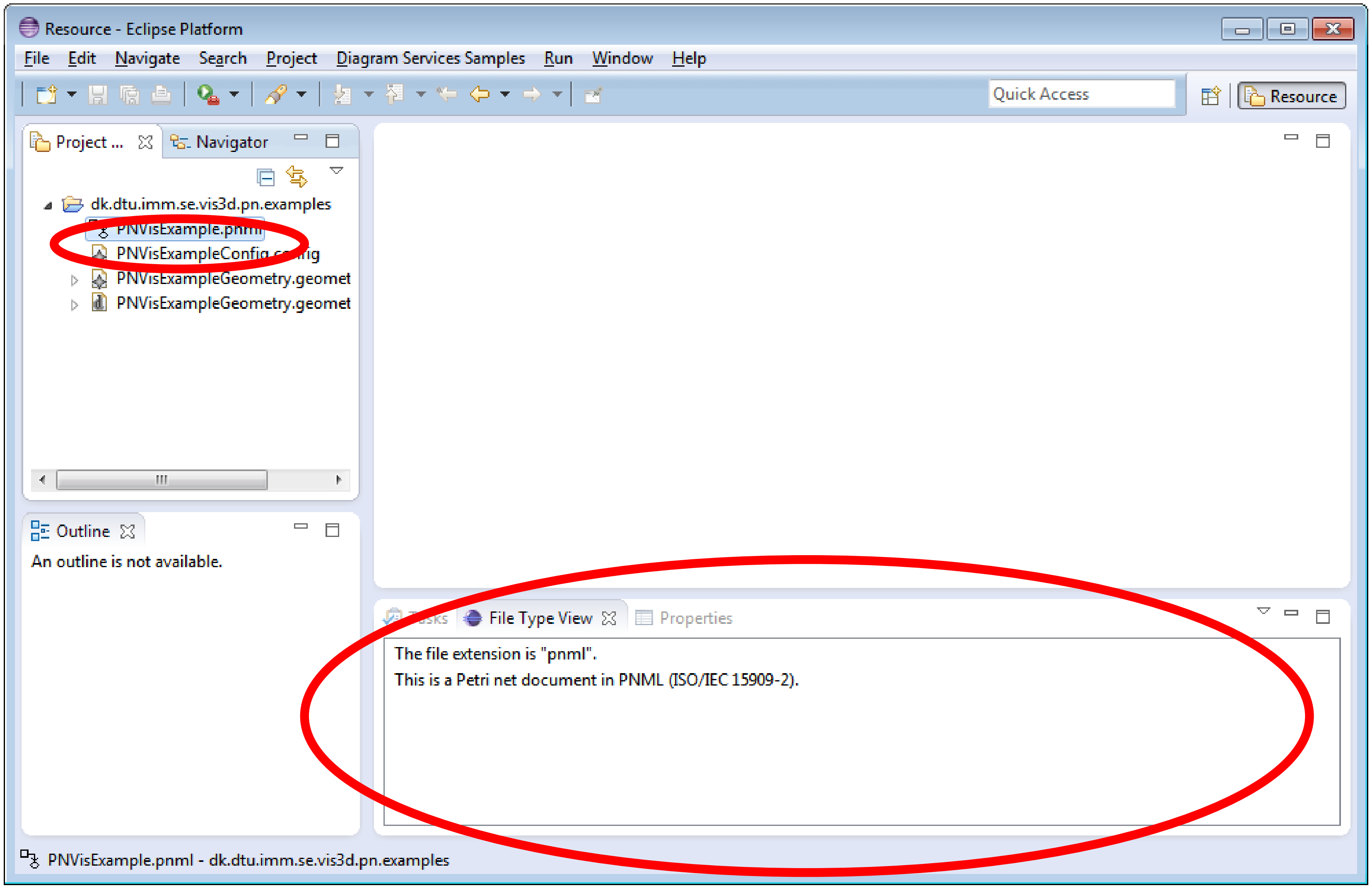
Extension Element Details

Set the properties of "view". Required fields are denoted by "*".

id*:	dk.dtu.imm.se2.tutorials.filetypeview.view
name*:	File Type View
class*:	dk.dtu.imm.se2.tutorials.filetypevi
category:	dk.dtu.imm.se2.tutorials
icon:	icons/sample.gif
fastViewWidthRatio:	
allowMultiple:	
restorable:	

Add...
Remove
Up
Down
Browse...
Overview | Dependencies | Runtime | Extensions | Extension Points | Build | MANIFEST.MF | plugin.xml | build.properties
Outline
Overview
Dependencies
org.eclipse.ui
org.eclipse.core.runtime
org.eclipse.core.resources
Runtime
Extensions
org.eclipse.ui.views
Extension Points
Build
Error Log | Tasks | Problems | Console | Search | History | Properties

File Type View (running)


Resource - Eclipse Platform
File Edit Navigate Search Project Diagram Services Samples Run Window Help
Quick Access
Resource
Project ...
Navigator
dk.dtu.imm.se.vis3d.pn.examples
PNVisExample.pnml
PNVisExampleConfig.config
PNVisExampleGeometry.geomet
PNVisExampleGeometry.geomet
Outline
An outline is not available.
Tasks
File Type View
Properties
PNVisExample.pnml - dk.dtu.imm.se.vis3d.pn.examples

The file extension is "pnml".
This is a Petri net document in PNML (ISO/IEC 15909-2).

Own applications

- Typically, an application defines many new extensions but only a few (if any) new extension points

  E.g. in the CASE Tool of an SE2 projects some years back, there was just one new extension point defined

- The ePNK has several extension points: the most important one for your: an extension point for Petri Net Types (details will be discussed in a later tutorial)

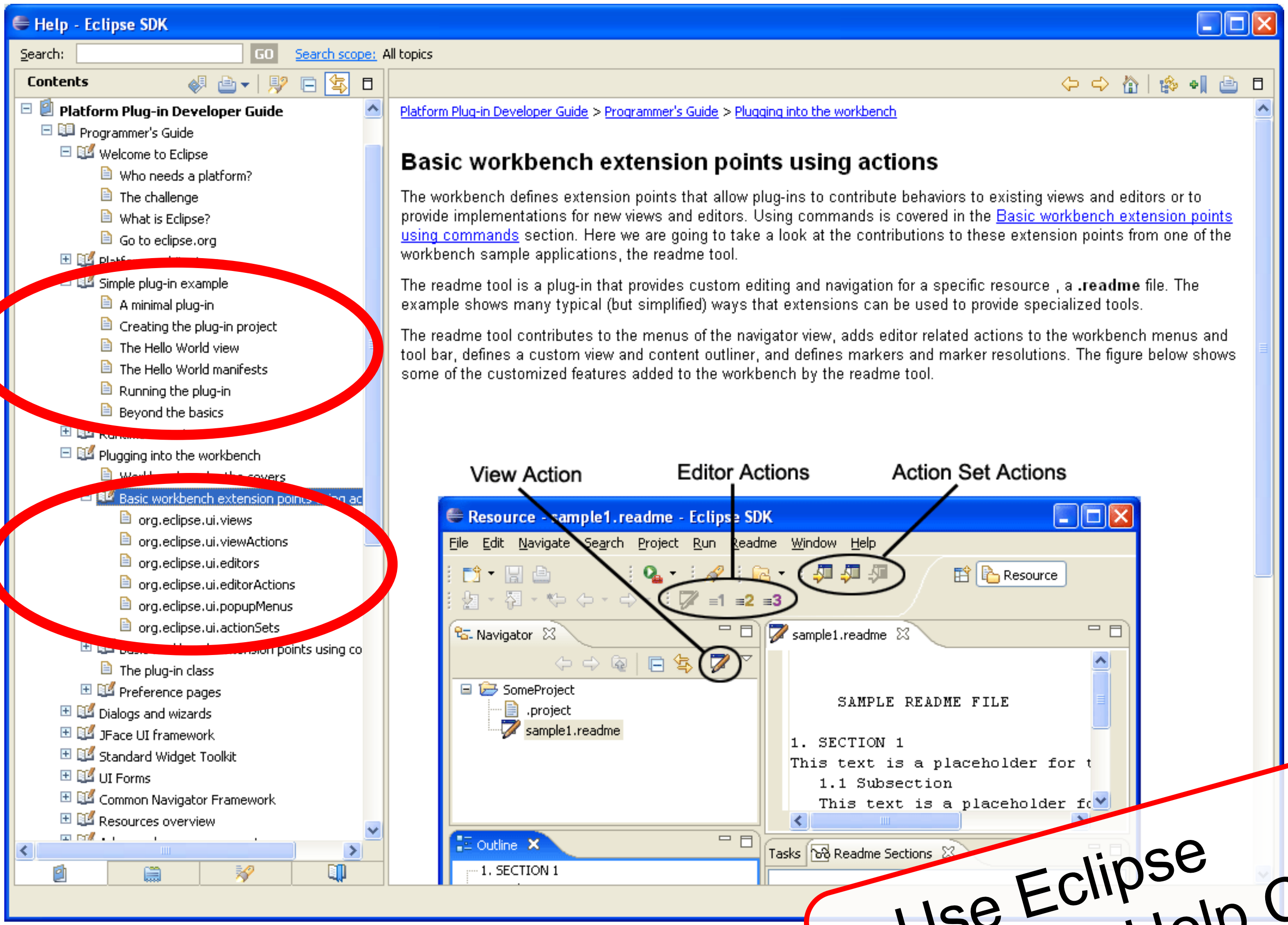
Platform PlugIn Development

- Typically an application extends "standard" extension points of the platform
  - views
  - editors
  - actions / commands
  - menus
  - …

For programming such plugins in Eclipse (see Help):

- SWT (Standard Widget Toolkit)
- JFace (UI toolkit based on SWT)

Info on IDE extensions



Help - Eclipse SDK

Search: GO Search scope: All topics

Contents

- Platform Plug-in Developer Guide
  - Programmer's Guide
    - Welcome to Eclipse
      - Who needs a platform?
      - The challenge
      - What is Eclipse?
      - Go to eclipse.org
    - Simple plug-in example
      - A minimal plug-in
      - Creating the plug-in project
      - The Hello World view
      - The Hello World manifests
      - Running the plug-in
      - Beyond the basics
    - Plugging into the workbench
      - Basic workbench extension points using ac
        - org.eclipse.ui.views
        - org.eclipse.ui.viewActions
        - org.eclipse.ui.editors
        - org.eclipse.ui.editorActions
        - org.eclipse.ui.popupMenus
        - org.eclipse.ui.actionSets
      - The plug-in class
      - Preference pages
    - Dialogs and wizards
    - JFace UI framework
    - Standard Widget Toolkit
    - UI Forms
    - Common Navigator Framework
    - Resources overview

Platform Plug-in Developer Guide > Programmer's Guide > Plugging into the workbench

Basic workbench extension points using actions

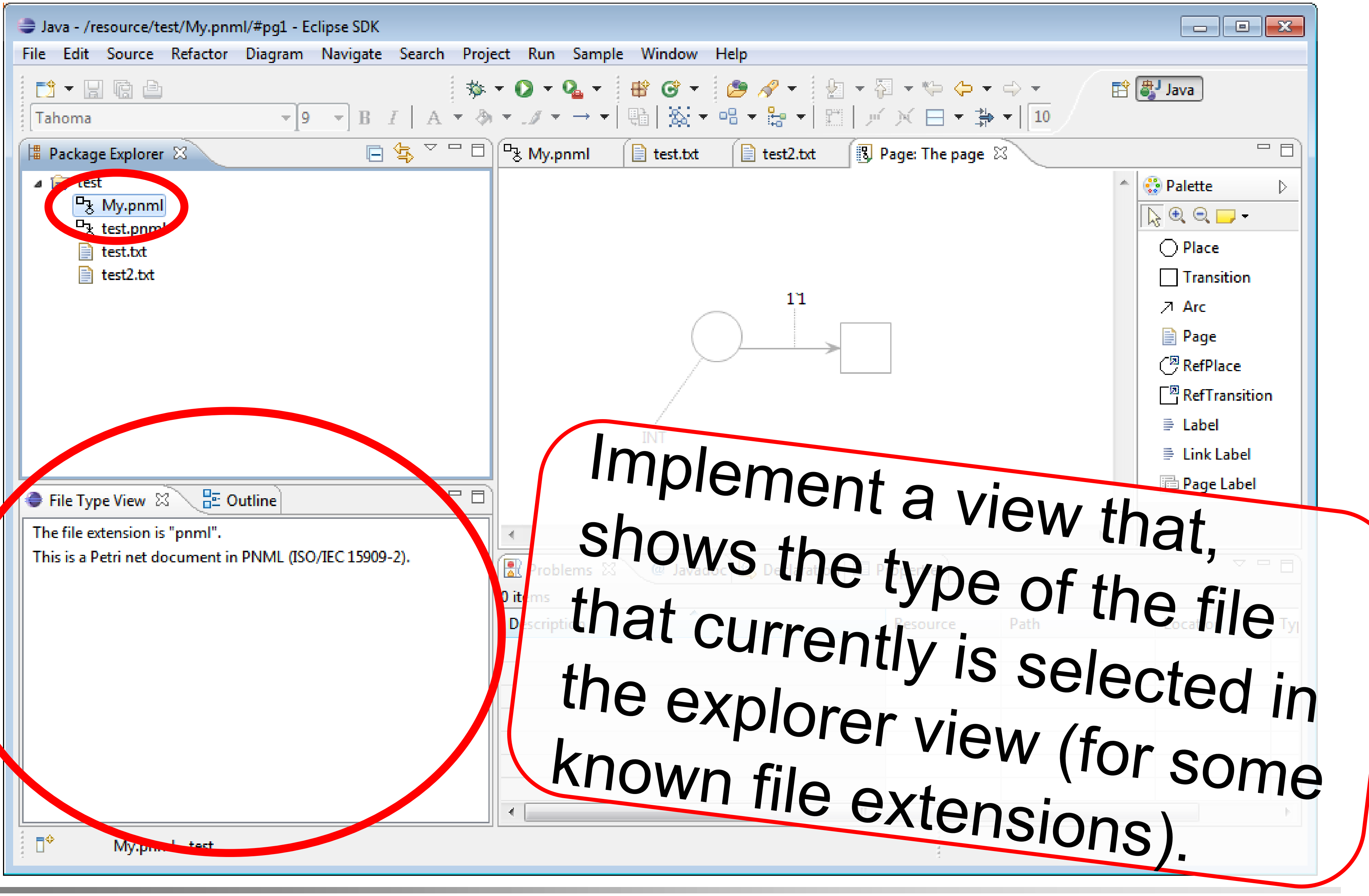
The workbench defines extension points that allow plug-ins to contribute behaviors to existing views and editors or to provide implementations for new views and editors. Using commands is covered in the Basic workbench extension points using commands section. Here we are going to take a look at the contributions to these extension points from one of the workbench sample applications, the readme tool.

The readme tool is a plug-in that provides custom editing and navigation for a specific resource , a **.readme** file. The example shows many typical (but simplified) ways that extensions can be used to provide specialized tools.

The readme tool contributes to the menus of the navigator view, adds editor related actions to the workbench menus and tool bar, defines a custom view and content outliner, and defines markers and marker resolutions. The figure below shows some of the customized features added to the workbench by the readme tool.

Use Eclipse
Help → Help Contents

3. Task: File Type View



The file extension is "pnml".
This is a Petri net document in PNML (ISO/IEC 15909-2).

Implement a view that, shows the type of the file that currently is selected in the explorer view (for some known file extensions).

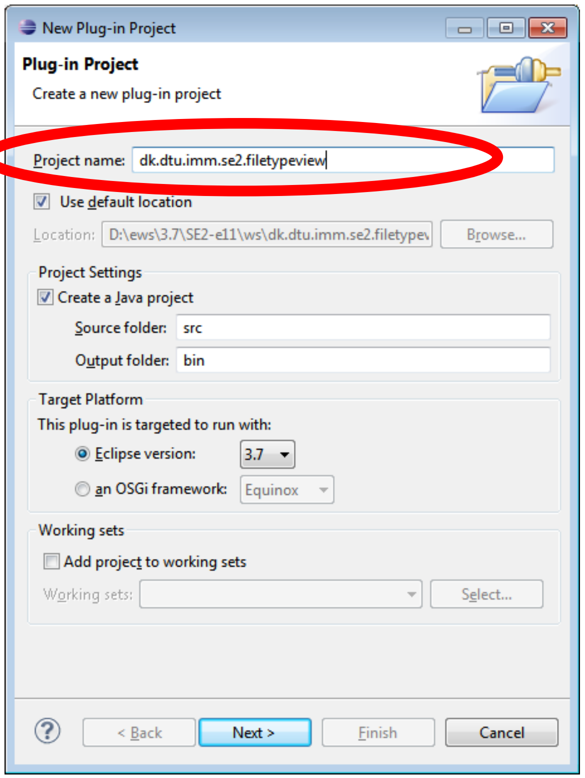
Steps

- The easiest way to getting started is to use an Eclipse wizard that creates a default view; and then change that view

NB: This is just for getting started; later you will create such plugins manually.

- Start Wizard: File → New → Plug-in Project

Wizard



New Plug-in Project

Plug-in Project

Create a new plug-in project

Project name: dk.dtu.imm.se2.filetypeview

[✓] Use default location

Location: D:\ews\3.7\SE2-e11\ws\dk.dtu.imm.se2.filetypev | Browse...

Project Settings

[✓] Create a Java project

Source folder: src

Output folder: bin

Target Platform

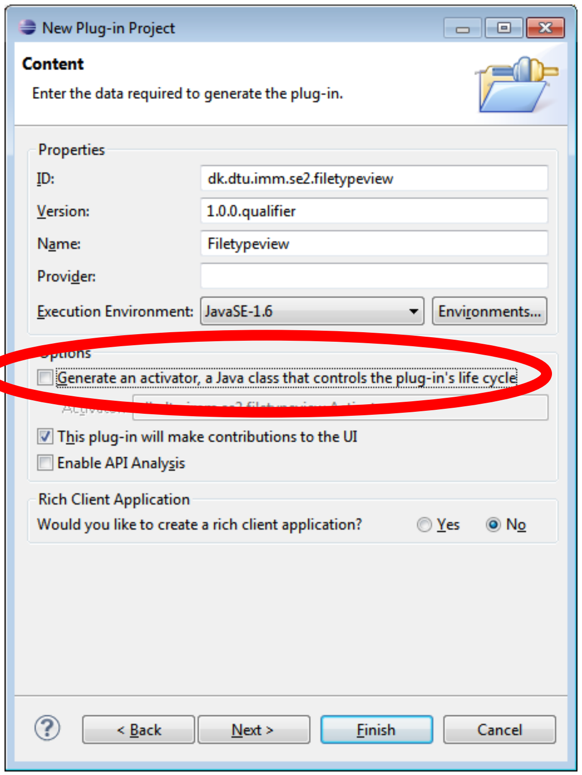
This plug-in is targeted to run with:

[✓ Eclipse version: 3.7] an OSGi framework: Equinox

Working sets

[ ] Add project to working sets

Working sets: | Select...



New Plug-in Project

Content

Enter the data required to generate the plug-in.

Properties

ID: dk.dtu.imm.se2.filetypeview

Version: 1.0.0.qualifier

Name: Filetypeview

Provider:

Execution Environment: JavaSE-1.6 | Environments...

Options

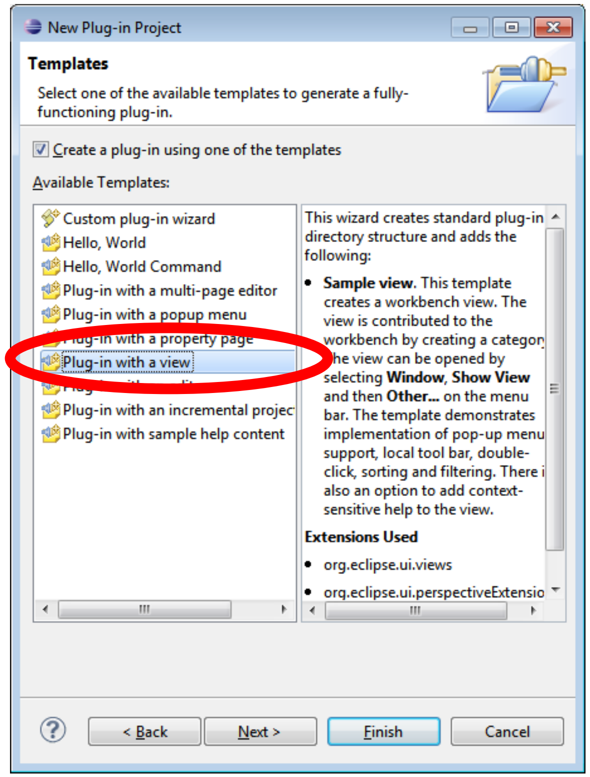
[ ] Generate an activator, a Java class that controls the plug-in's life cycle

[✓] This plug-in will make contributions to the UI

[ ] Enable API Analysis

Rich Client Application

Would you like to create a rich client application? Yes [✓ No]



New Plug-in Project

Templates

Select one of the available templates to generate a fully-functioning plug-in.

[✓] Create a plug-in using one of the templates

Available Templates:

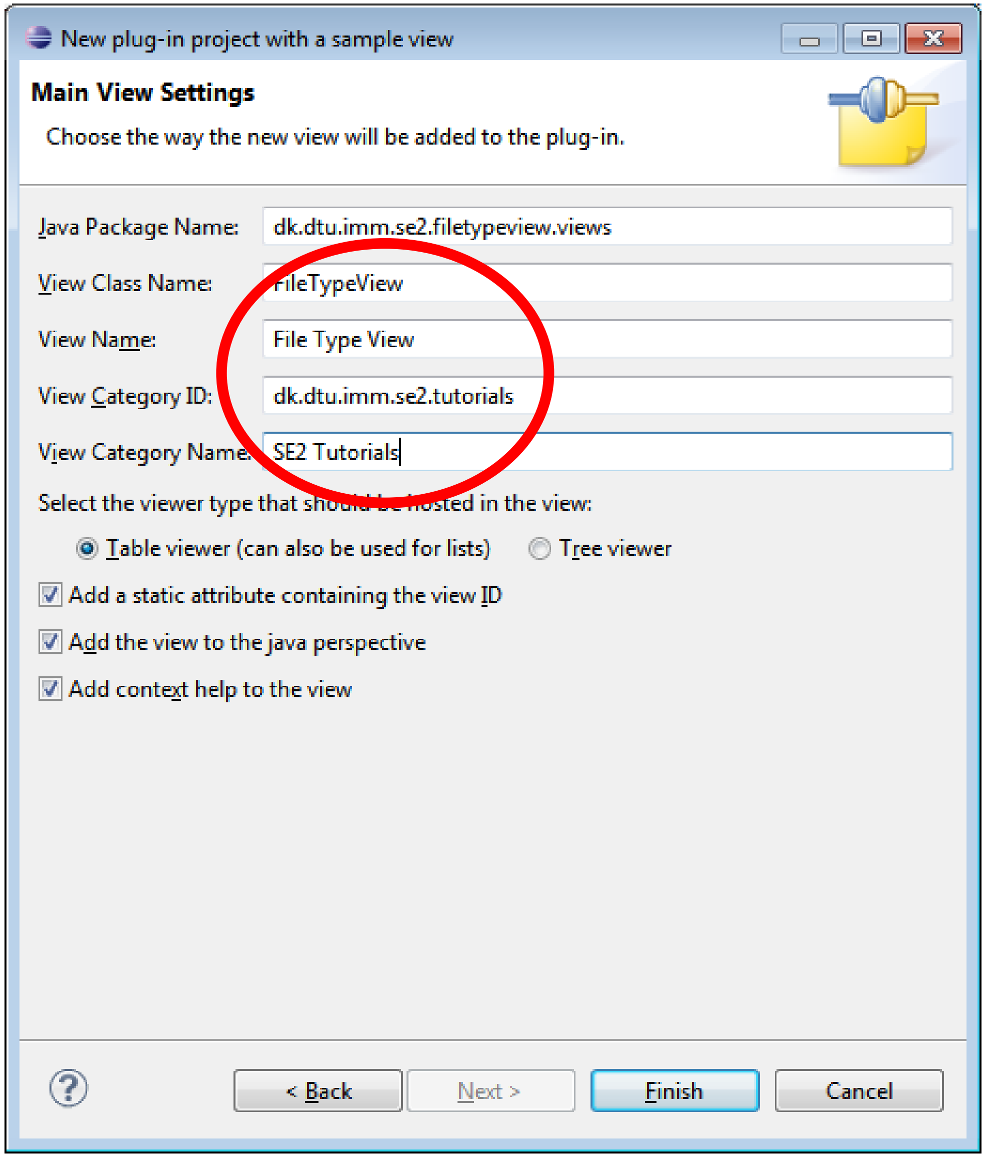
- Custom plug-in wizard
- Hello, World
- Hello, World Command
- Plug-in with a multi-page editor
- Plug-in with a popup menu
- Plug-in with a property page
- [✓ Plug-in with a view]
- Plug-in with an incremental projec
- Plug-in with sample help content

This wizard creates standard plug-in directory structure and adds the following:

- **Sample view.** This template creates a workbench view. The view is contributed to the workbench by creating a categor[y.] The view can be opened by selecting **Window, Show View** and then **Other...** on the menu bar. The template demonstrates implementation of pop-up menu support, local tool bar, double-click, sorting and filtering. There i[s] also an option to add context-sensitive help to the view.

Extensions Used

- org.eclipse.ui.views
- org.eclipse.ui.perspectiveExtensio


New plug-in project with a sample view

Main View Settings

Choose the way the new view will be added to the plug-in.

Java Package Name:	dk.dtu.imm.se2.filetypeview.views
View Class Name:	FileTypeView
View Name:	File Type View
View Category ID:	dk.dtu.imm.se2.tutorials
View Category Name:	SE2 Tutorials

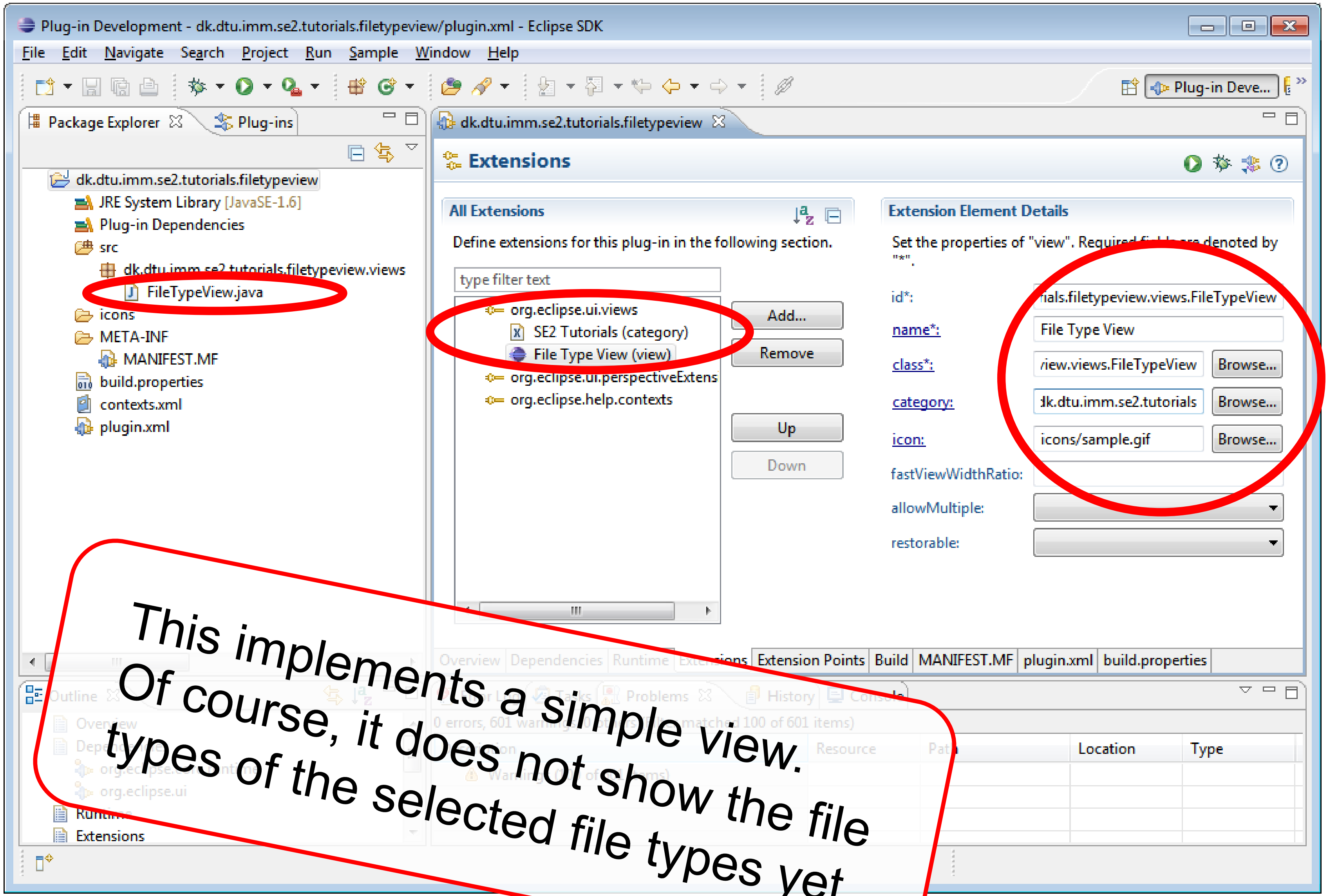
Select the viewer type that should be hosted in the view:

● Table viewer (can also be used for lists) ○ Tree viewer

☑ Add a static attribute containing the view ID

☑ Add the view to the java perspective

☑ Add context help to the view



Plug-in Development - dk.dtu.imm.se2.tutorials.filetypeview/plugin.xml - Eclipse SDK

Package Explorer:
- dk.dtu.imm.se2.tutorials.filetypeview
  - JRE System Library [JavaSE-1.6]
  - Plug-in Dependencies
  - src
    - dk.dtu.imm.se2.tutorials.filetypeview.views
      - FileTypeView.java
  - icons
  - META-INF
    - MANIFEST.MF
  - build.properties
  - contexts.xml
  - plugin.xml

Extensions

All Extensions

Define extensions for this plug-in in the following section.

- org.eclipse.ui.views
  - SE2 Tutorials (category)
  - File Type View (view)
- org.eclipse.ui.perspectiveExtens
- org.eclipse.help.contexts

Extension Element Details

Set the properties of "view". Required fields are denoted by "*".

id*:	ials.filetypeview.views.FileTypeView
name*:	File Type View
class*:	/iew.views.FileTypeView
category:	dk.dtu.imm.se2.tutorials
icon:	icons/sample.gif
fastViewWidthRatio:	
allowMultiple:	
restorable:	

This implements a simple view. Of course, it does not show the file types of the selected file types yet.

Impementing File Types

- Make the FileTypeView class implement the interface **`ISelectionListener`**

- In the **`createPartControl`** method, add this view as a selection listener to the Eclipse Workbench
- In the dispose method, make sure that the view unregisters itself as a selection listener again

- In the **`createPartControl`** change the **`viewers`** content provider to the **`ArrayContentProvider`** and the LabelProvider to the **`LabelProvider`**

Impementing File Types

- Implement the method **`selectionChanged()`** so that the view content is updated when the selection changes: To this end,
  - analyse the current selection, check whether only one element is selected,
  - get the selected element, check whether it is of type IFile (add **`org.eclipse.core.resources`** to the dependencies) ,
  - get the file extension; based on this extension, create an array of strings that shows the respective information and pass this array as input to the viewer

Clean up the Plug-in project

- The automatically created view plug-in contains many other things (e.g. actions added to the menu bar and the view).

  Clean up what you do not need.

For further inspiration, see slides 3 and 5.

You can also have a look at the Readme Example, that comes with Eclipse (see web pages how to install this example).

Tasks (until 13. 9. 2012)

- Install Eclipse (with EMF, GMF and SVN): see
  http://www2.imm.dtu.dk/courses/02162/e113project/eclipse-installation.html

- Get acquainted to the use of Eclipse as an IDE: see Eclipse Help (cf. p. 7)

- Install the ePNK: see
  http://www2.imm.dtu.dk/courses/02162/e13/project/
  and start it and create some first nets (see ePNK manual)

- Implement the File Type View that for, a selected resource, shows the file extension and, for known extensions (pdf, doc, txt, …), the file type as text.