Eclipse

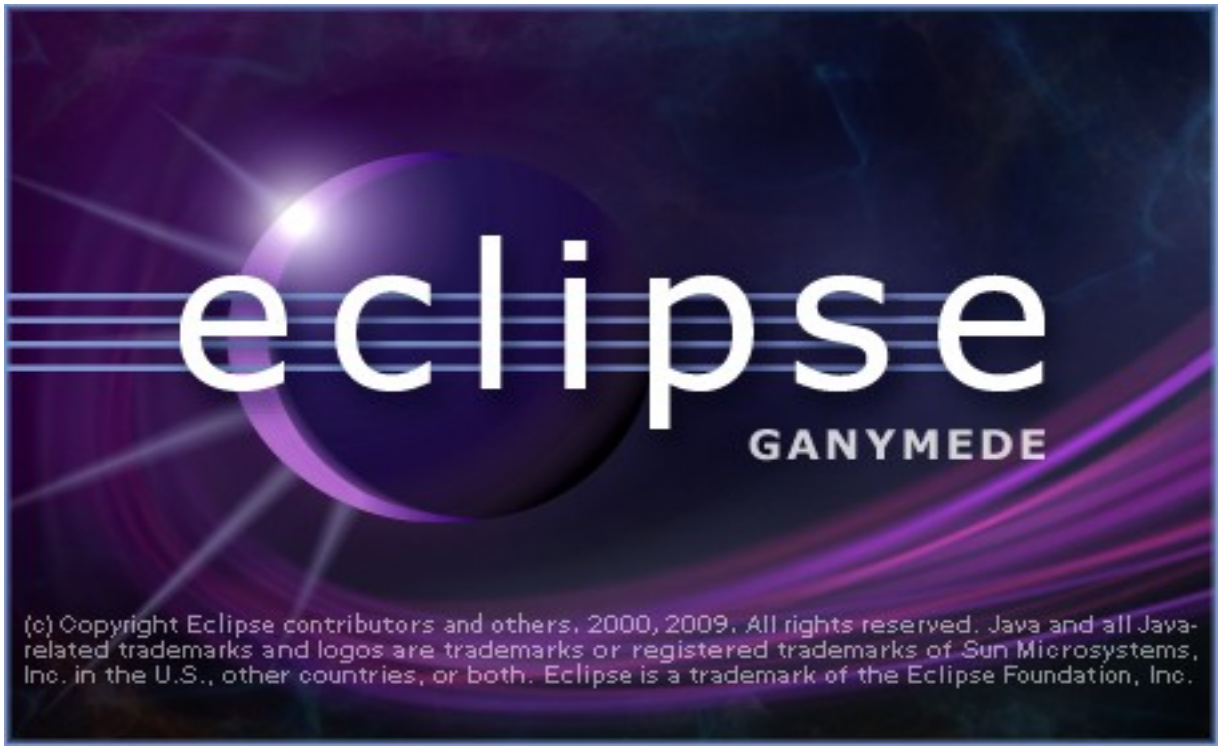
Eclipse @ Tues: Overview

k.mitov@sap.com

Eclipse

"Eclipse is an open source community whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle"

www.eclipse.org



k.mitov@sap.com

Eclipse

Eclipse as a Java Integrated Development Environment (IDE)

Eclipse Software Development Kit (SDK)

Eclipse is a platform for building IDEs

Eclipse is a platform for building tools

Eclipse is a Rich Client Platform

Eclipse is an open source project

Eclipse is a community and an eco-system

Eclipse is also a foundation

http://wbeaton.blogspot.com/2007/05/what-is-eclipse.html

k.mitov@sap.com

Eclipse

Eclipse се състои от множество проекти. Всеки проект може да има свои под проекти.

- Eclipse Project
- Eclipse Tools Project
- Eclipse Modeling Project
- Eclipse Technology Project
- Eclipse Web Tools Platform Project
- The Eclipse Test and Performance Tools Platform (TPTP) Project
- Business Intelligence and Reporting Tools (BIRT) Project
- Data Tools Platform Project (DTP)
- и много повече...

k.mitov@sap.com

Eclipse

Когато свалим Eclipse SDK всъщност сваляме четири основни проекта.
Тези проекти биват

k.mitov@sap.com

Equinox

- представлява имплементация на OSGi спецификацията - www.osgi.org
- Основата върху, която се изгражда всичко останало в Eclipse.
- OSGi служи с понятието bundle – компонента изградена от набор от класове имаща определена структура. Даден bundle посредством класовете, които съдържа предоставя на останалите bundle-и определена функционалност, от която те могат да се възползват.
- Даден bundle може да бъде инсталиран в системата, стартиран, спиран и деинсталиран от системата.
- Структурирането и изграждането на системата от bundle-и позволява висока модулност. Системата може да бъде изградена от ограничено количество bundle-и без наличието на излишна функционалност.

k.mitov@sap.com

Platform

- проектът предоставя основните рамки(frameworks) и услуги(services), върху които се изграждат plugin-ите.
- Служи с понятието plugin - компонента изградена от набор класове имаща определена структура.
- Всеки plugin представлява bundle, но не всеки bundle е plugin. Разликата е в това, че всеки plugin може да предоставя допълнителни възможности и разширения.
- Проектът им за цел да изгради платформа, върху която другите програмисти с лекота да могат да изградят своите приложения.
- Някой подпроекти
  - SWT
  - UI – JFace и Workbench
  - Core
  - Debug

k.mitov@sap.com

JDT

- предоставя набор от plugin-и, с основна цел да предоставят възможност за разработка на Java приложения използвайки Eclipse
- Пример за компоненти част от JDT проекта
  - Java перспективата
  - Всички изгледи, иконки, менюта и т.н., които са част от Java и Java Browsing перспективата
  - Java проектът и всички настройки относно Java в Window ->Preference.
  - Java компилатор – Основният компилатор използван за обработка на Java код е предоставен от Sun Microsystems® и се казва javac. JDT предоставя свой компилатор осигуряващ допълнетелни възможности.
  - и много повече
- Позволява Eclipse да бъде използван за разработката на Eclipse

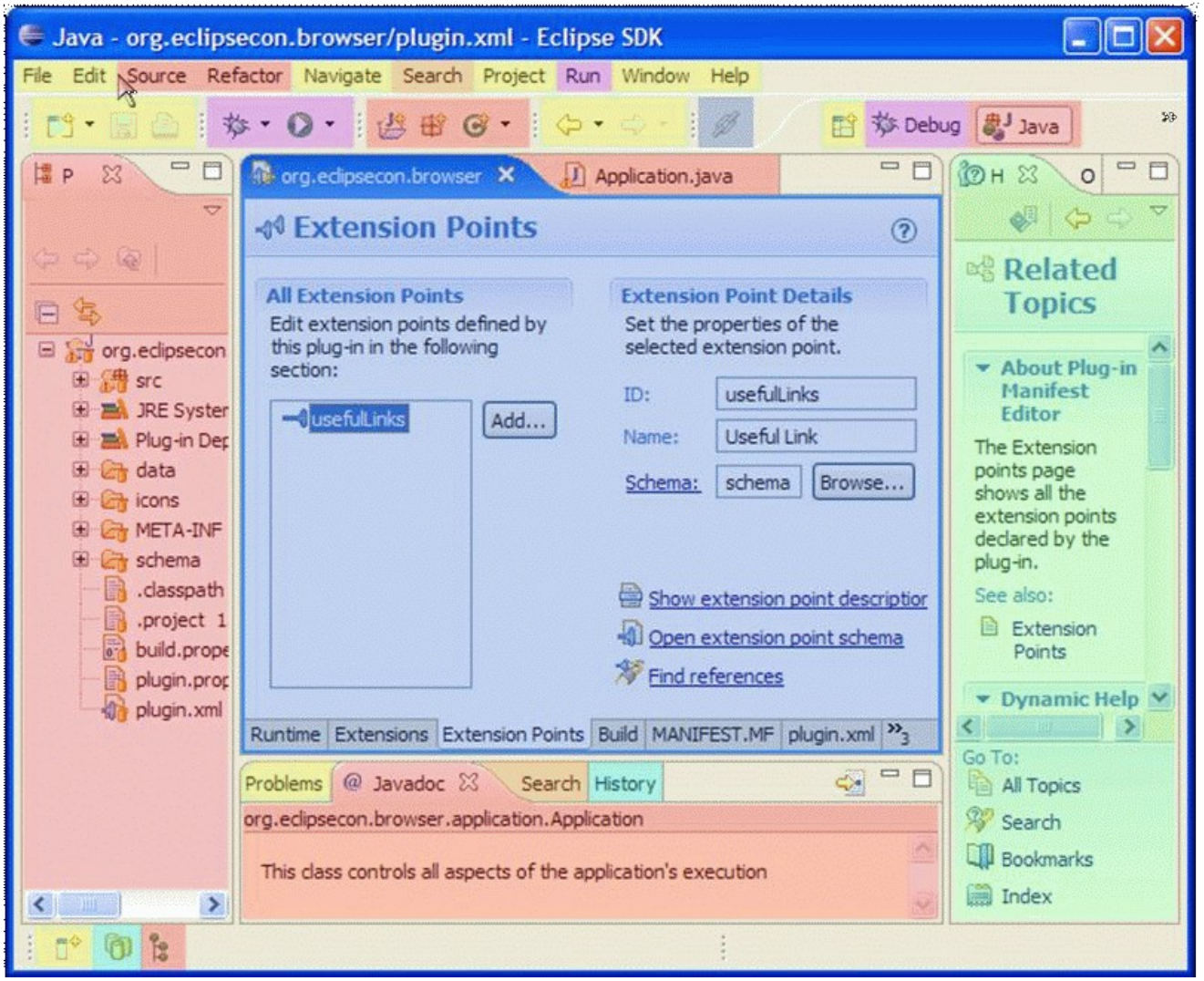
k.mitov@sap.com

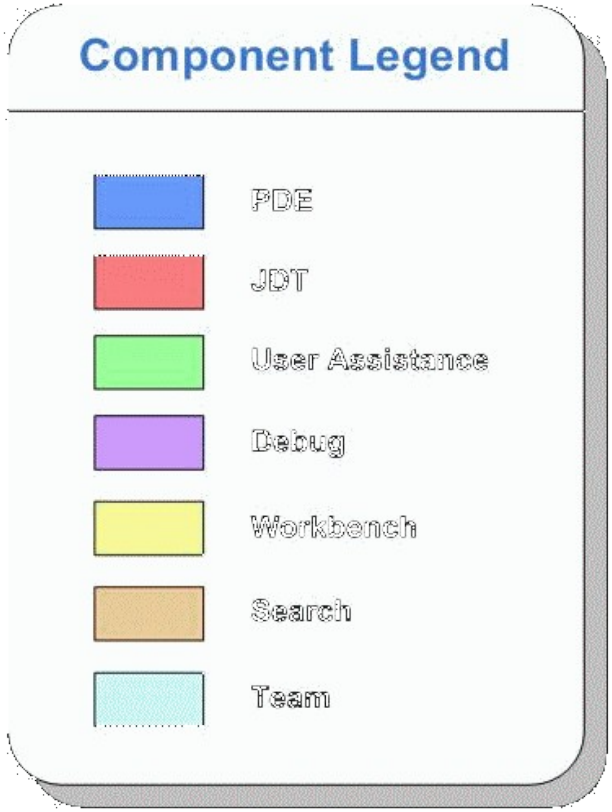
PDE

- предоставя възможности за създаване, разработване и изграждане на plugin-и
- Всичко в Eclipse е plugin. Обсолютно всяка функционалност, характеристика или услуга предоставена от платформата е капсулирана и обосновена като част от някой plugin. Не съществува такова нещо като – основна част и добавки (plugins). Всичко е добавка (plugin).
- За да се улесни изграждането на plugin-и, PDE проектът предоставя значителни улеснения:
  - Перспектива
  - Изгледи, редактори, иконки, менюта, магюсници
  - Възможност за стартиране на Runtime Workbench – стартираме Eclipse чрез Eclipse.
  - и много други

k.mitov@sap.com

Eclipse

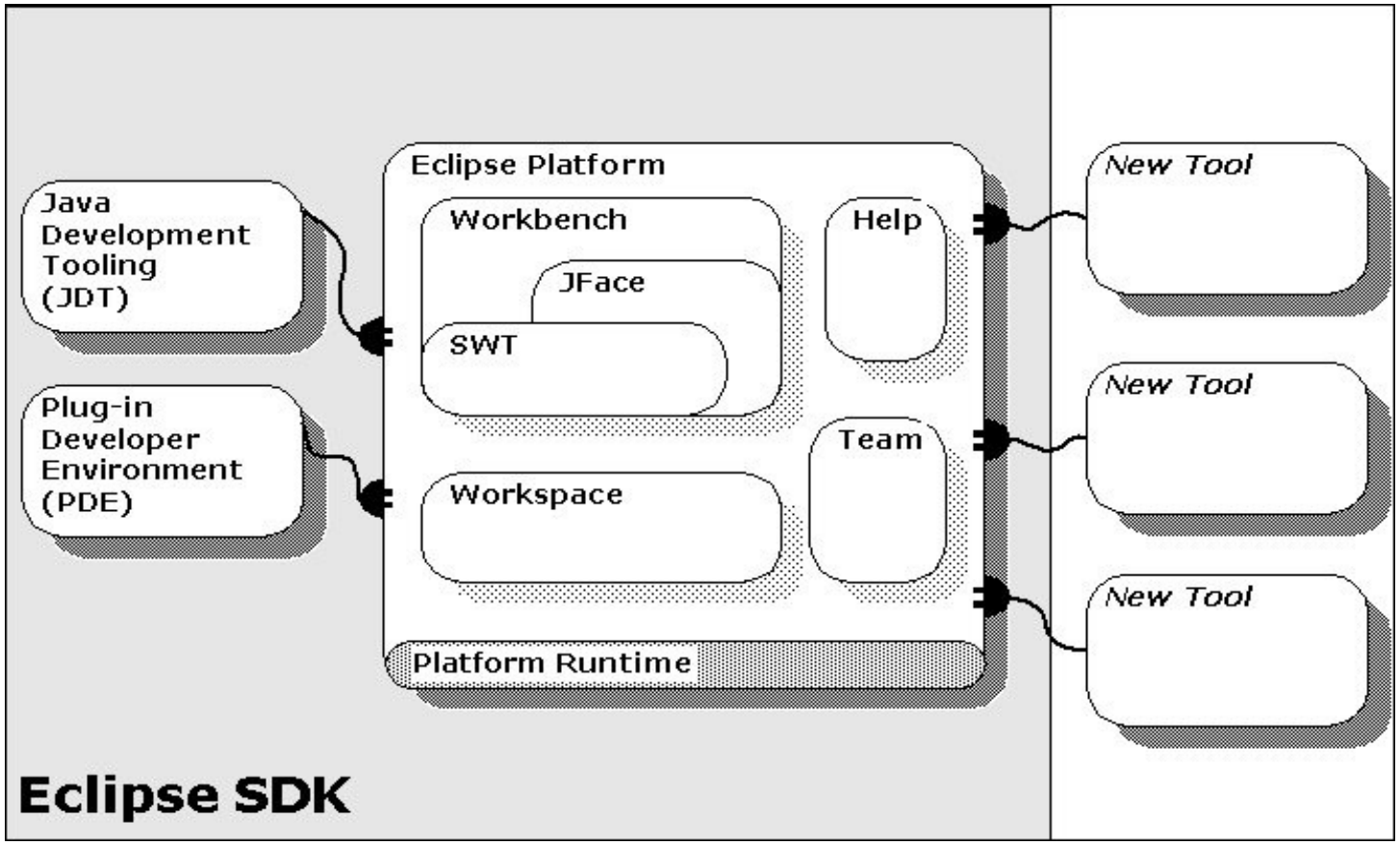




k.mitov@sap.com

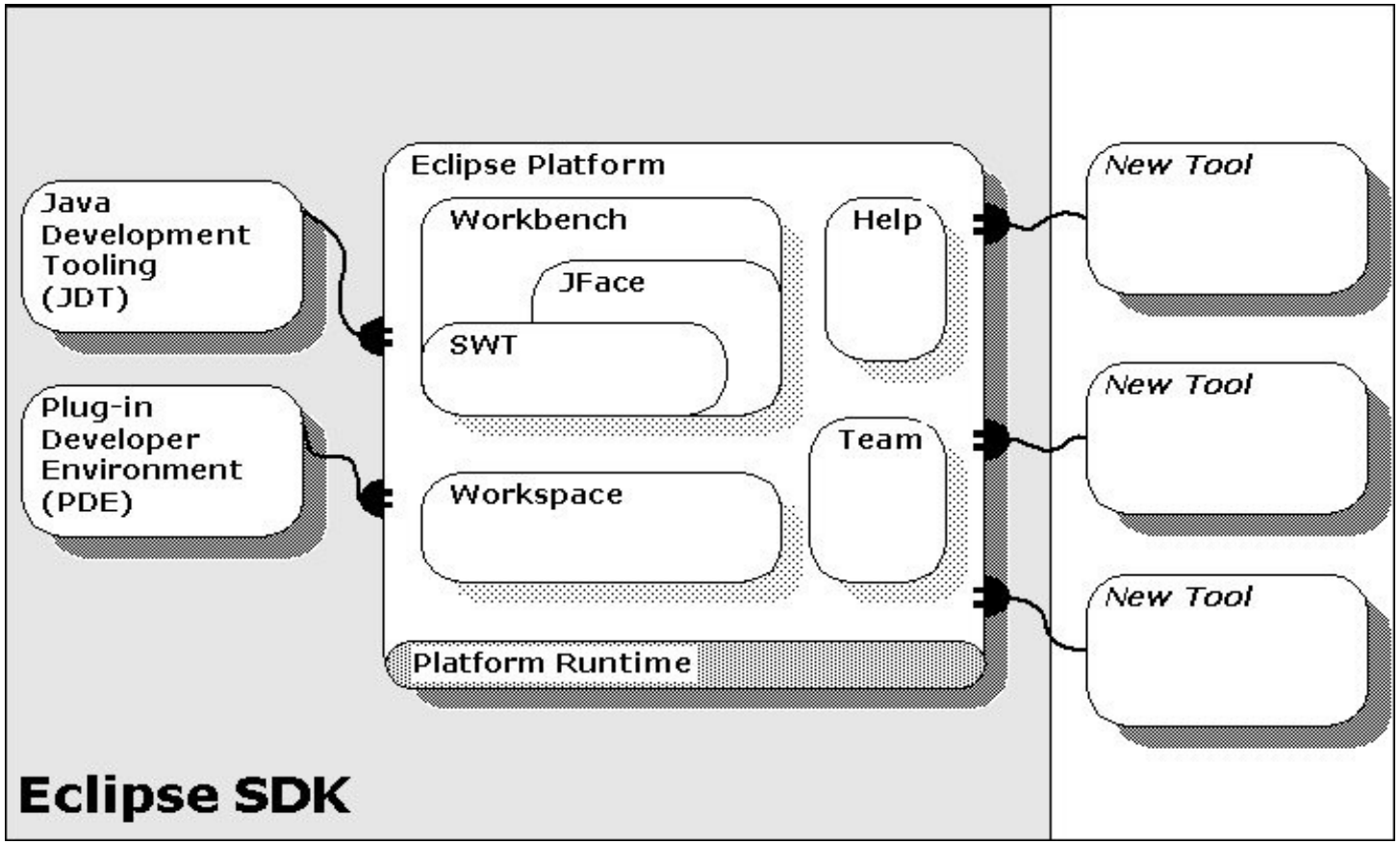
Eclipse

Архитектурата на Eclipse е изключително гъвкава като позволява използването на различни продукти в рамките на платформата, допълнително разширяване на тези продукти както и интеграция между продуктите и платформата.



k.mitov@sap.com

Eclipse



JDT и **PDE** представляват едно допълнение към платформата. Разширението на платформата се извършва чрез така наречените **extension points.** Правилата за разширение и допълнение са еднакви за всички plugin-и. Няма **“забранени класове”**, няма **“скрити техники”** и тн.

k.mitov@sap.com

Plugin Example

Всичко в Eclipse е plugin. Ще изясним това понятие чрез следния пример.

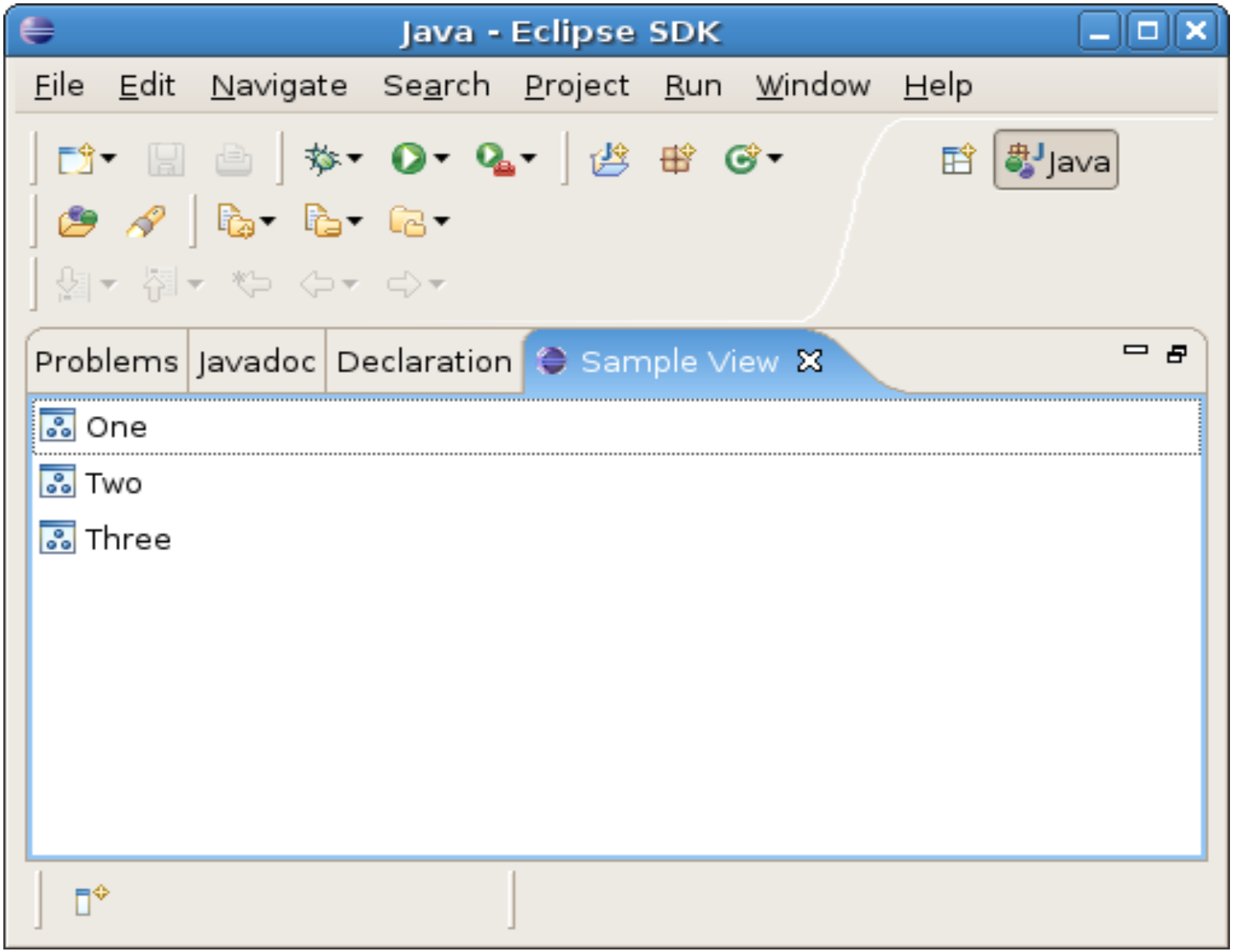
Пример:

- **В Eclipse има голям брой изгледи (Views) предназначени да изобразяват различна информация. Всичките изгледи предоставени от текущо инсталираните plugin-и може да бъдат видяни чрез Windows -> Show View...**
- **Целта на примера е да се изгради plugin-и, който да предоставя изглед.**
- **Изгледът трябва да има три реда, в които да е изписано съответно One, Two, Three.**

k.mitov@sap.com

Plugin Example

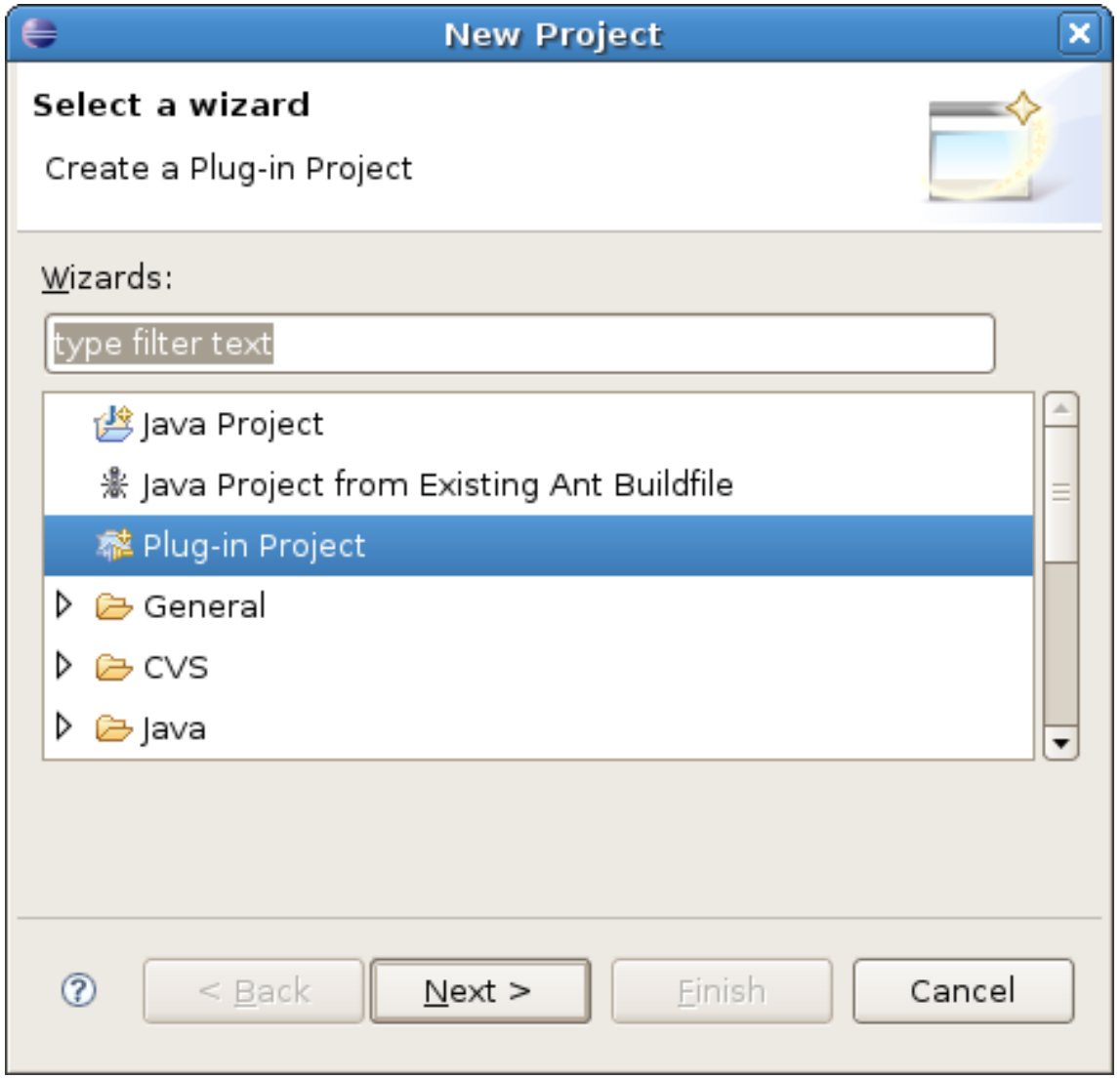
Резултатът от примера ще изглежда по следния начин:



k.mitov@sap.com

Plugin Example

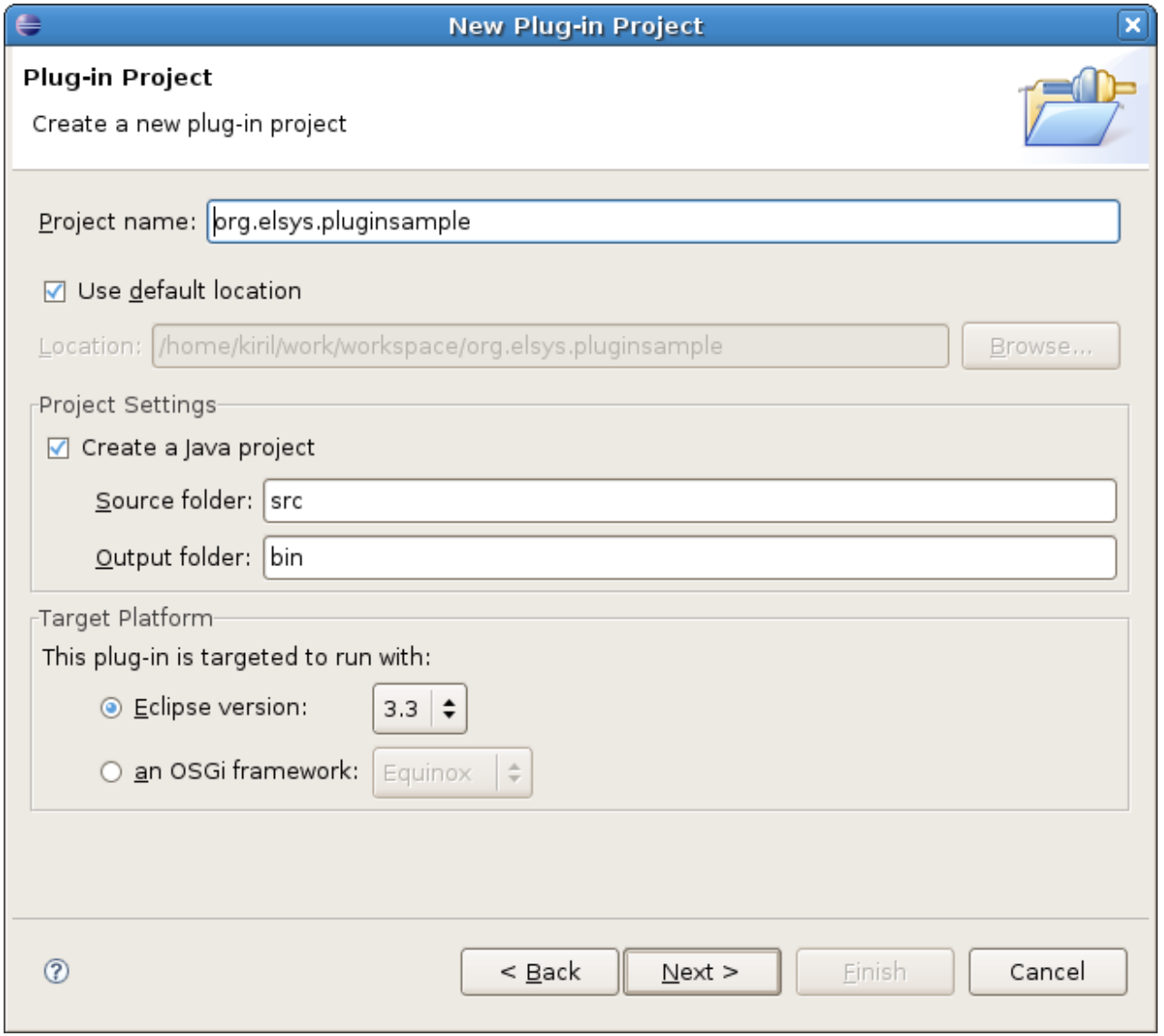
Първата стъпка представлява създаването на Plugin Project. New->Project->Plug-in Project.



k.mitov@sap.com

Plugin Example

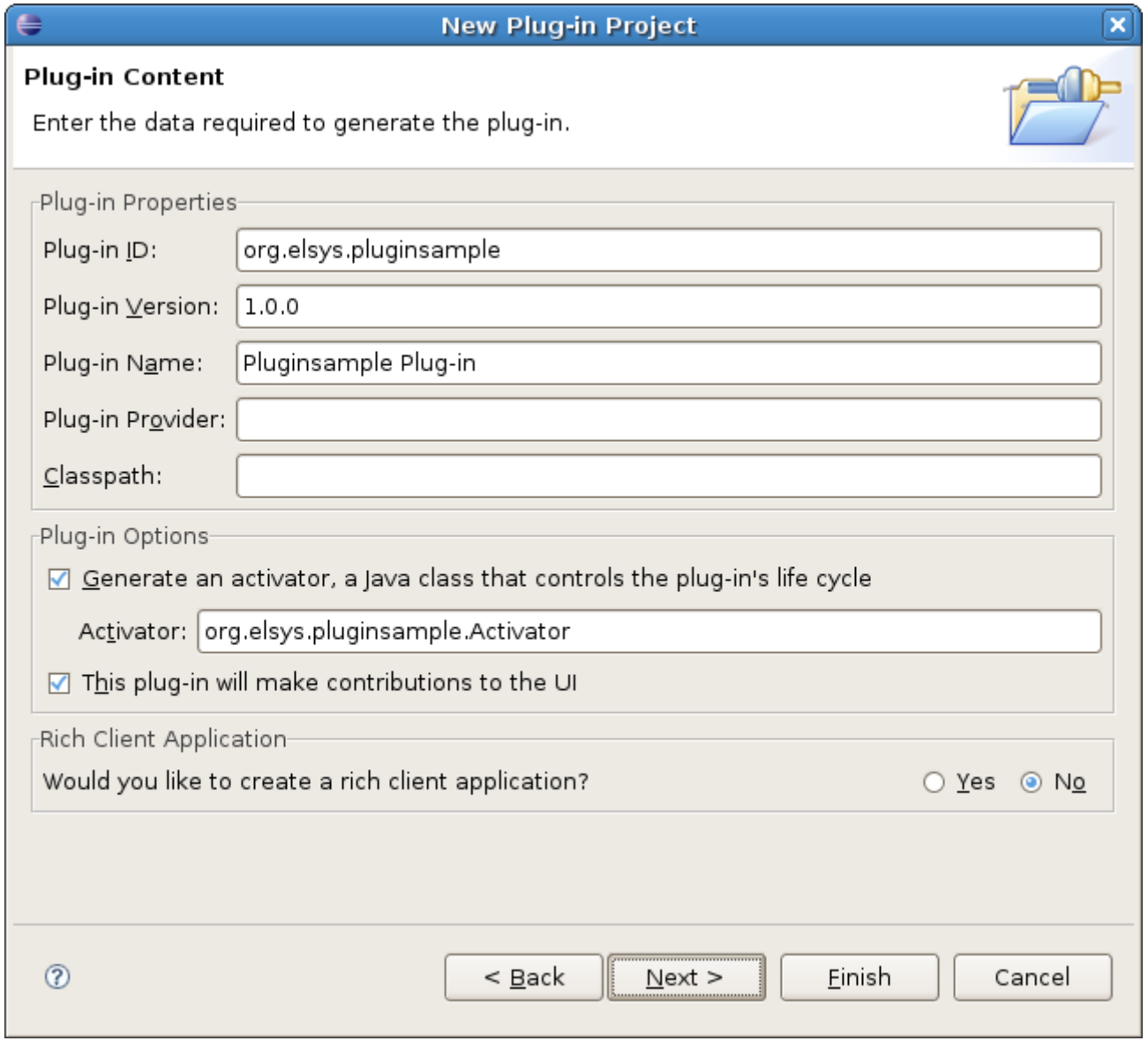
На втората страница на помощника се задава името на дадения plugin. В случая това име ще бъде org.elsys.pluginsample.



k.mitov@sap.com

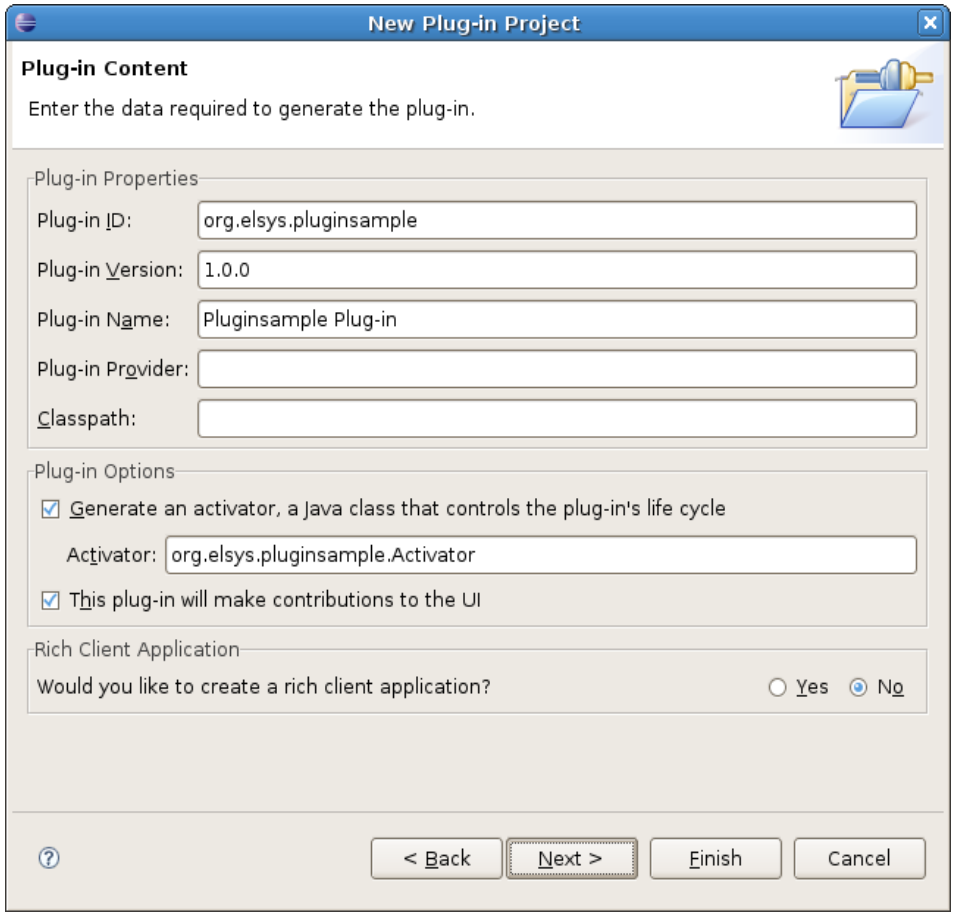
Plugin Example

Следва дефиниране на характеристиките на този plugin.



k.mitov@sap.com

Plugin Example



Основно всеки plugin има:

- **идентификатор – org.elsys.pluginsample**
- **версия – 1.0.0**
- **име разбираемо за потребителите – Pluginsample Plug-in,**
- **Разработчик**
- **Клас “Activator”**

k.mitov@sap.com

Plugin Example - Съдържание

На следващата стъпка помощникът ни предоставя възможност да създадем plugin с предварително определено съдържание. Наподобява идеята на шаблони (templates) от други софтуерни продукти.
Тъй като искаме да създадем plugin предоставящ изглед избираме „Plug-in with a view“

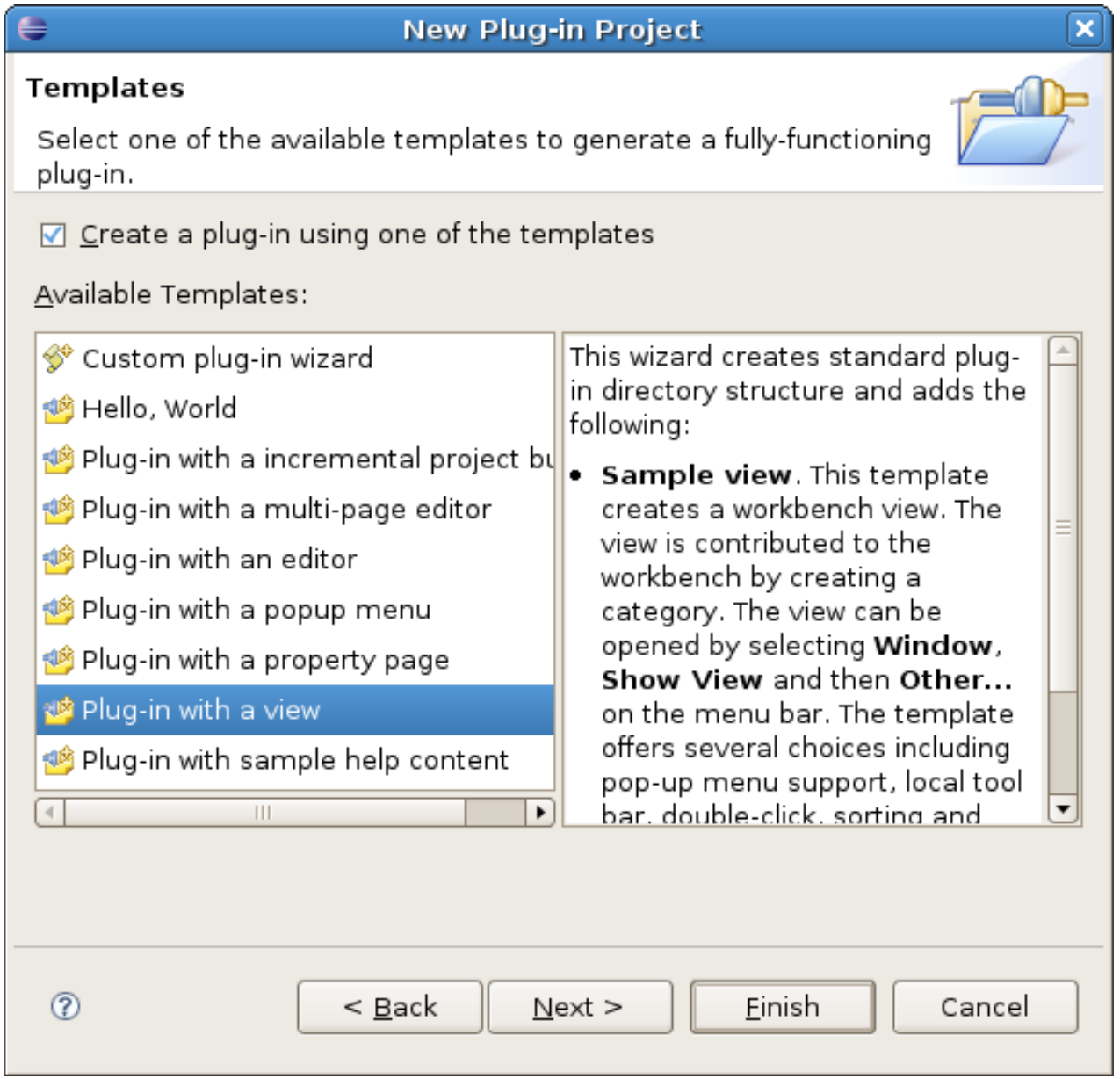
Някой от останалите шаблони предоставят възможност за създаване на plugin предоставящ:

- редактор – “Plug-in with an editor”
- меню - “Plug-in with a popup menu”
- примерно помощтно съдържание - “Plug-in with sample help content”

Възможно е да създаден plugin проект без да използваме никой от предложените шаблони.

k.mitov@sap.com

Plugin Example



k.mitov@sap.com

Plugin Example

Досега с помощта на помощника постигнахме:

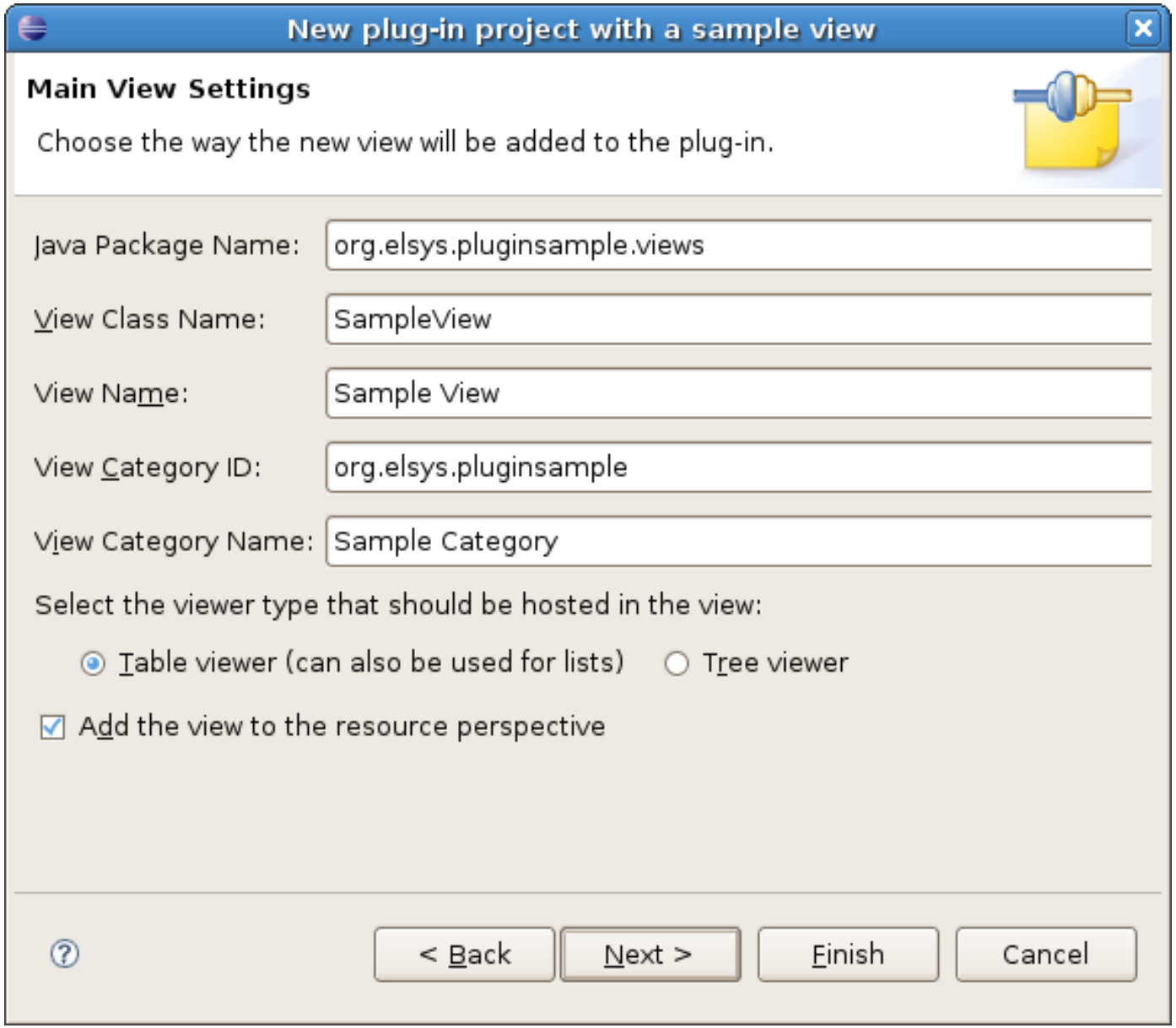
- Определихме идентификатора на разработвания plugin,
- неговата версия, и име на разработчика,
- както и това, че разработваният plug-in трябва да предоставя изглед, който да може да се покаже на потребителя. За създаването на изглед се използва предварително дефиниран шаблон.

Предстои да се определят специфичните характеристики на изгледа.

- Такива биват името на класа представляващ изгледа, името на пакета, в който се намира този клас, името на изгледа както и това дали изгледът да съдържа данните в таблична или дървовидна форма.
- Извършва се на следващата страница от магьосника.

k.mitov@sap.com

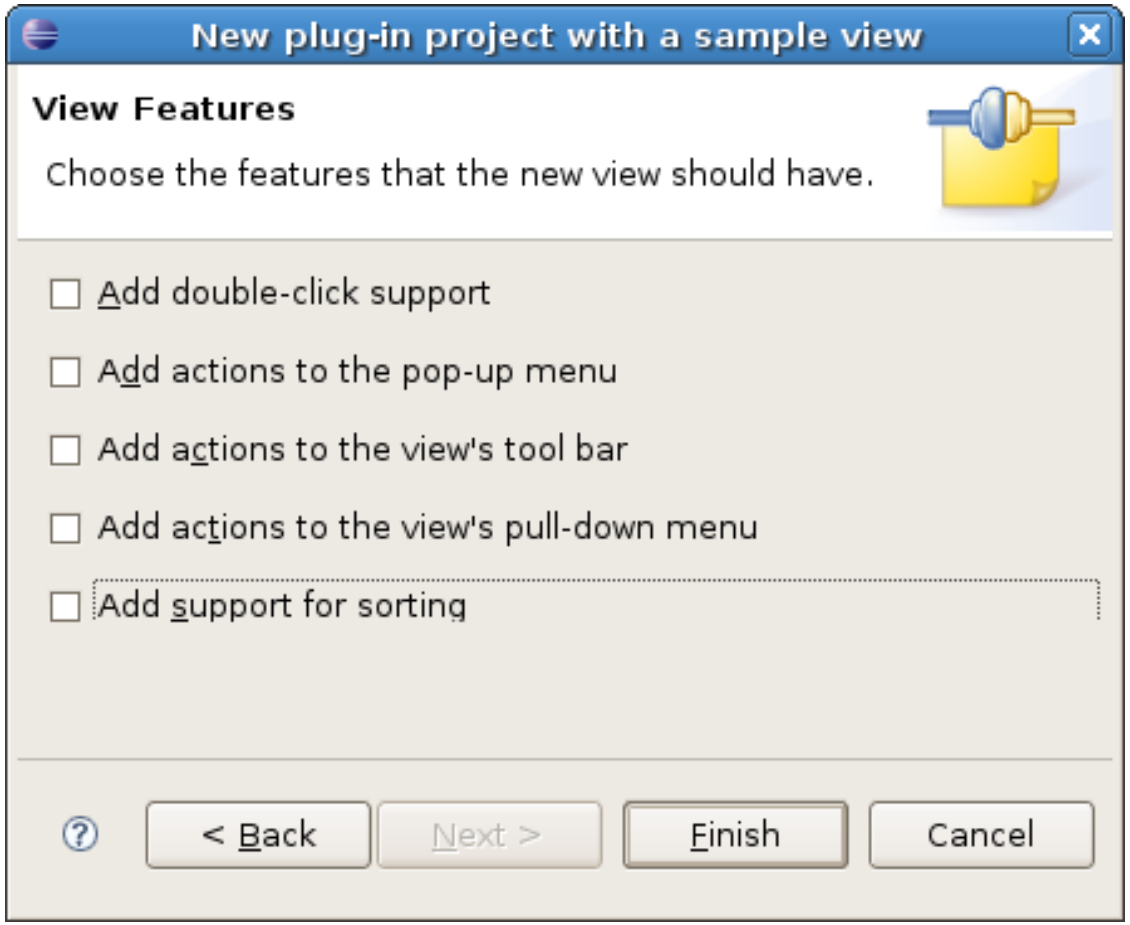
Plugin Example



k.mitov@sap.com

Plugin Example

Последната страница от магьосника дава възможност за добавяне на допълнителна функционалност към изграждания изглед. За простота на примера премахваме всички отметки, така че съответната функционалност да не бъде поддържана.

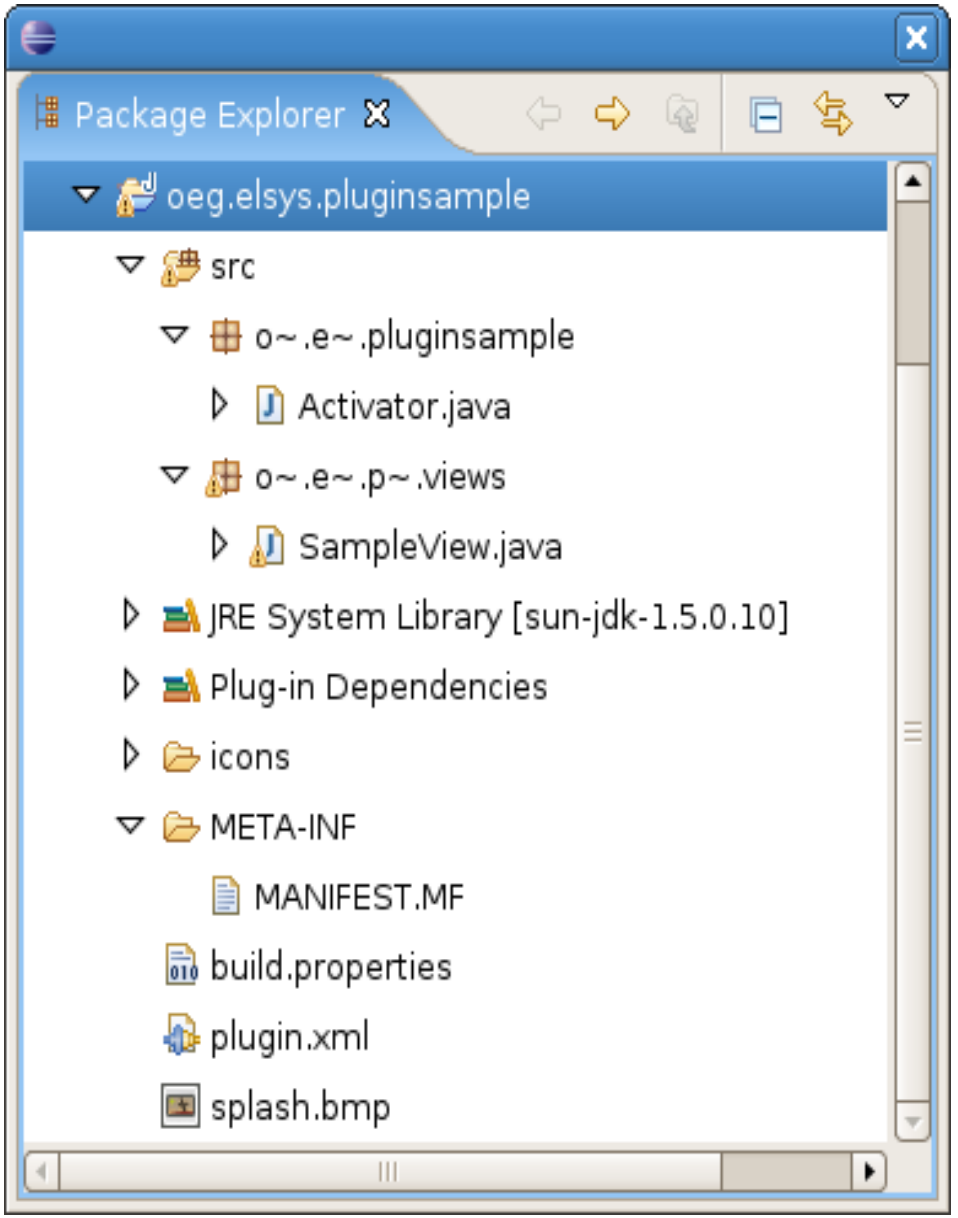


k.mitov@sap.com

Plugin Example

Стартиране на Plug-in проекта

С помощта на магьосника беше създаден нов plugin проект.
Структурата му е показана на фигурата.



k.mitov@sap.com

Plugin Example

Единствената ни цел за момента е да можем да стартираме нашия plugin и да видим резултата от неговата работа.
Всеки Plug-in добавя определена функционалност към платформата, в която е инсталиран. Самият Plug-in не може да бъде „стартиран“ като самостоятелна единица.

Target Platform

- **платформата с помощта на която разработваме нашия plugin. Платформата, с която текущо работим. В нея има инсталирани и стартирани множество plugin-и, но текущо разработваният от нас не е инсталиран и следователно не може да бъде стартиран. Необходимо е стартирането на друга платформа. PDE проектът предоставя тази възможност.**

k.mitov@sap.com

Plugin Example

Runtime Platform

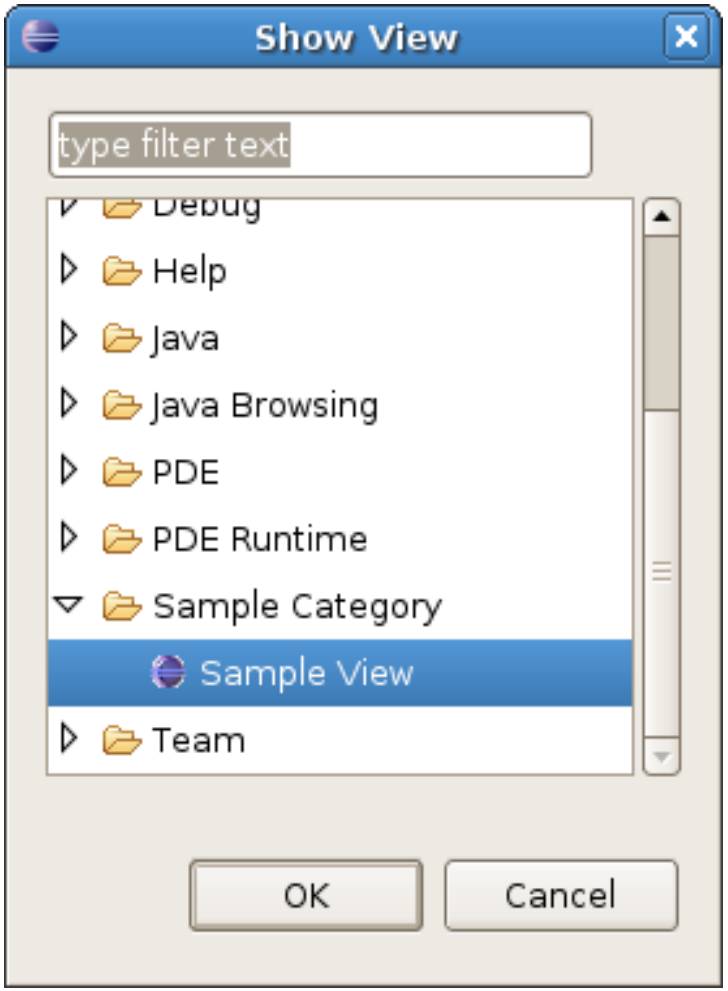
- С помощта на Target Platform може да стартираме нова инстанция на платформата, т.е. да стартираме нов Eclipse.
- Разликата е, че примерният plugin ще добави своята функционалност към новостартираната инстанция на платформата.
- Тази нова инстанция се нарича Runtime Platform или Runtime Workbench. Стартирането на инстанцията става от менюто Run-> Run As -> Eclipse Application.
- По подразбиране новостартираната платформа съдържа същите plugin-и както и Target платформата (това може да се конфигурира), но съдържа и разработения org.elsys.pluginsample.

След стартирането на Runtime Platform можем да проверим резултата от примера – т.е. да покажем разработение изглед

k.mitov@sap.com

Plugin Example

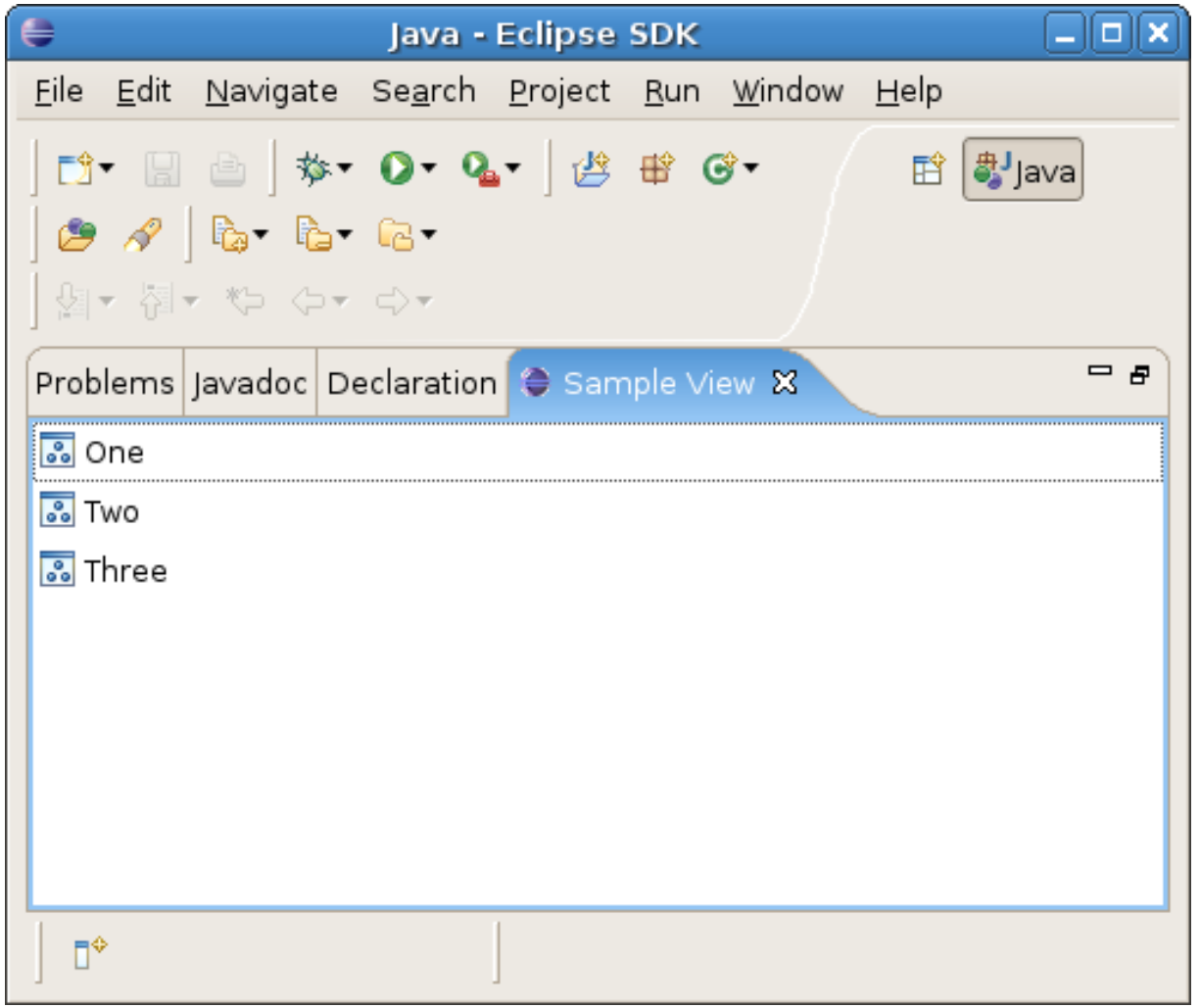
org.elsys.pluginsample предоставя изглед. Изгледът може да бъде отворен с помощта от Windows->Show View -> Other-> Sample Category -> Sample View



k.mitov@sap.com

Plugin Example

На фигурата е показан разработения изглед интегриран в платформата

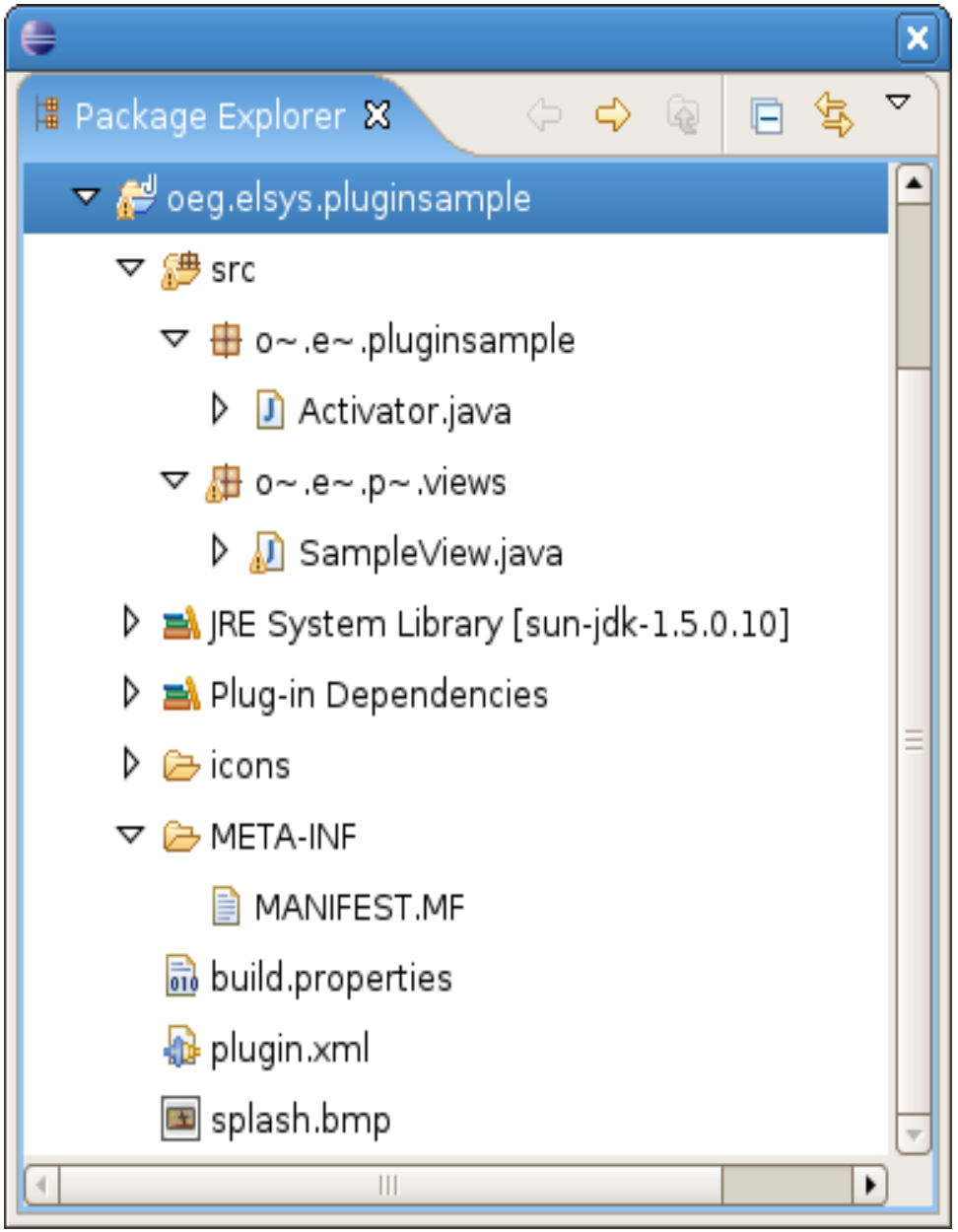


k.mitov@sap.com

Структура на org.elsys.pluginsample

Ще бъде разгледана структурата на проекта.

- Plugin manifests – META-INF/MANIFEST.MF и plugin.xml
- Plugin класът - Activator.java
- Класът на изгледа - SampleView.java



k.mitov@sap.com

Plugin manifests - MANIFEST.MF

Всеки plugin предоставя определена функционалност и взаимодейства с другите plugin-и инсталирани в платформата. Това взаимодействие се определя от съдържанието на META-INF/MANIFEST.MF и plugin.xml файловете

MANIFEST.MF

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Pluginsample Plug-in
Bundle-SymbolicName: org.elsys.pluginsample;
singleton:=true
Bundle-Version: 1.0.0
Bundle-Activator: org.elsys.pluginsample.Activator
Require-Bundle: org.eclipse.ui,
 org.eclipse.core.runtime
Eclipse-LazyStart: true
```


k.mitov@sap.com

Plugin manifests - MANIFEST.MF

`Bundle-Name:` - име на bundle-а.

`Bundle-SymbolicName:` - идентификатор на bundle-а.

`Bundle-Version:` - версия на bundle-а.

`Bundle-Activator:` - Всеки bundle може да бъде стартиран и спиран. При тези действия се извикват методите start() и stop() на класът org.elsys.pluginsample.Activator.

`Require-Bundle:` - Всеки bundle може да предоставя функционалност и следователно да зависи на такава предоставена от други bundle-и. В случая org.elsys.pluginsample зависи от два такива bundle-а.

k.mitov@sap.com

Plugin manifests - plugin.xml

```
<plugin>
   <extension point="org.eclipse.ui.views">
      <category name="Sample Category"
            id="org.elsys.pluginsample"/>
      <view name="Sample View"
           icon="icons/sample.gif"
           category="org.elsys.pluginsample"
           class="org.elsys.pluginsample.views.SampleView"
           id="org.elsys.pluginsample.views.SampleView"/>
   </extension>
   <extension
      point="org.eclipse.ui.perspectiveExtensions">
      <perspectiveExtension
            targetID="org.eclipse.ui.resourcePerspective">
         <view ratio="0.5"
               relative="org.eclipse.ui.views.TaskList"
               relationship="right"
         id="org.elsys.pluginsample.views.SampleView"/>
      </perspectiveExtension>
   </extension>
</plugin>
```


k.mitov@sap.com

Plugin manifests - plugin.xml

- Eclipse платформата е изградена, така че да позволява лесно добавяне на допълнителна функционалност. Механизмът се базира на така наречените „extension points“.
- В един обикновен xml файл (plugin.xml) се описва какво точно предоставя даденият plugin. В платформата е инсталиран plugin (org.eclipse.core.runtime) , който знае как да прочете plugin.xml файловете на другите plugin-и и който отговаря за интегрирането им с платформата.
- В случая нашият plugin предоставя изглед и добавя този изглед към Resource перспективата.
- org.elsys.pluginsample декларира разширение (extension), което добавя допълнителна категория изгледи наречена – Sample Category и допълнителен изглед – Sample View намиращ се в тази категория.

k.mitov@sap.com

plugin.xml – extension point

```
<extension point="org.eclipse.ui.views">
   <!-- code missed -->
</extension>
```

- `org.elsys.pluginsample` декларира разширение на функционалност предоставена от `org.eclipse.ui`.
- Това разширение е с идентификатор `org.eclipse.ui.views`.
- При необходимост (когато SampleView трябва да се активира) `org.eclipse.ui` ще предаде управлението на `org.elsys.pluginsample`

k.mitov@sap.com

plugin.xml – view

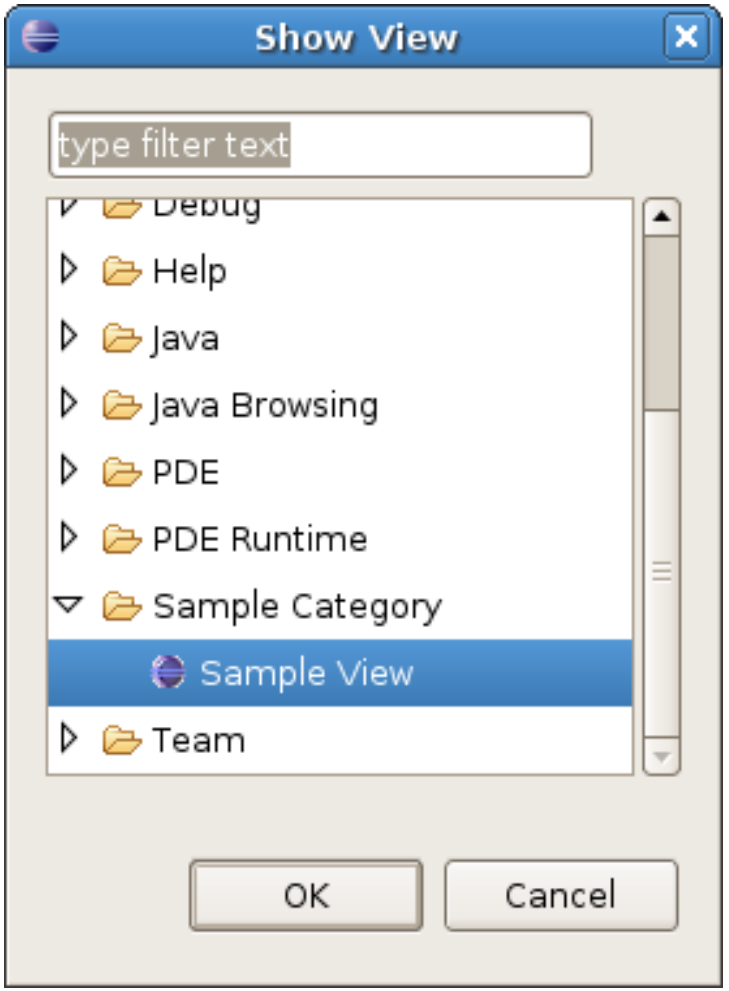
```
<extension point="org.eclipse.ui.views">
      <category name="Sample Category"
            id="org.elsys.pluginsample"/>
      <view name="Sample View"
            icon="icons/sample.gif"
            category="org.elsys.pluginsample"
            class="org.elsys.pluginsample.views.SampleView"
            id="org.elsys.pluginsample.views.SampleView"/>
</extension>
```

- `org.elsys.pluginsample` декларира изглед с име Sample View
- Всеки изглед има единтификатор зададен чрез атрибута id
- Атрибутът class показва кои клас капсулира функционалността на изгледа. Класът трябва да имплементира определен интерфейс `org.eclipse.ui.IViewPart`
- Изгледът може да се намира в определена категория. Категорията представлява чисто визуално групиране на изгледите

k.mitov@sap.com

plugin.xml – view category

```
<extension point="org.eclipse.ui.views">
      <category name="Sample Category"
            id="org.elsys.pluginsample"/>
      <!-- code missed -->
</extension>
```



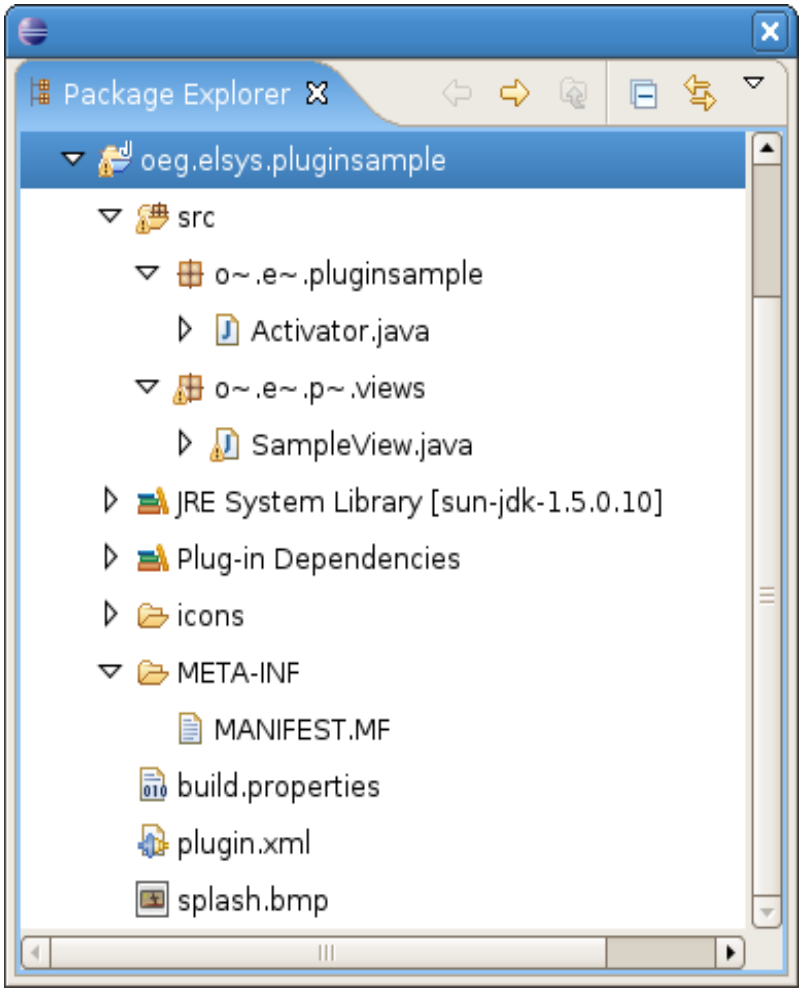
Категориите имат

- име – name
- идентификатор - id

k.mitov@sap.com

Plugin Class

- Всеки plugin може да има клас, който програмно да го представлява – т.е. от други класове да може да се достъпи информация за дадения plugin.
- Класът не е задължителен, но ако е зададен е първият използван при стартиране на дадения plugin и е последният достъпван при неговото спиране.
- За дадения пример това е класът Activator. Удобно е всеки plugin, който предоставя графичен компонен – изглед, редактор, меню и т.н. да наследява от AbstractUIPlugin. Когато това не е нужно може да се наследи класът Plugin.



k.mitov@sap.com

Plugin class - instance

```
public class Activator extends AbstractUIPlugin {
   public static final String PLUGIN_ID =
"org.elsys.pluginsample";
   private static Activator plugin;

   /* code missed ... */

   public static Activator getDefault() {
      return plugin;
   }
}
```

Полето `plugin` и методът `getDefault()` са статични. Това означава, че във всеки един момент може да съществува само една инстанция на Activator и тази инстанция може да бъде достъпвана само през `getDefault()`.

k.mitov@sap.com

Plugin class - instance

```
public class Activator extends AbstractUIPlugin {
    /* code missed ...*/
    public void start(BundleContext context) throws
Exception {
        super.start(context);
        plugin = this;
    }
    public void stop(BundleContext context) throws
Exception {
        plugin = null;
        super.stop(context);
}}
```

Методите `start()` и `stop()` са методи на интерфейса `org.osgi.framework.BundleActivator`.
start() се изиква при стартиране на дадения plugin. stop() се извиква при неговото спиране.
Методите може да се извикат произволен брой пъти. Трябва да са изключително бързи - без никакви ненужни инициализации.

k.mitov@sap.com

SampleView class

Функционалността на изградения изглед е капсулирана в класът SampleView. В класът имаме поле от тип TableViewer както и класове имплементиращи IStructuredContentProvider и ItableLabelProvider.

```
public class SampleView extends ViewPart {
   private TableViewer viewer;

   class ViewContentProvider implements
IStructuredContentProvider {
      /* code missed... */
   }

   class ViewLabelProvider extends LabelProvider
implements ITableLabelProvider {
      /* code missed... */
   }

      /* code missed... */
}
```


k.mitov@sap.com

SampleView class

`SampleView` **наследява от** `org.eclipse.ui.part.ViewPart` **– базов клас за всички изгледи в Eclipse.**
Предоставя рамка, според която лесно да се изградят изгледи.
Изграждането на изглед се състои в наследяване на ViewPart и добавяне на желаната функционалност към наследника.

```
public class SampleView extends ViewPart {

    /* code missed ... */

}
```


k.mitov@sap.com

SampleView class - createPartControl()

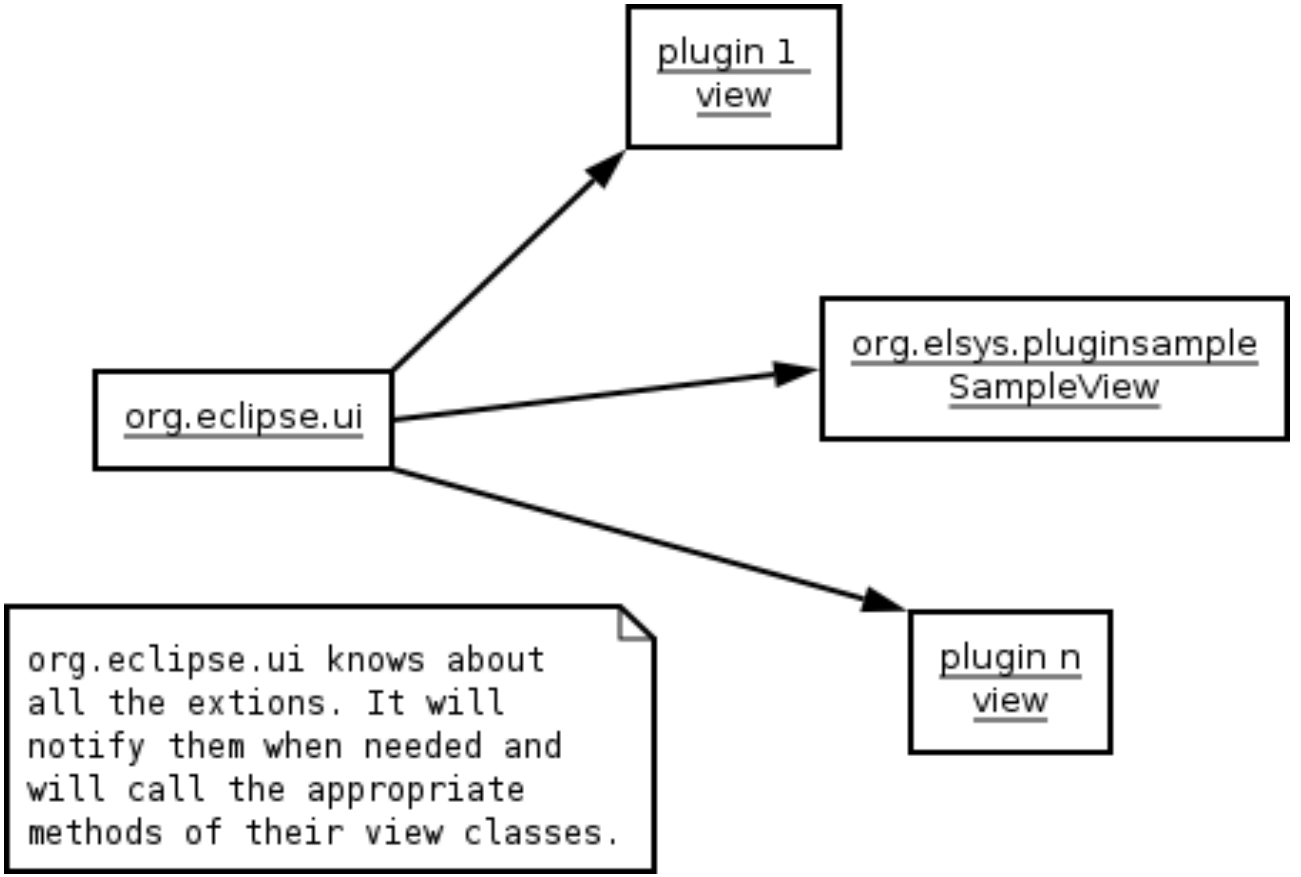
В първият момент когато се наложи инициализиране и показване на изгледа ще се извика методът `createPartControl()`.
В него създаваме и инициализираме съответния табличен viewer.

```
public class SampleView extends ViewPart {
   /* code missed ... */

   public void createPartControl(Composite parent) {
      viewer = new TableViewer(parent, SWT.MULTI |
SWT.H_SCROLL | SWT.V_SCROLL);
      viewer.setContentProvider(new
ViewContentProvider());
      viewer.setLabelProvider(new ViewLabelProvider());
      viewer.setInput(getViewSite());
   }

   /* code missed ... */
}
```


k.mitov@sap.com

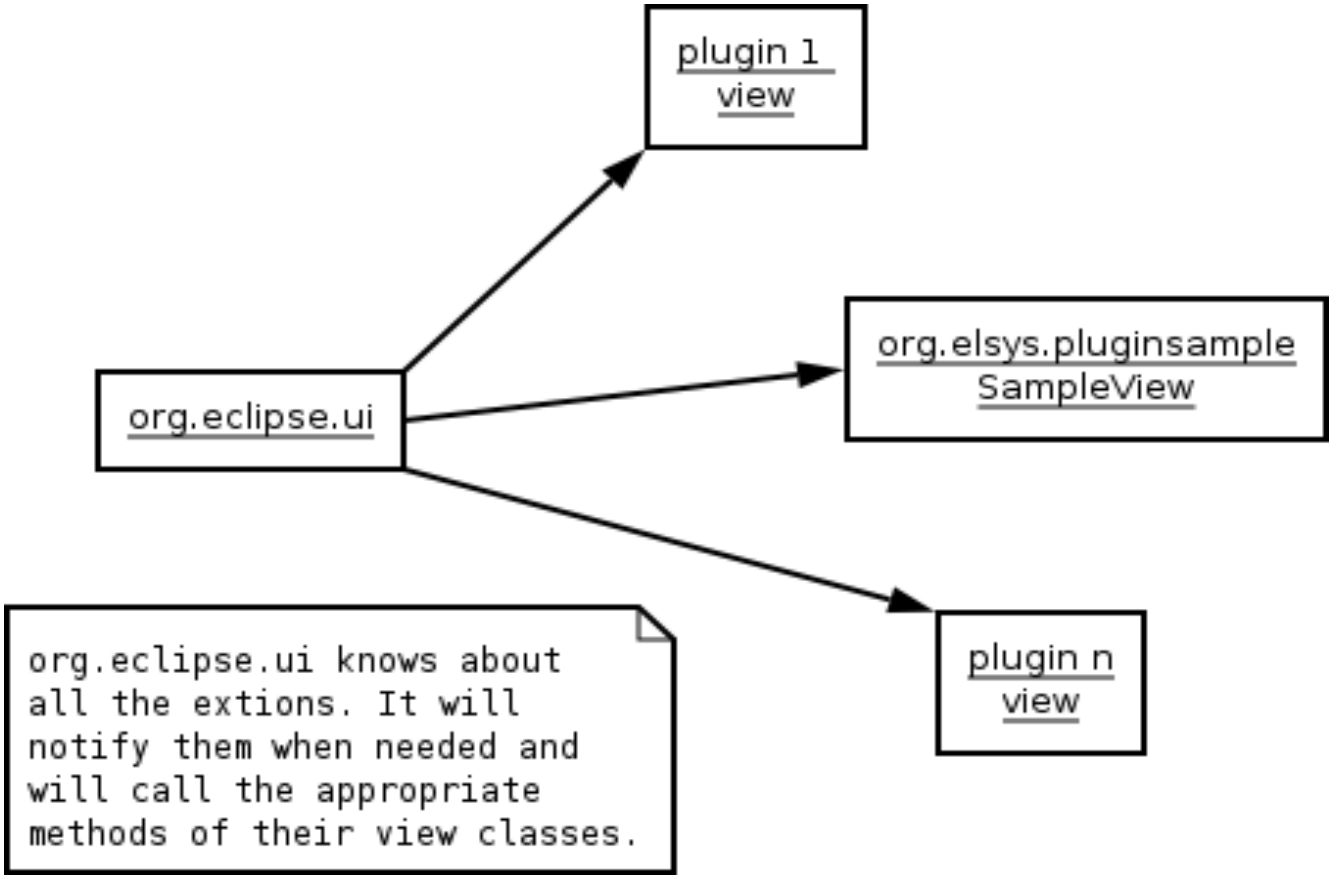
Extensions

В разработения пример създадохме разширение на `org.eclipse.ui.views`.
`org.eclipse.ui` и plugin, който следи за това какви изгледи потребителят иска да покаже и кога иска да ги покаже - т.е. той предоставя функционалност, която `org.elsys.pluginsample` допълва.



k.mitov@sap.com

Extensions



След регистрирането на нашият изглед `org.eclipse.ui` се задължава да извиква методите на нашият клас в подходящия момент. Затова той изисква всички класове изгледи да имплементират общ интерфейс - `org.eclipse.ui.IViewPart`

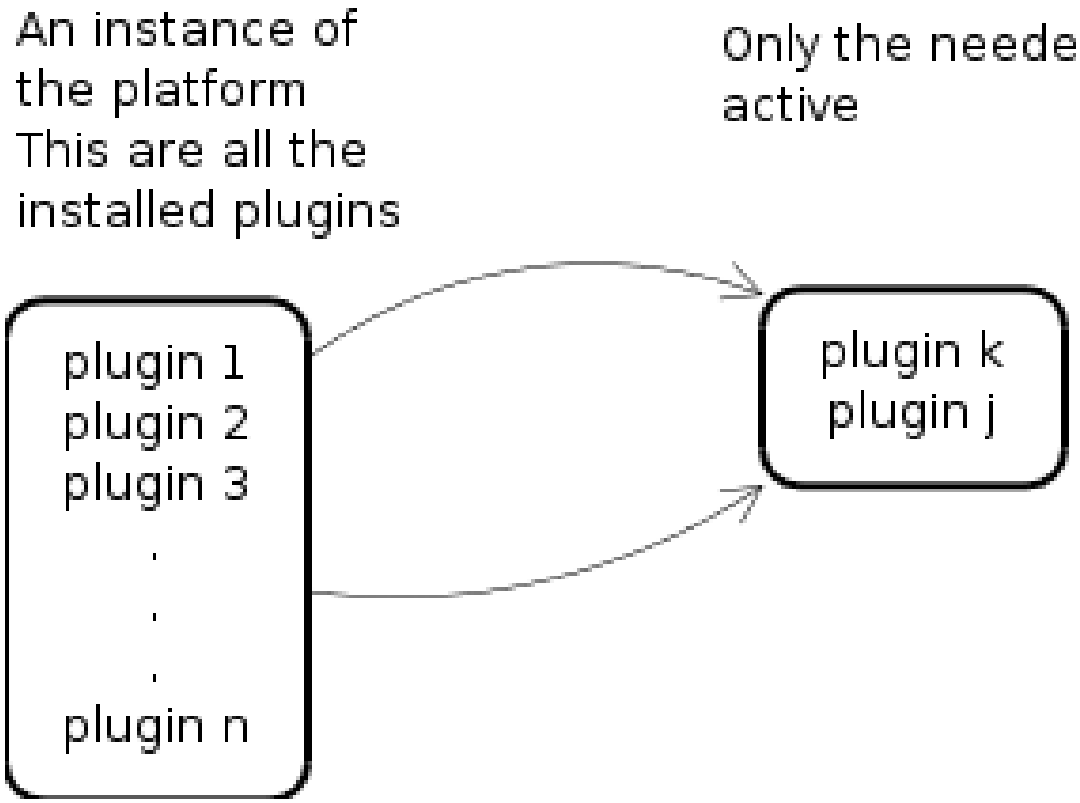
k.mitov@sap.com

Предназначение на манифестните файлове

- Всичко в Eclipse е plugin. Всяка една функционалност е обусловена като plugin.
- Инстанция на Eclipse платформата може да съдържа неограничено количество plugin-и.
- За да се стартира един plugin са необходими определени ресурси – процесорно време и памет. Стартирането на всички plugin-и при стартиране на платформата не е задължително и дори е задължително да не се прави. Стартират се само plugin-ите, от които потребителят се нуждае.
- В процеса на работа може да бъдат стартирани и други plugin-и, но само когато се наложи.

k.mitov@sap.com

Предназначение на манифестните файлове

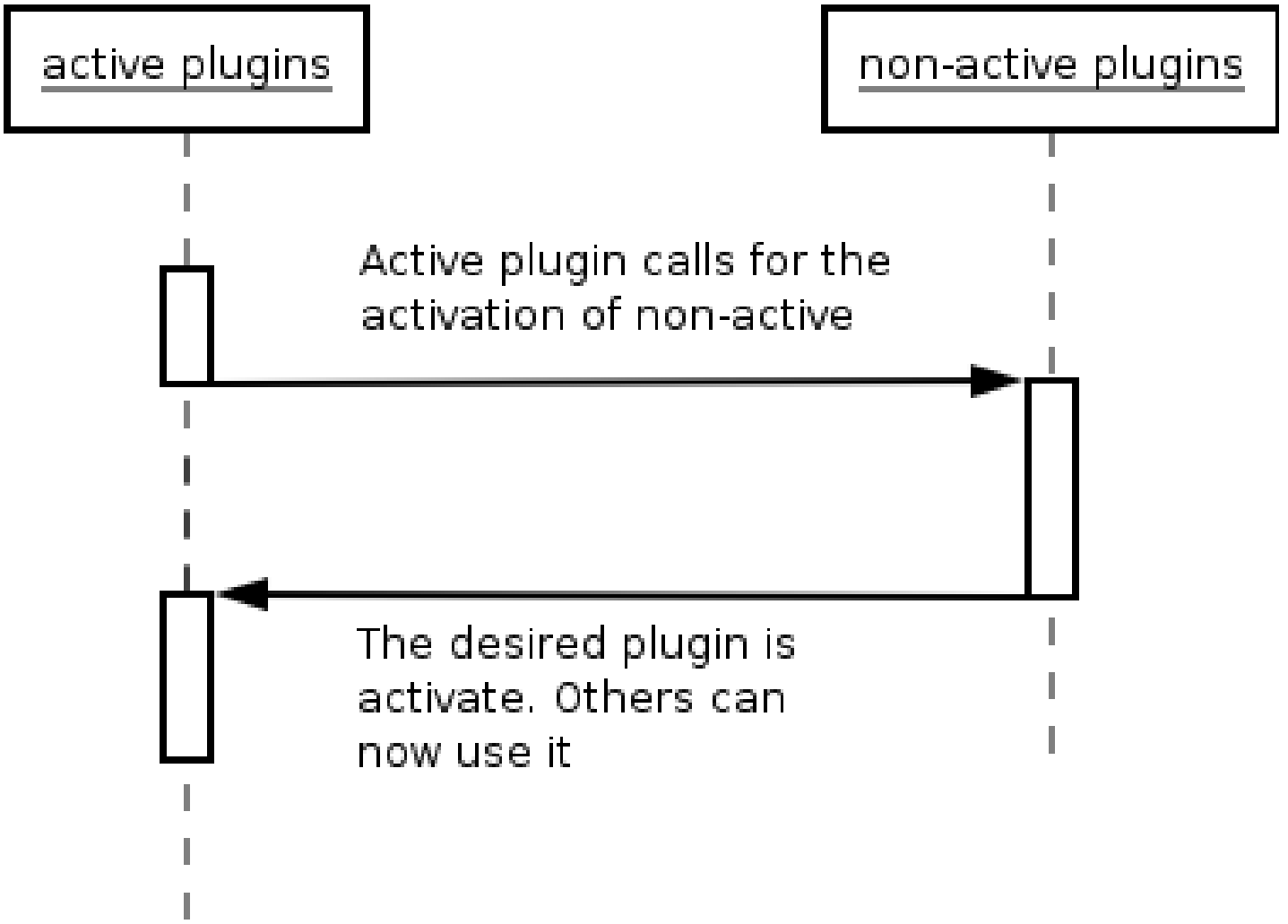


Системата може да съдържа неограничено количество plugin-и. Ограничено количество от тях са стартирани – само необходимите

k.mitov@sap.com

Предназначение на манифестните файлове

В произволен момент от време активен plugin може да изиска активирането на неактивен.



k.mitov@sap.com

Lazy Loading (Lazy initialisation)

Основно правило при изграждането на eclipse plugin-и е функционалността предоставена от тях да се инициализира във възможно най-късен момент.

Примерно plugin предоставящ изглед не се активира (не заема системни ресурси) докато потребителят не пожелае да отвори предоставения изглед.

Поставят се въпросите

- **кой активира plugin-ите?**
- **как разбира кога да ги активира?**

k.mitov@sap.com

Lazy Loading (Lazy initialisation)

Кой активира plugin-ите?

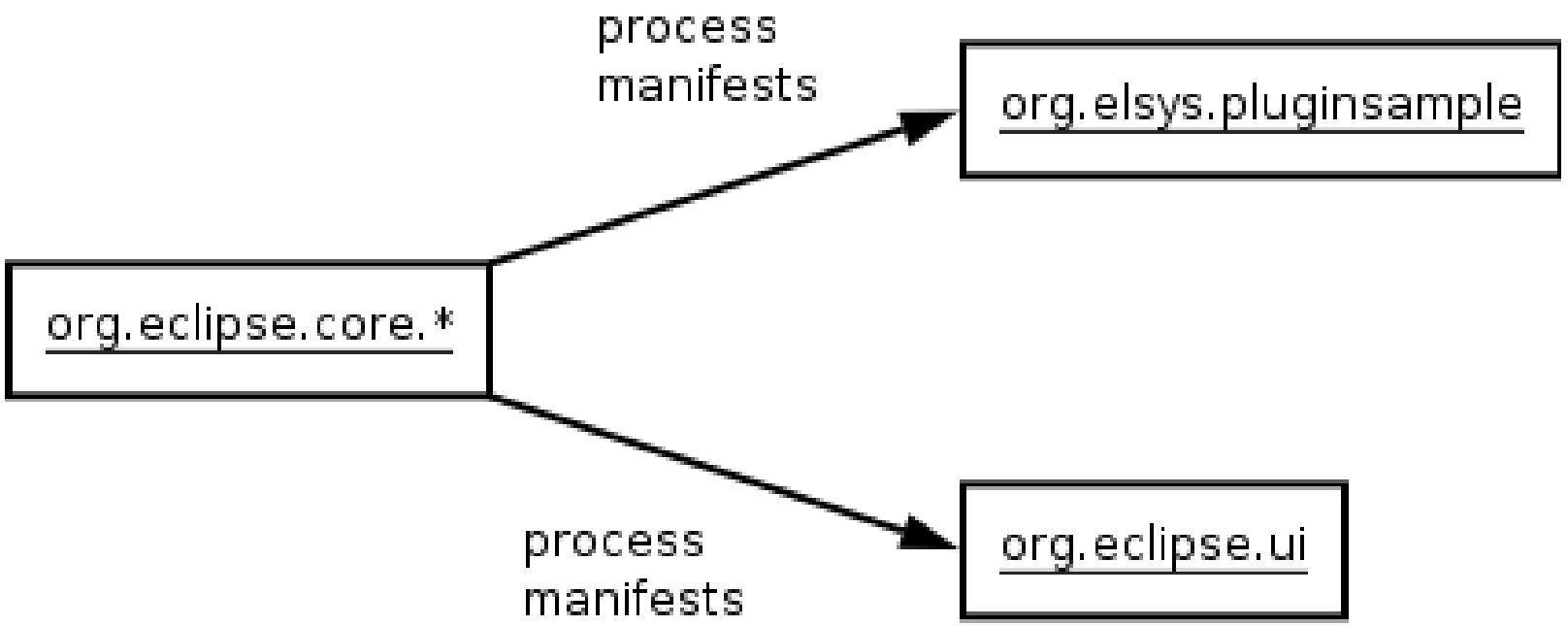
plugin-ите с имена org.eclipse.equinox.* и org.eclipse.core.*. (активирането е сложен процес и изисква взаимодействието на различни plugin-и)

Как разбират кога?

всеки plugin има един или два манифестни файла – MENIFEST.MF и plugin.xml (незадължителен). Тези два файла описват дадения plugin – име, версия, разширения (extensions), extension points и др. Plugin-ите org.eclipse.equinox.* и org.eclipse.core.* се стартират преди всички останали, обработват манифестните файлове на другите plugin-и, и преценяват, кой трябва да се активира.

k.mitov@sap.com

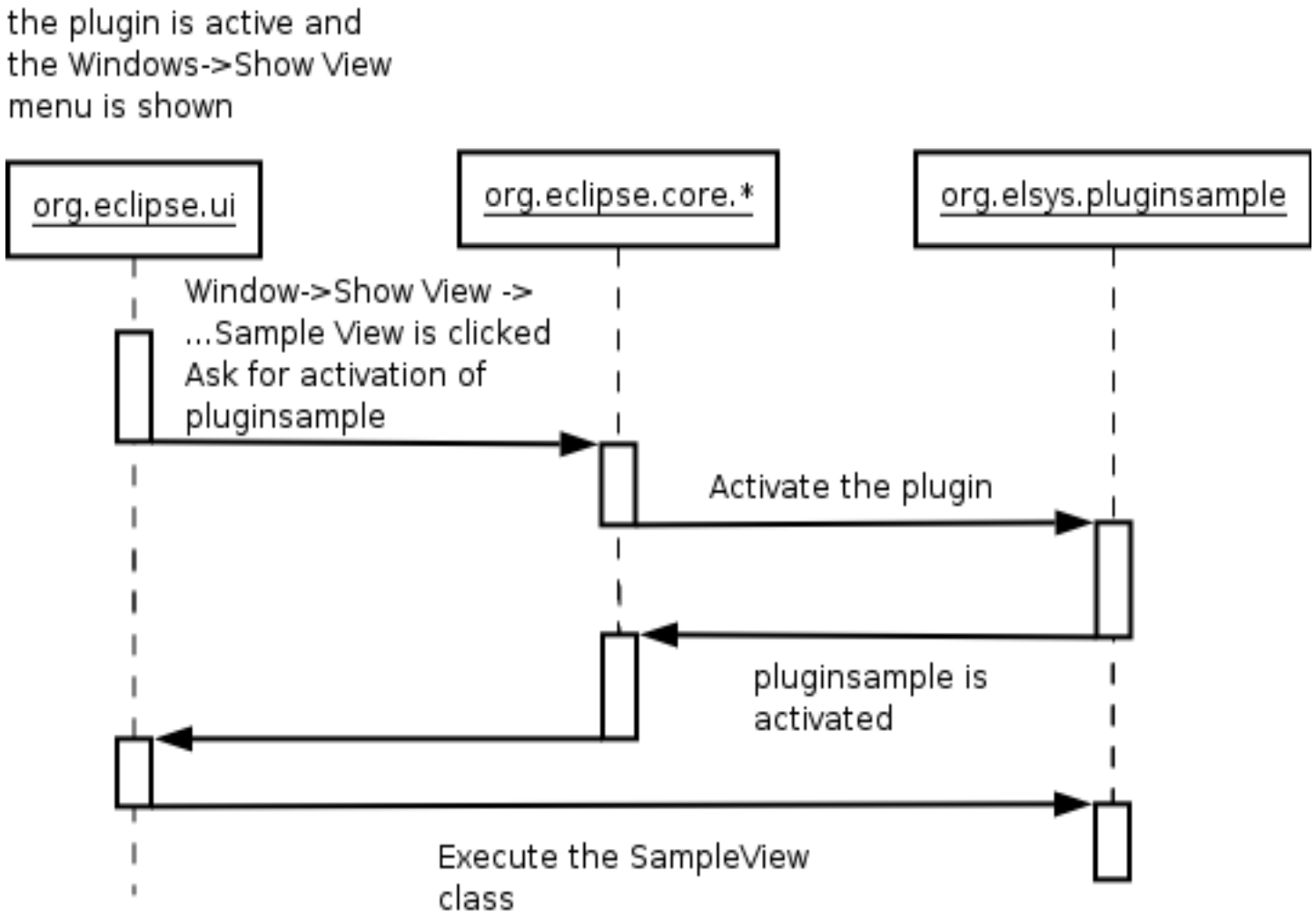
Lazy Loading (Lazy initialisation)



k.mitov@sap.com

Lazy Loading (Lazy initialisation)

На следващата фигура е показано опростено активиране на даден plugin и изпълнение на негов клас.



k.mitov@sap.com

Lazy Loading (Lazy initialisation)

- `org.eclipse.ui` е активиран.
  - Той знае, че `org.elsys.pluginsample` предоставя разширение (extension). При натискане на Windows->Show View->...->Sample View се остановява, че трябва да се изпълни разширението на `org.elsys.pluginsample`. Ако `org.eclipse.pluginsample` не е активиран то `org.eclipse.core.*` го активира. След това org.eclise.ui предава изпълнението на класа `SampleView`.
  - В следващ момент `org.eclipse.ui` може да изпълни разширение от друг plugin, но това няма да деактивира `org.elsys.pluginsample.`

k.mitov@sap.com

Actions (Действия)

В Eclipse съществува понятието action – действие.

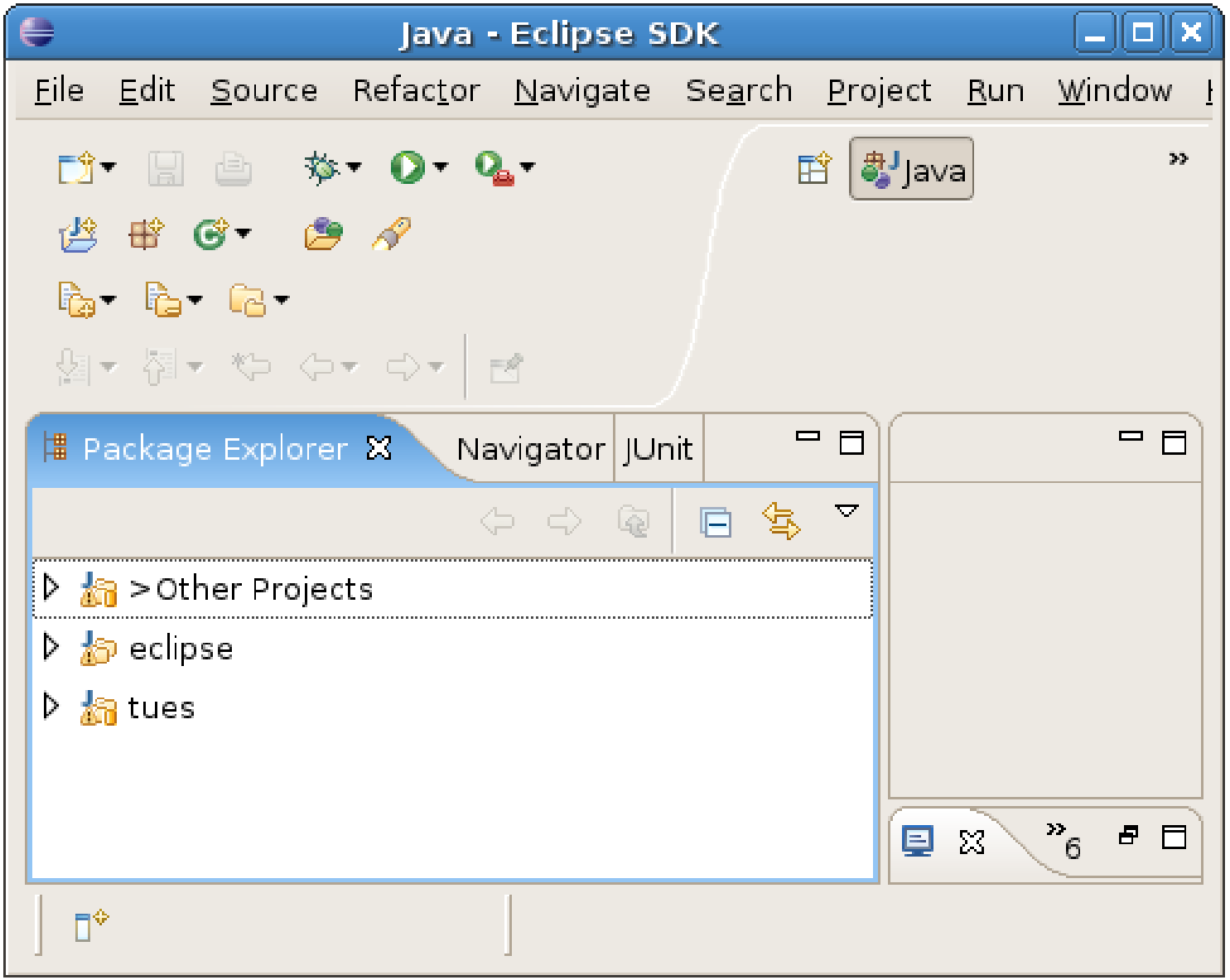
Действията (actions) позволяват да се дефинират команди, които потребителят може да изпълни без да се интересува от използвания потребителски интерфейс.

Позволява да променяме визуалното представяне на дадено действие без да променяме кода, който реално изпълнява действието.

Действията се показват на няколко места в Eclipse IDE – менюта, ленти с инструменти (toolbars), контекстни менюта.

k.mitov@sap.com

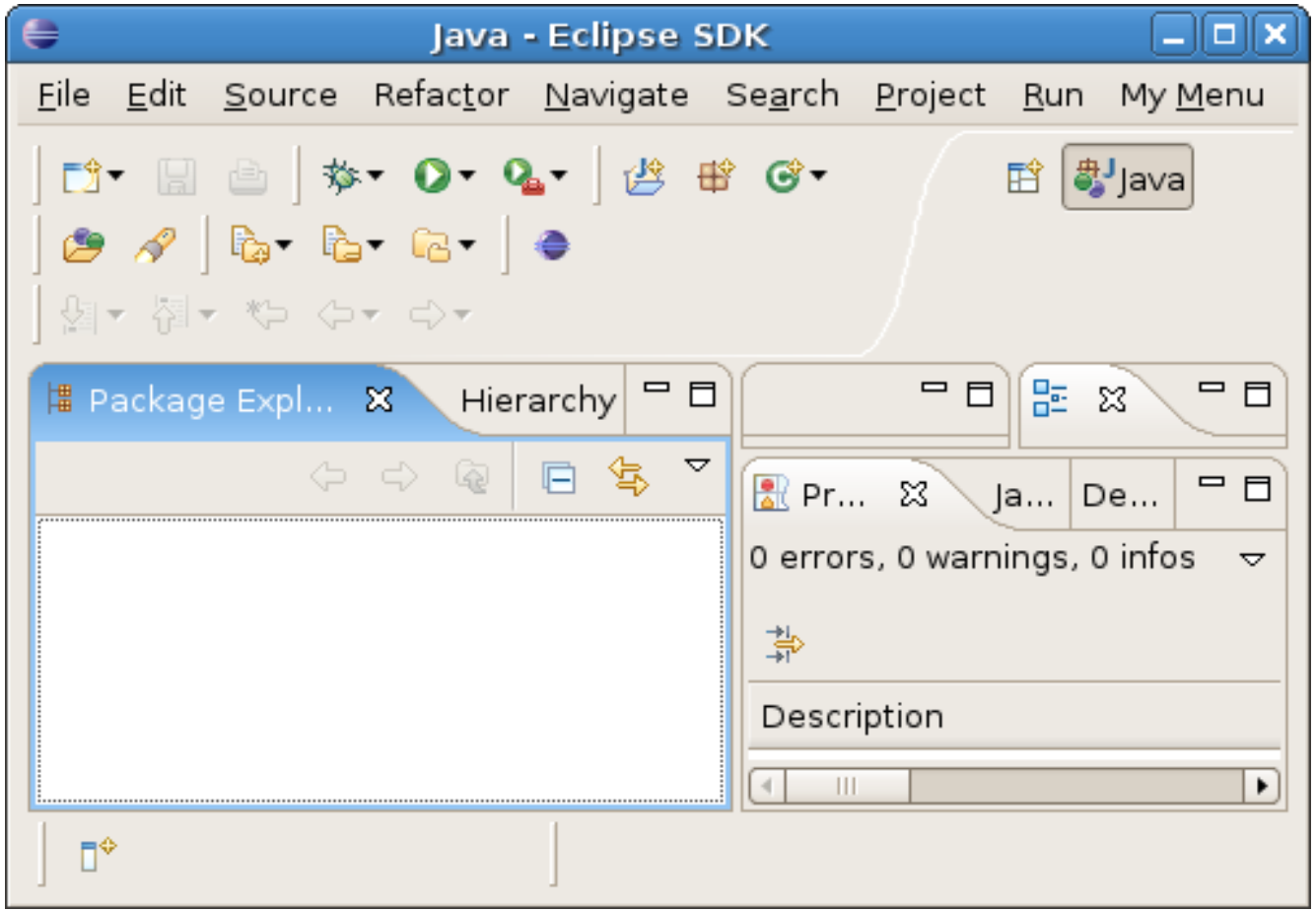
Actions (Действия)



k.mitov@sap.com

Actions (Действия)

Пример:
Целта на примера е да се изгради действие „action“, което да се изобразява в лентата с инструменти както и като елемент от главното меню. При изпълнение на действието трябва да се показва диалогов прозорец с текст „Hello World“



k.mitov@sap.com

Actions (Действия)

Резултатът от изпълнението на действието е следният:

k.mitov@sap.com

Actions - org.eclipse.ui.actionsSet

Целта на примера е да се изгради plugin, който да добавя съответното действие към платформата.

Платформата предоставя extension point – `"org.eclipse.ui.actionSets"`, който може да бъде разширен и действието добавено.

Този extension point предоставя възможност за създаване на елемент в менюто – menu и действие – action.

Действието може да се разположи в лентата с инструменти – toolbar, главното меню или в едно или повече контексни менюта.

k.mitov@sap.com

Actions - Plugin.xml

```
<extension
         point="org.eclipse.ui.actionSets">
      <actionSet
            label="My Action Set"
            visible="true"
            id="org.elsys.actions.actionSet">
         <menu
               label="My &amp;amp;Menu"
               id="myMenu">
            <separator
                  name="myGroup">
            </separator>
         </menu>
         <action
               label="&amp;amp;My Action"
               icon="icons/sample.gif"
               class="org.elsys.actions.actions.MyAction"
               tooltip="Hello world"
               menubarPath="myMenu/myGroup"
               toolbarPath="myGroup"
               id="org.elsys.actions.actions.MyAction">
         </action>
      </actionSet>
   </extension>
```


k.mitov@sap.com

Actions - org.eclipse.ui.actionsSet

От клиентът се изисква да предостави клас - `org.elsys.actions.actions.MyAction`, който да имплементира интерфейса `org.eclipse.ui.IWorkbenchWindowActionDelegate`

```
<action
    /*  ... */
    class="org.elsys.actions.actions.MyAction"
    /*  ... */
</action>
```

Класът на клиента няма да бъде зареден и няма да заема системни ресурси докато потребителят на пожелае изпълнение на съответното действие. - Lazy Initialization

`IWorkbenchWindowActionDelegate` има метод `run`. Този метод се извиква тогава когато потребителят натисне съответния визуален компонент.

k.mitov@sap.com

Actions - MyAction

Класът съдържащ имплементацията на реалното действие.

```
public class MyAction implements
IWorkbenchWindowActionDelegate {
   private IWorkbenchWindow window;
   public MyAction() {}

   public void run(IAction action) {
      MessageDialog.openInformation(window.getShell(),
"Actions Plug-in", "Hello, Eclipse world");
   }
   public void selectionChanged(IAction action, ISelection
selection) {}

   public void dispose() { }

   public void init(IWorkbenchWindow window) {
      this.window = window;
   }
}
```


k.mitov@sap.com

Actions (Действия)

Резултатът от изпълнението на действието е следният:

k.mitov@sap.com

Перспективи

Информацията в Eclipse се изобразява изгледи и редактори. Потребителят може да взаимодейства със средата посредством различни менюто, действия, бързи клавиши.

Платформата е отворена за разширение и поради тази причина броят на изгледите, редакторите и тн. е неограничен. В зависимост от действията извършвани от потребителя той се нуждае от ограничен брой изгледи и редактори. Нуждае се от групирането им.

Това групиране се извършва с помощта на песпективите. Всяка една перспектива съдържа в себеси определени изгледи и определя тяхното разположение.

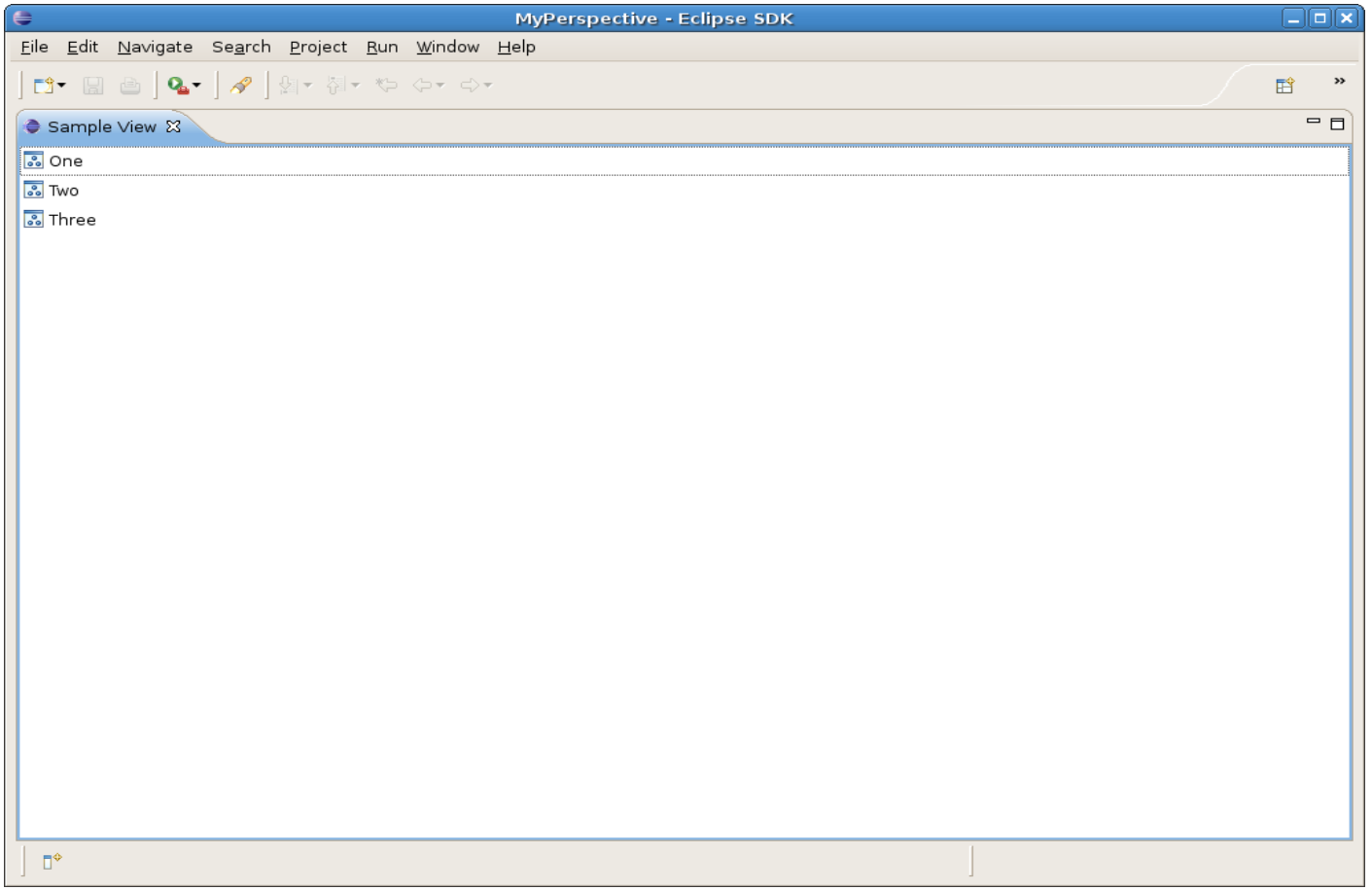
Всички налични перспективи може да бъдат видяни чрез Window-> Open Perspective -> Other...

(Java, CVS, Resource, Debug...)

k.mitov@sap.com

Перспективи

Пример:
Да се създаде перспектива изобразяваща единствен изглед.



k.mitov@sap.com

Перспективи – Plugin.xml

Съществува extension point – `org.eclipse.ui.perspective`.

```
<extension
         point="org.eclipse.ui.perspectives">
      <perspective
            class="org.elsys.pluginsample.PerspectiveFactory1"
            id="org.elsys.pluginsample.perspective1"
            name="MyPerspective">
      </perspective>
   </extension>
```

Разширеняването на този extension point ще доведе до създаването на нова перспектива. Всяка перспектива има:

- id- уникален идентификатор
- name – име подходящо за потребителя
- class – клас управляващ работата на перспективата

k.mitov@sap.com

Перспективи - IPerspectiveFactory

Класът зададен чрез атрибута class трябва да имплементира `org.eclipse.ui.IPerspectiveFactory`. Интерфейсът има единствен метод – `createInitialLayout()`.

```
public  class PerspectiveFactory1 implements
IPerspectiveFactory {

   public void createInitialLayout(IPageLayout layout) {
      // Get the editor area.
      String editorArea = layout.getEditorArea();
      // Put the Sample View
 layout.addView("org.elsys.pluginsample.views.SampleView",
IPageLayout.LEFT, 1f,   editorArea);
      layout.setEditorAreaVisible(false);
   }
}
```


k.mitov@sap.com

Перспективи - IPerspectiveFactory

Всяка перспектива съдържа множество изгледи и редактори. Областа за отваряне на редакторите е единствина. Изгледите се нареждат около тази област

```
String editorArea = layout.getEditorArea();
```

Методът `addView` добавя изгледа с посочения идентификато към перспективата. Изгледът ще се намира от ляво на областа за редактори. Изгледът ще заема цялото пространство – 1f.

```
layout.addView("org.elsys.pluginsample.views.SampleView"
, IPageLayout.LEFT, 1f,editorArea);
```

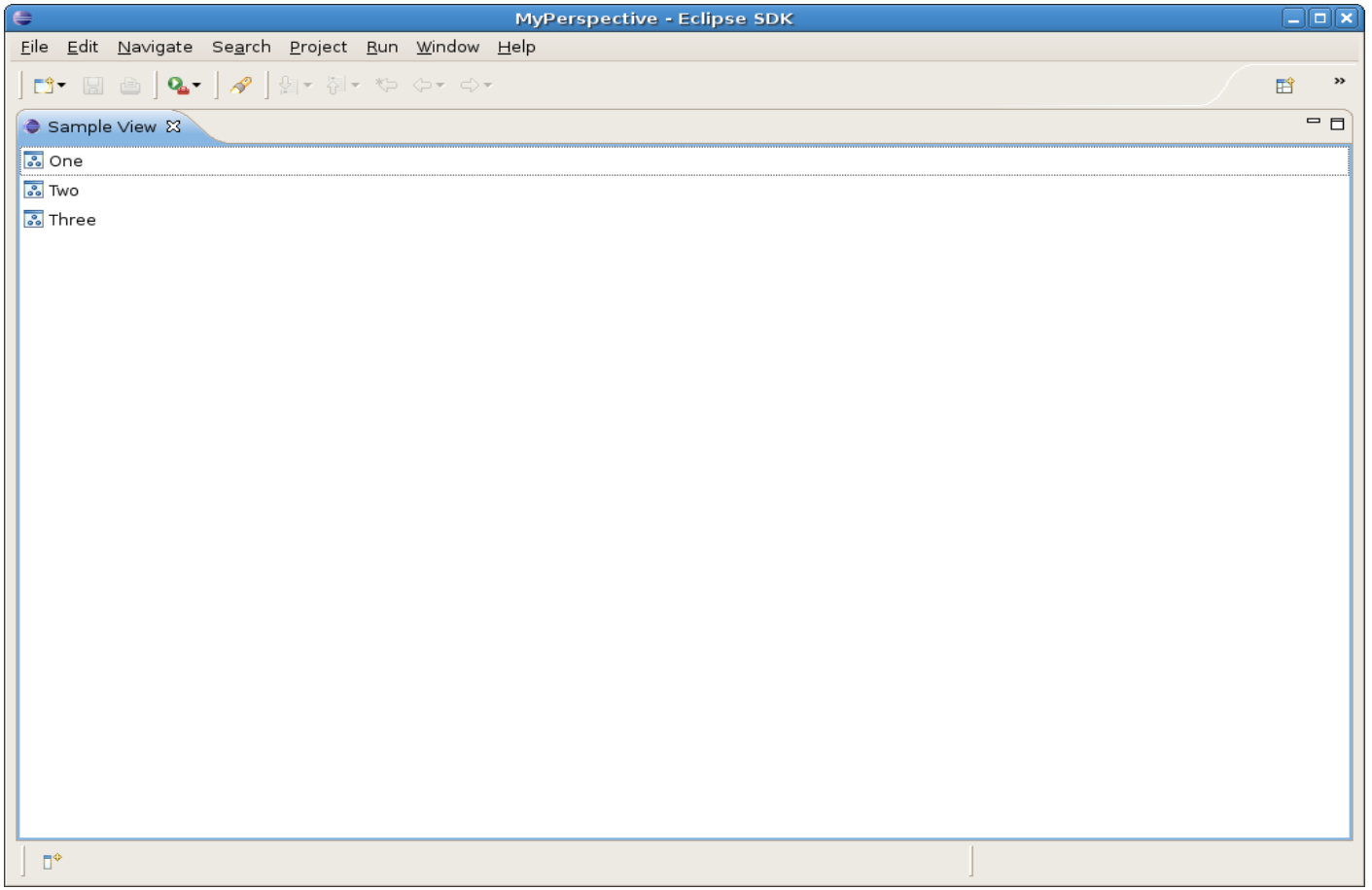
За да може изгледът да заеме целия екран областа за редакторите не трябва да е видима.

```
layout.setEditorAreaVisible(false);
```


k.mitov@sap.com

Перспективи

Перспективата ще изглежда по следния начин:



k.mitov@sap.com

Обобщение

- Представа за основните Eclipse проекти – Equinox, Platform, JDT, PDE
- Пример за изграждане на plugin проект.
- Манифестни файлове – MANIFEST.MF и plugin.xml
- Класът активатор – методите start() и stop() трябва да са бързи и да не извършват излишна инициализация.
- Разширения (extensions) – org.eclipse.ui.views е разширението за дефиниране на изгледи
- Късна инициализация (Lazy Loading) – зареждането на даден plugin трябва да се извършва само когато това е необходимо.

k.mitov@sap.com