Procesadores de Lenguajes 2

La plataforma Eclipse

Curso 2013-2014

Iván Ruiz Rube
Departamento de Ingeniería Informática
Escuela Superior de Ingeniería
Universidad de Cádiz

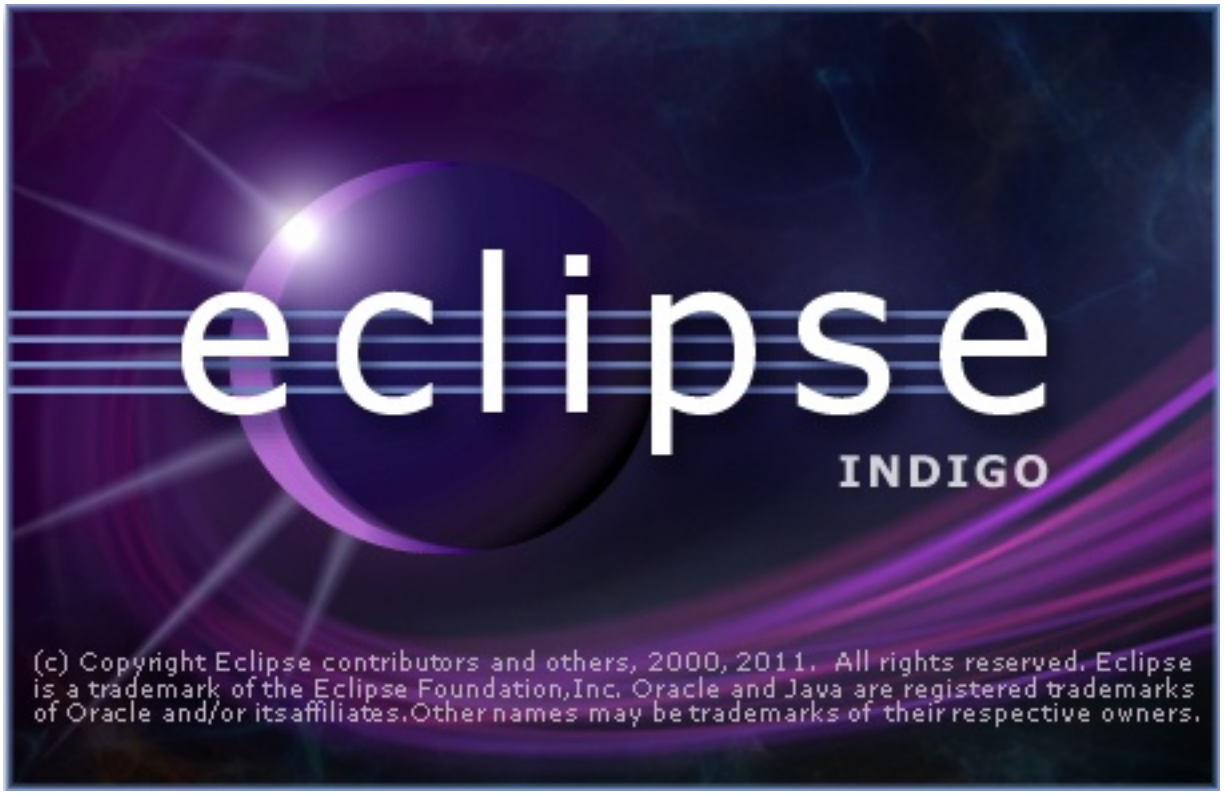
Contenidos

- Introducción
- Proyectos
- Componentes
- Arquitectura
- Desarrollo de un proyecto Java
- Desarrollo de un plug-in
- Desarrollo de una aplicación RCP

LA PLATAFORMA ECLIPSE

INTRODUCCIÓN

No sólo un IDE…



Instalación

- Se requiere la JDK 1.6 o superior
- Acceder a http://www.eclipse.org/
- Entrar en *Downloads*
- Clic en *Eclipse Modeling Tools*
- Descargar versión *Win/Mac/Linux, 32/64*
- Ajustar parámetros de memoria *XX:MaxPermSize* y *Xmx* en *eclipse.ini (opcional)*
- Inicia *eclipse*

Eclipse IDE

- Eclipse es una plataforma de desarrollo integrada de código abierto multiplataforma.
- Eclipse fue desarrollado originalmente por IBM. Ahora es mantenido por la Fundación Eclipse.
- Participan empresas como HP, Intel, Motorola, etc.
- Dispone de una gran comunidad de usuarios.

Eclipse IDE

- Es un entorno de desarrollo de software escrito en su mayor parte en Java.
- Ideal para el desarrollo de aplicaciones web, móviles o de escritorio con Java.
- Soporte al refactoring y análisis de código.
- Integración con servidores web (Tomcat, Glassfish)
- La propia plataforma sirve para construir **aplicaciones de cliente enriquecidas**.

LA PLATAFORMA ECLIPSE

PROYECTOS

Runtimes Eclipse

- Eclipse Communication Framework
- EclipseLink Project
- Equinox
- embedded Rich Client Platform
- Gemini - Enterprise Modules Project
- Jetty - Servlet Engine and Http Server
- Rich Ajax Platform
- Riena Project
- SMILA
- Virgo

Eclipse Web Tools Platform Project

- WTP Common Tools
- Dali Java Persistence Tools
- WTP EJB Tools
- WTP Incubator
- WTP Java EE Tools
- JavaScript Development Tools
- JavaServer Faces
- Enterprise Tools for the OSGi Service Platform
- Pave
- Webtools Releng
- Server Tools
- WTP Source Editing
- Web Services Tools

SOA Platform Project

- BPEL Designer
- BPMN2 Modeler Project
- BPMN modeler
- eBAM
- eBPM
- Java Workflow Tooling
- Mangrove - SOA Modeling Framework
- SCA Tools
- Stardust
- Swordfish

Tools Project

- AJDT - AspectJ Development Tools Project
- AspectJ
- Ajax Tools Framework (ATF)
- C/C++ Development Tooling (CDT)
- WindowBuilder
- Graphical Editing Framework (GEF)
- Memory Analyzer
- Eclipse Orbit Project
- PHP Development Tools
- Parallel Tools Platform (PTP)
- Target Management

Technology Project

- Dynamic Languages Toolkit
- Eclipse Git Team Provider
- Eclipse Process Framework Project
- Eclipse Tools for Microsoft Silverlight
- The Eclipse Examples Project
- Subversive - SVN Team Provider
- Graphical Editing Framework 3D
- Hudson
- Eclipse IAM (Integration for Apache Maven)
- IDE for Education
- Linux Tools
- Open Financial Market Platform
- Open Healthcare Framework

Otros proyectos

- Eclipse Project
- Data Tools Platform
- Business Intelligence and Reporting Tools (BIRT)
- Mylyn
- Test and Performance Tools Platform Project

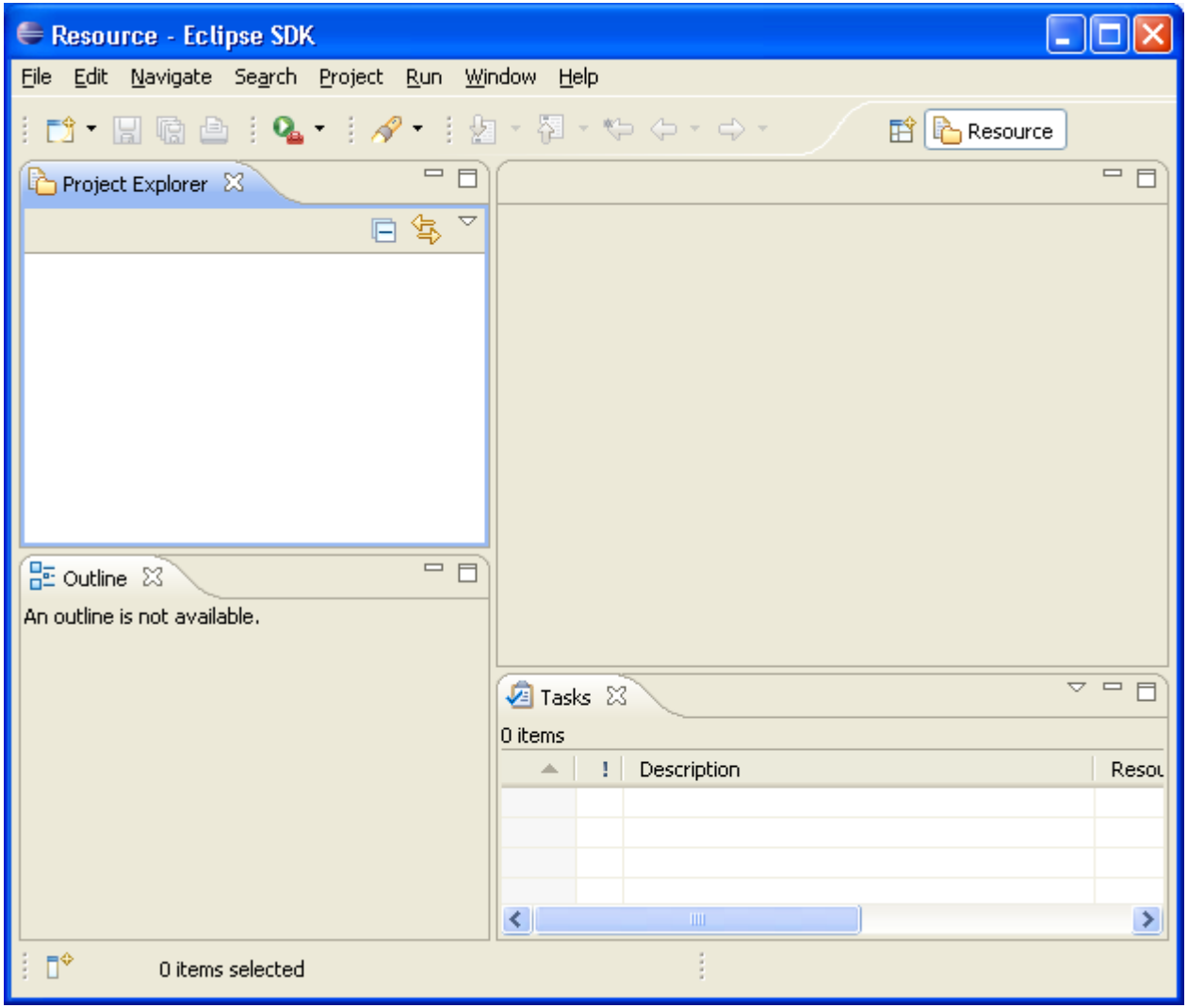
Eclipse Modeling Project (EMP)

The Eclipse Modeling Project focuses on the evolution and promotion of model-based development technologies within the Eclipse community by providing a unified set of modeling frameworks, tooling, and standards implementations.

LA PLATAFORMA ECLIPSE

COMPONENTES

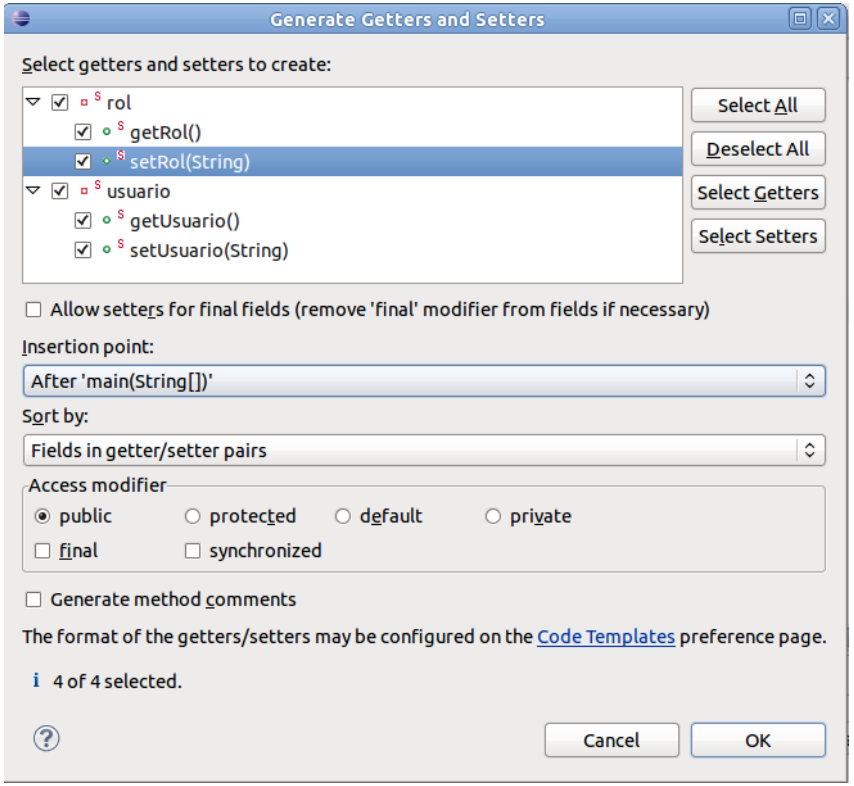
Workbench


Resource - Eclipse SDK
File Edit Navigate Search Project Run Window Help
Resource
Project Explorer
Outline
An outline is not available.
Tasks
0 items
Description
0 items selected

Workspace

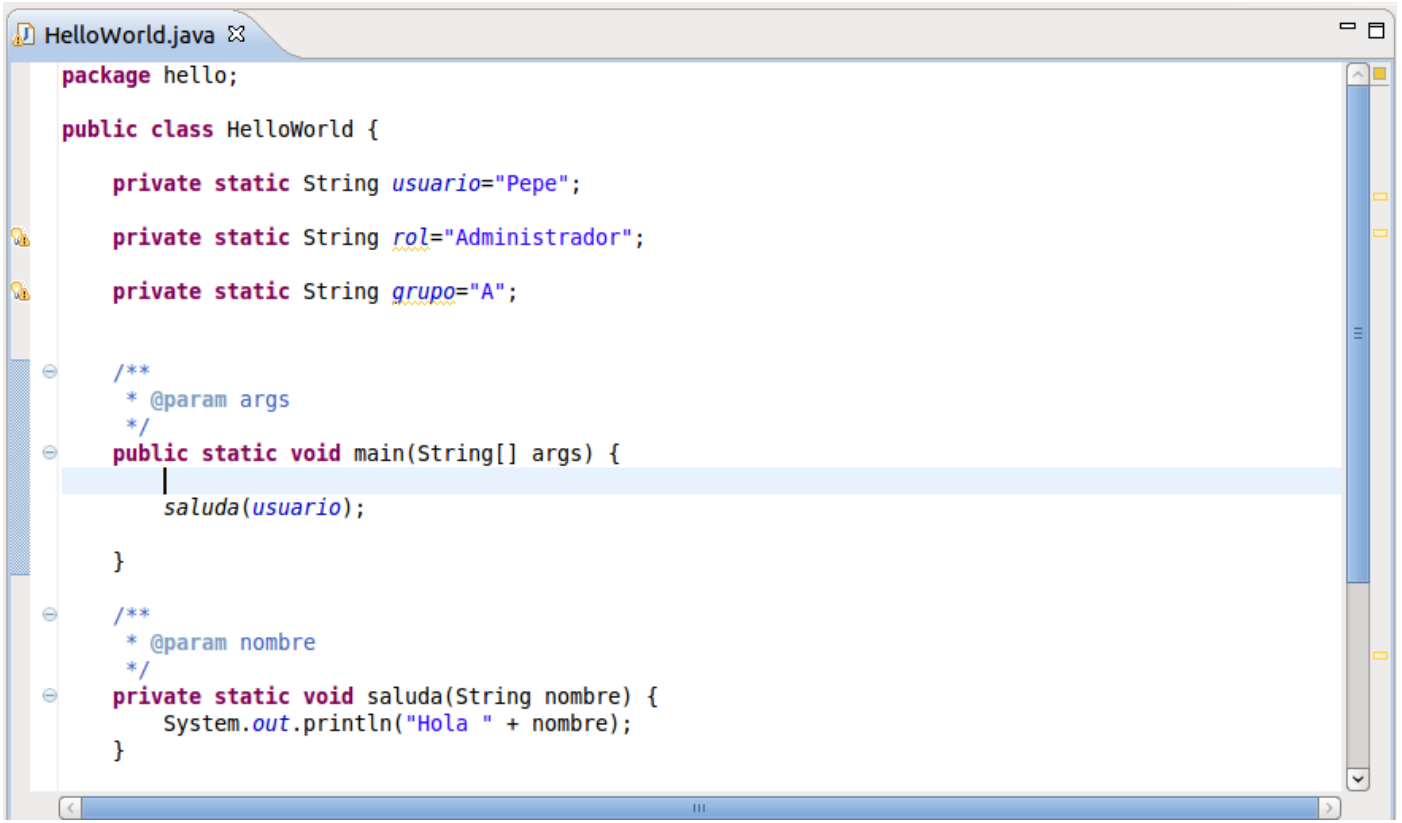
- Representa el espacio de trabajo del usuario.
- Se trata de un directorio local de la máquina donde se guardan los ficheros de trabajo.
- Utiliza metadatos (ocultos para el usuario) para llevar el histórico de modificaciones en los recursos y recordar las preferencias del usuario.
- Detecta cambios externos realizados directamente en el sistema de ficheros.

Asistentes



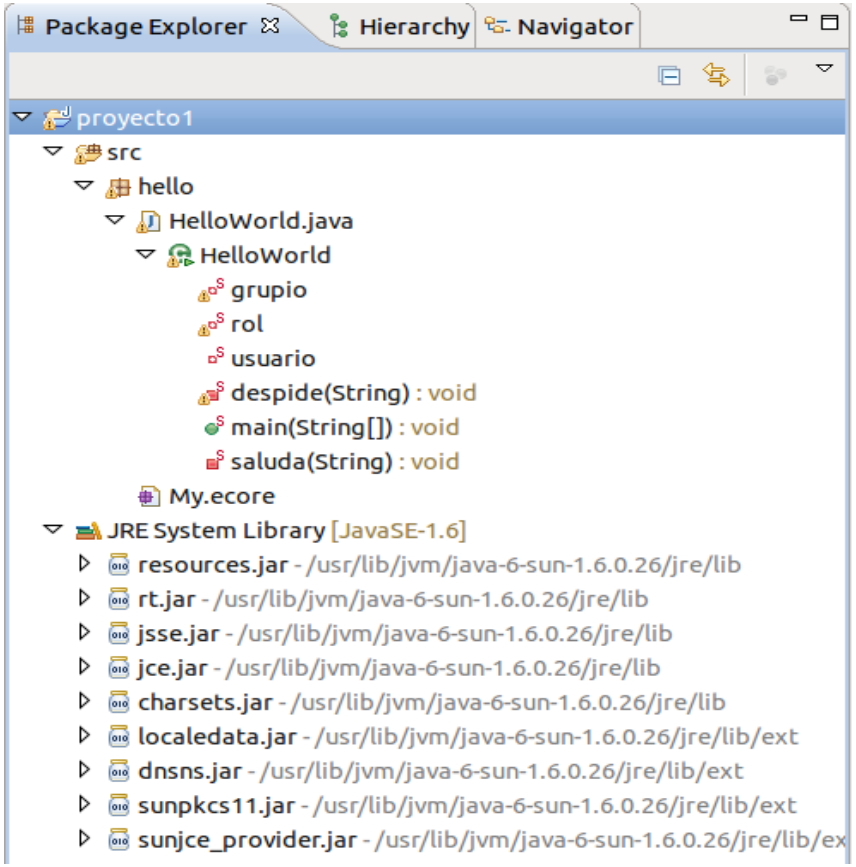
Los asistentes (wizards) guían al usuario a la hora de llevar a cabo un conjunto de tareas: crear un nuevo proyecto, hacer refactoring de código, etc.

Editores



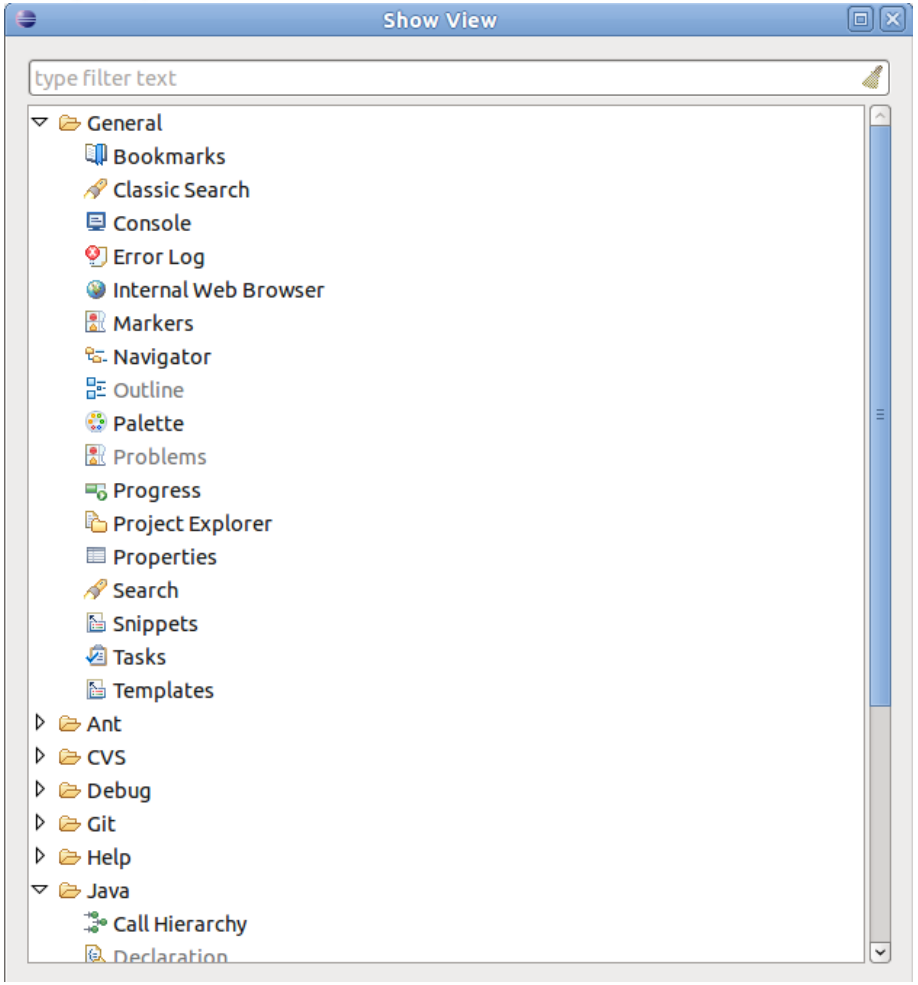
Componente visual utilizado para editar los datos de un recurso determinado. Los editores pueden ser textuales, gráficos o basados en formularios.

Vistas



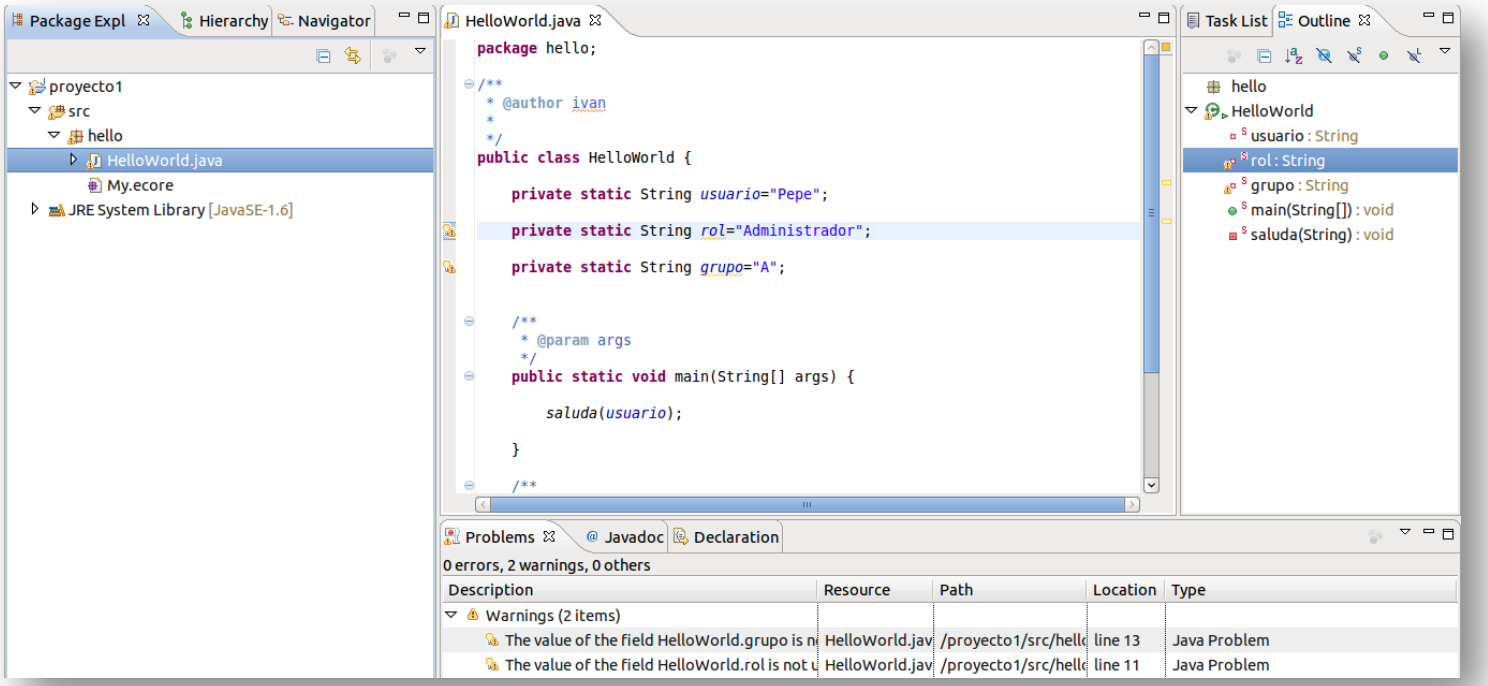
Componentes visuales que permiten navegar a través de una lista o jerarquía de elementos. También presentan propiedades específicas para para el editor activo.

Vistas (II)



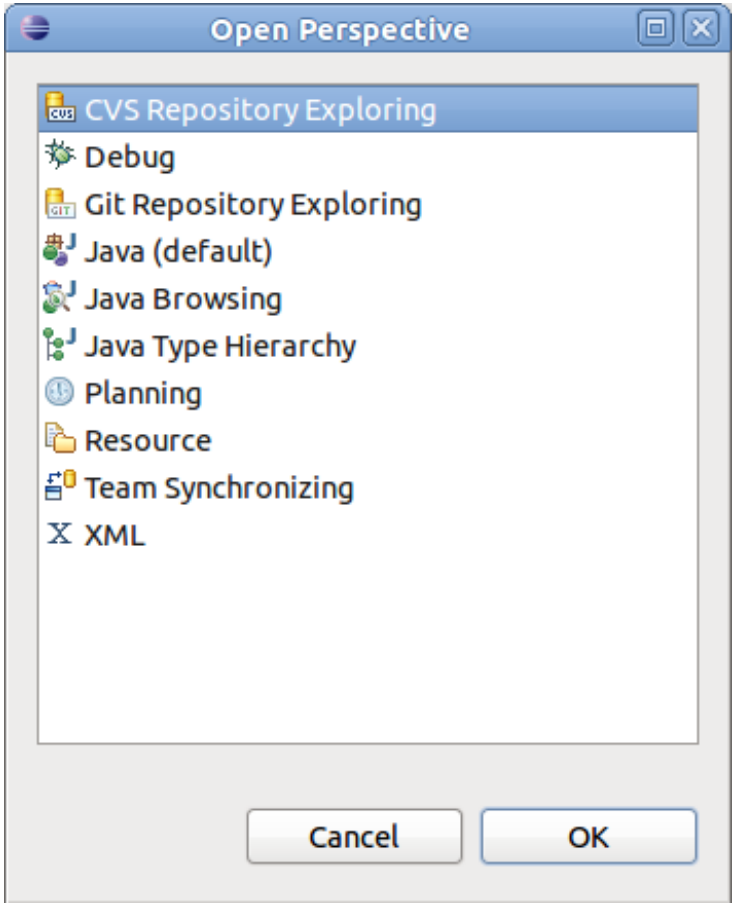
`Window → Show View`

Perspectivas (I)



Las perspectivas son una colección de vistas y editores

Perspectivas (II)

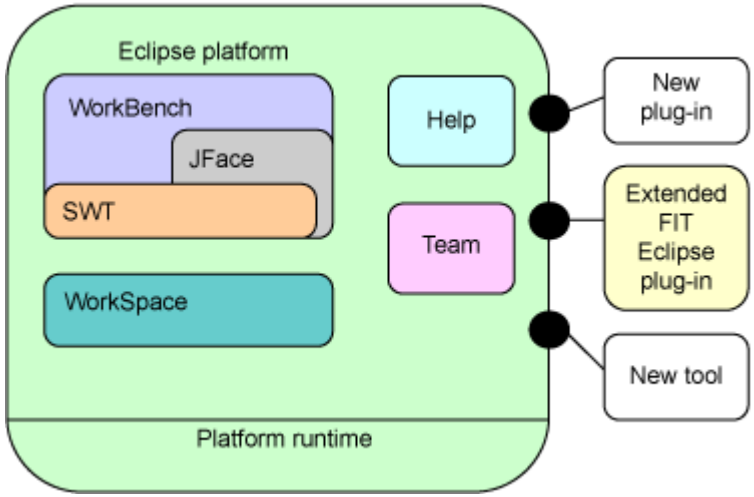


`Window → Open Perspective`

LA PLATAFORMA ECLIPSE

ARQUITECTURA

Arquitectura

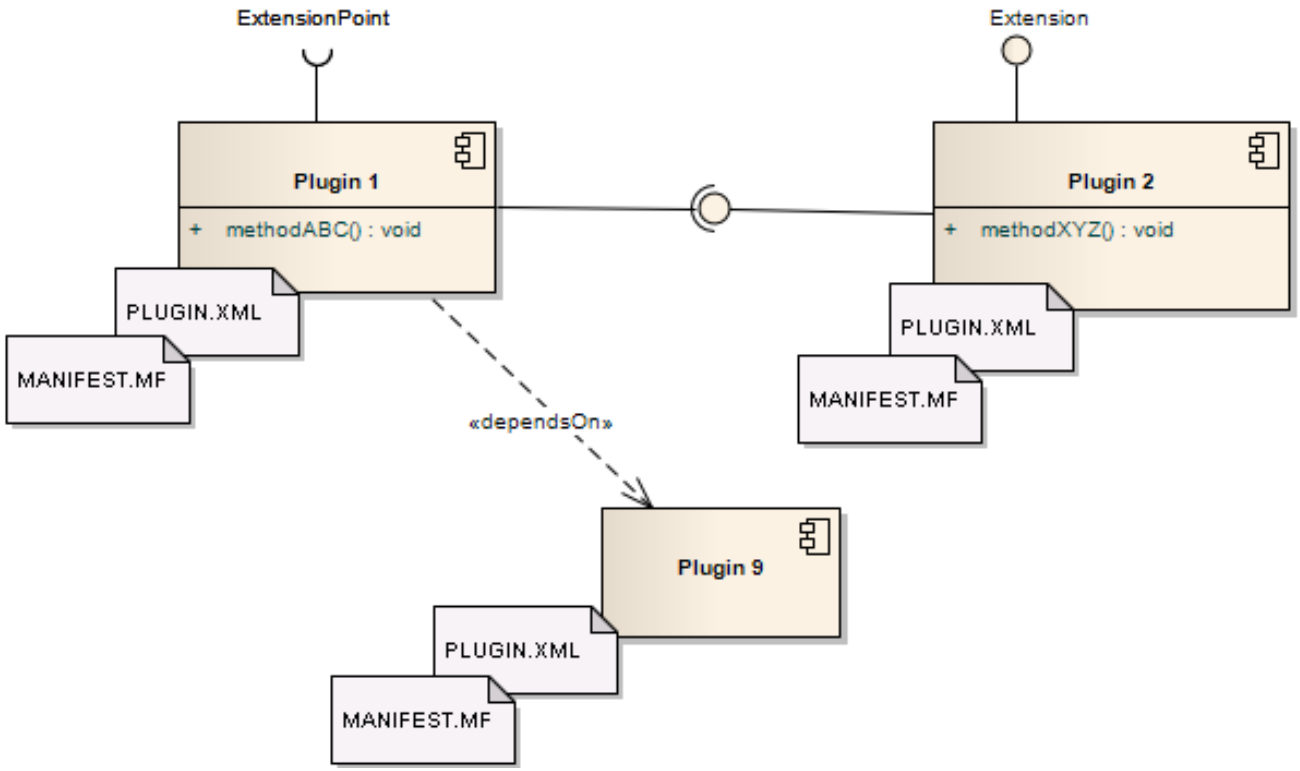


- Eclipse se compone de una base de código (kernel) y un conjunto de extensiones adicionales (plug-ins).
- La GUI de Eclipse se compone de widgets, desarrollados con SWT y utilizando un MVC con JFace.

Plug-in Development Environment (PDE)

- Eclipse permite extender la funcionalidad del IDE mediante plugins.
- Utilizando PDE podemos crear y/o editar editores, vistas, menús y asistentes del workbench de Eclipse
- Eclipse se basa en la tecnología OSGI, para proporcionar una arquitectura completamente modular.

Componentes de un Plug-in



Cada plugin define su propia API (conjunto de clases públicas), las dependencias con otros plugins, los puntos de extensión (para que otros plugins lo extiendan) y las extensiones que aportan (contribuciones a otros plugins).

Plug-ins Eclipse

- Soporte a lenguajes de programación distintos de Java: PHP, C++, Ruby, Latex, etc.
- Edición de modelos basados en lenguajes de la OMG: UML, BPMN, etc.
- Control de versiones: SVN, GIT, etc.
- Integración con sistemas de gestión de tareas: Bugzilla, Trac, JIRA, etc.
- Etc.

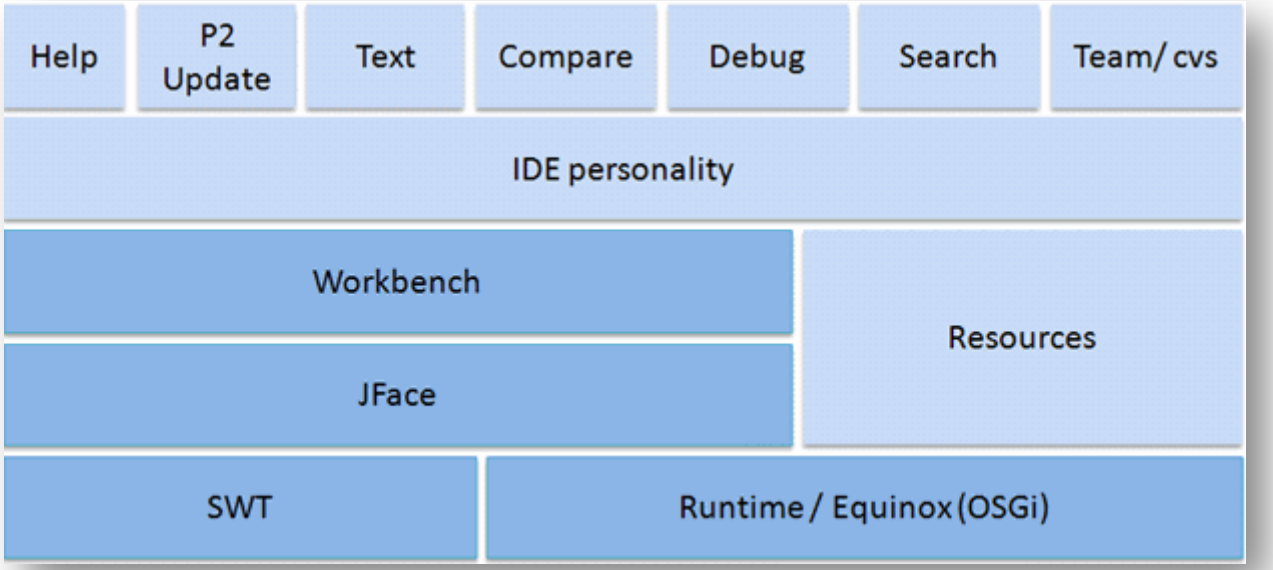
Eclipse para todos

- Eclipse IDE for Java EE Developers
- Eclipse IDE for JS Web Developers
- Eclipse for RCP and RAP Developers
- Eclipse IDE for C/C++ Linux Developers
- Eclipse for Testers
- Eclipse IDE for Parallel Application Developers
- **Eclipse Modeling Tools**
- …

Rich Client Platform (RCP)

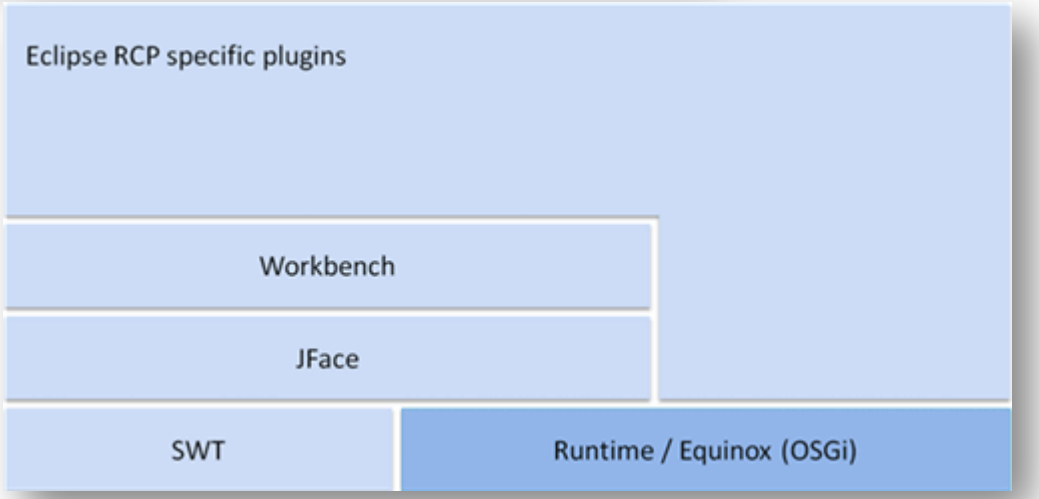
- Desde la versión 3.0, podemos utilizar la propia plataforma para crear aplicaciones de escritorio extensibles.
- Estas aplicaciones (RCP) se basan en el kernel de Eclipse y un conjunto de plugins seleccionados o desarrollados.
- Se distribuyen como aplicaciones independientes.

Eclipse IDE y Aplicaciones RCP



Eclipse IDE

Aplicación RCP



* Lars Vogel

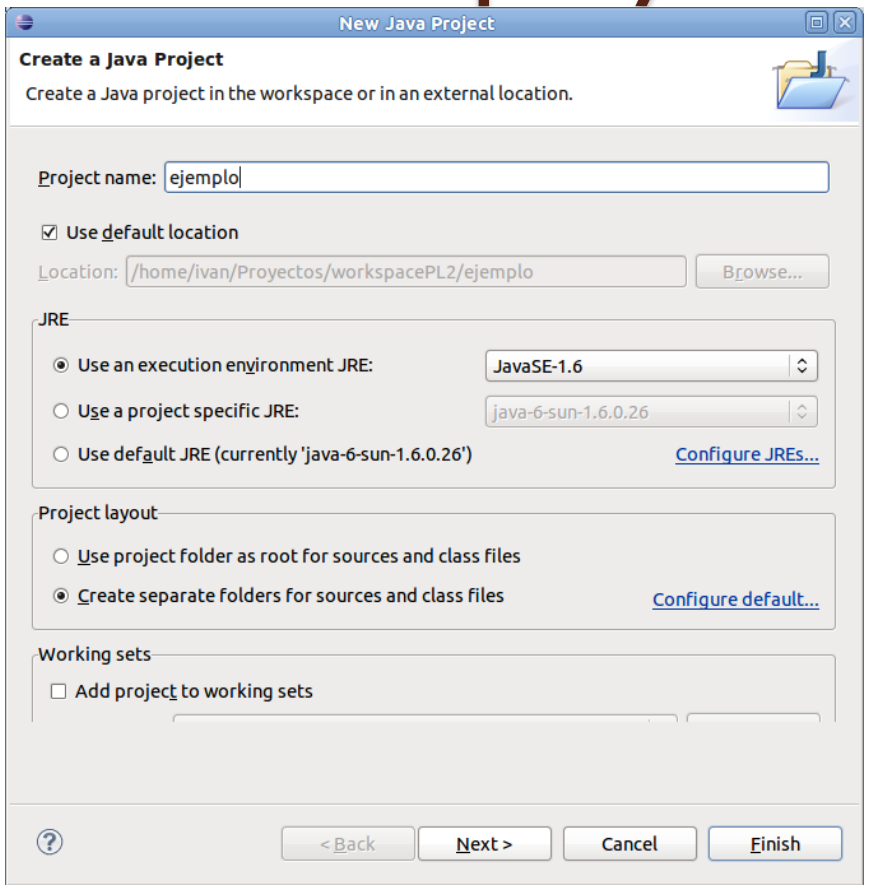
Aplicaciones basadas en Eclipse

- Adobe Flex Builder
- IBM Rational Method Composer,
- MyEclipse
- IBM Lotus Notes 8
- SpringSource Tool Suite
- Etc.

LA PLATAFORMA ECLIPSE

DESARROLLO DE UN PROYECTO JAVA

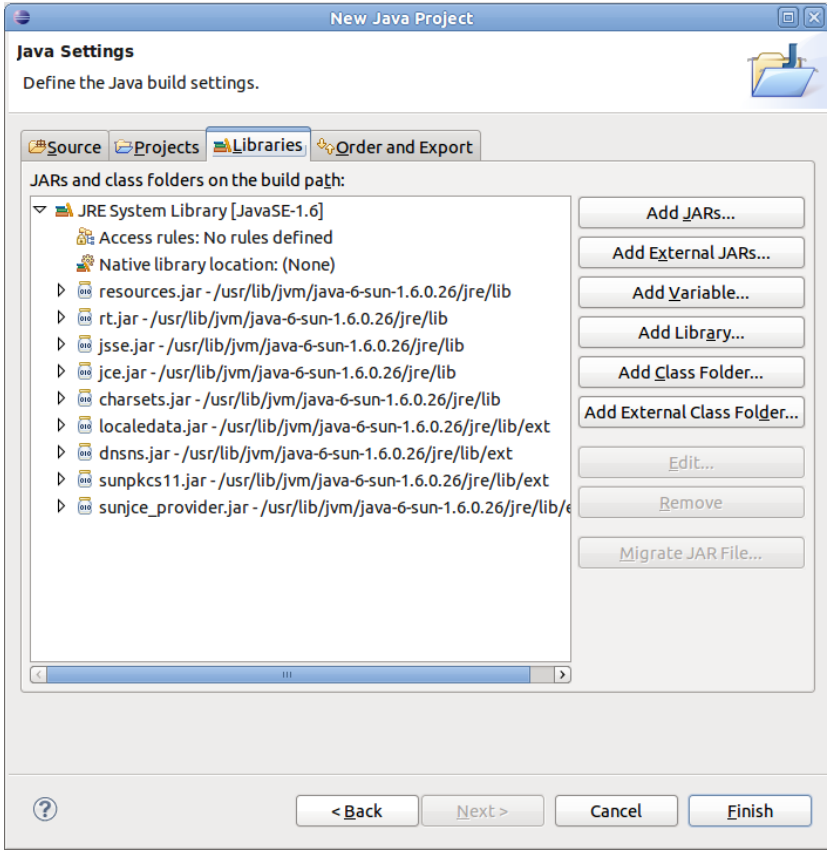
Creación de un proyecto Java



`File → New → Java Project`

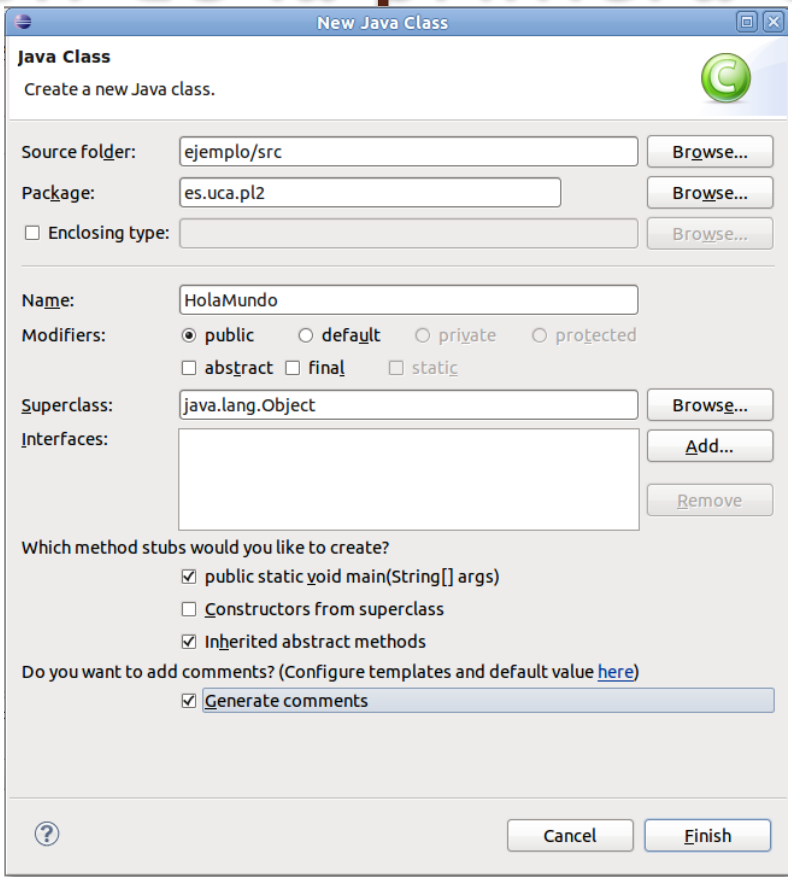
Es preciso definir, al menos, el nombre del proyecto y el entorno de ejecución de Java (JRE o JDK)

Configuración del proyecto



Estableceremos la carpeta del código fuente (*Source*), las dependencias con otros proyectos del workspace (*Projects*) y con JARs externos (*Libraries*) y el orden de compilación (*Order and Export*)

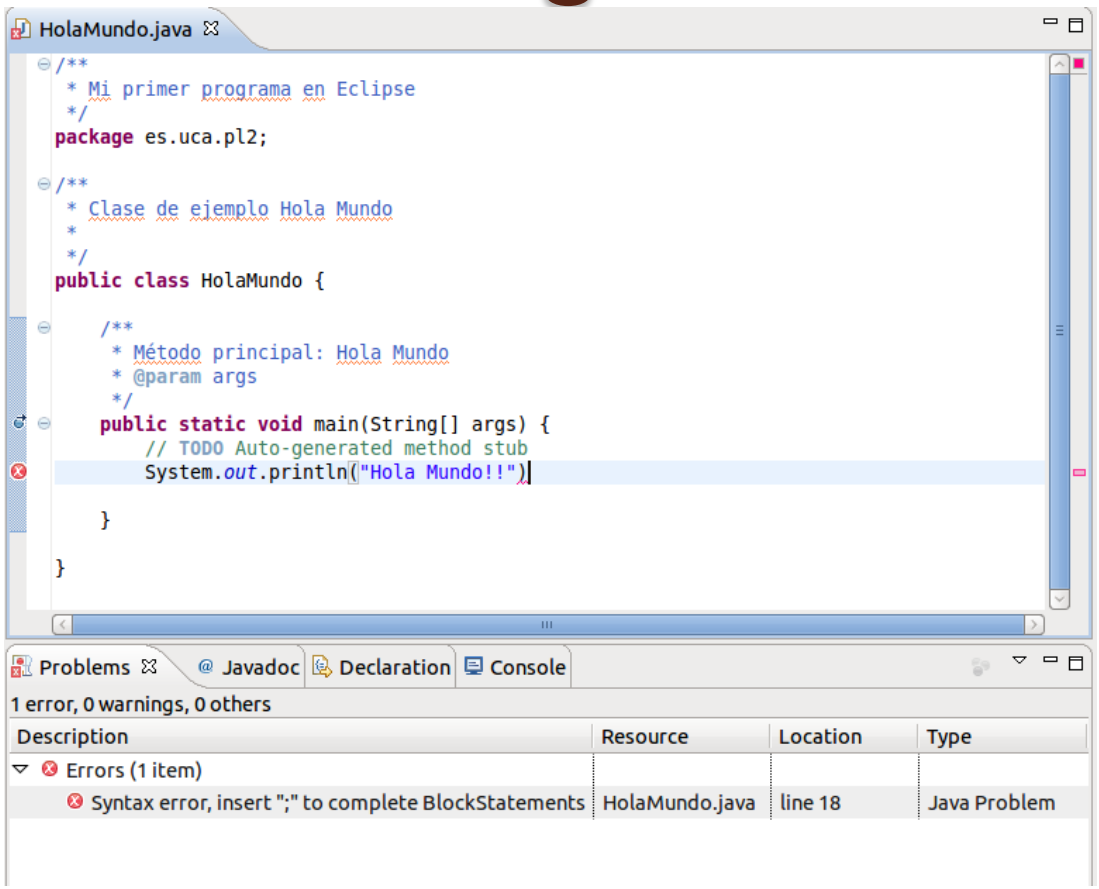
Creación de la primera clase Java



`File → New → Class`

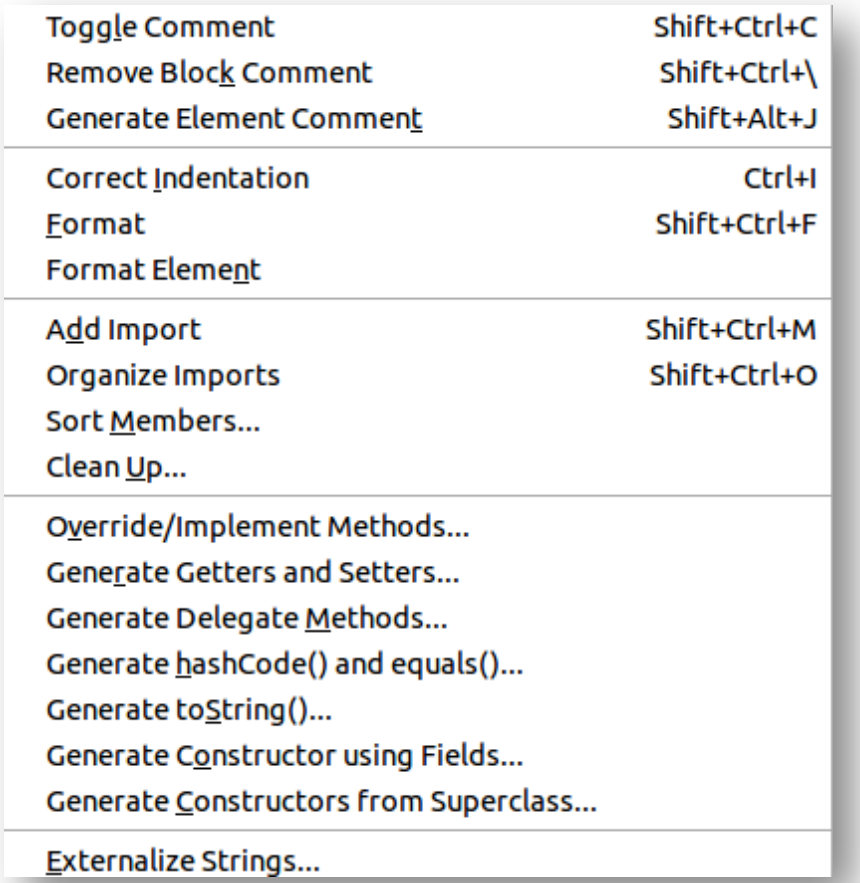
Definimos *nombre* de la clase, *paquete* donde se almacenará, *modificadores* de acceso, *superclases*, *interfaces* a implementar y si queremos añadir *method stubs* y *comentarios*

Escribiendo código

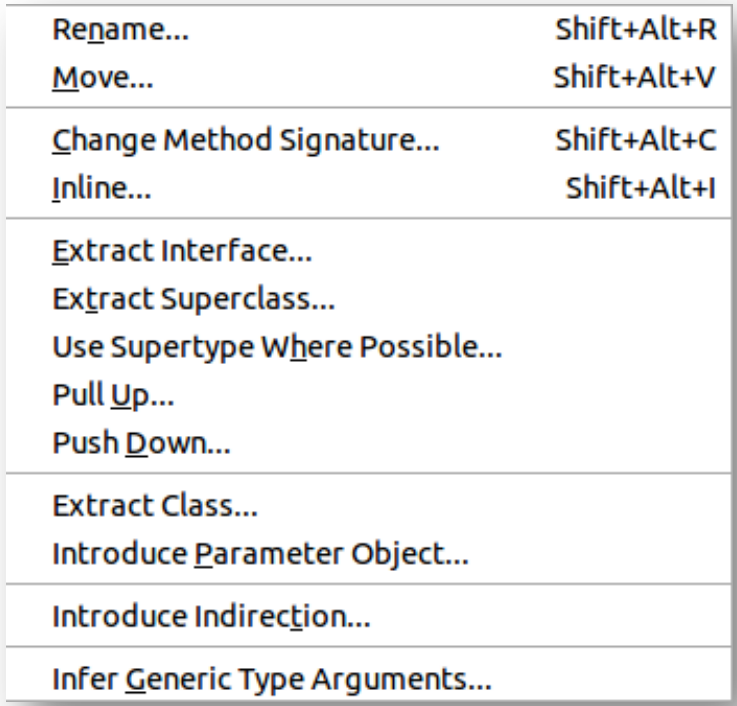


El editor de código de Eclipse ofrece autocompletado, quick fixes, coloreado de sintaxis e inclusión de plantillas de código y comentarios (JavaDoc), entre otras capacidades.

Mejorando el código



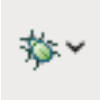
Source



Refactoring

Compilación y ejecución 

- Eclipse compila "al vuelo" los ficheros fuente que estamos modificando.
- Para ejecutar, tenemos que seleccionar la clase con el método *main* deseado y luego *Run As* → *Java Application* en el menú contextual (botón derecho).
- La vista *Console*, muestra la salida de la aplicación

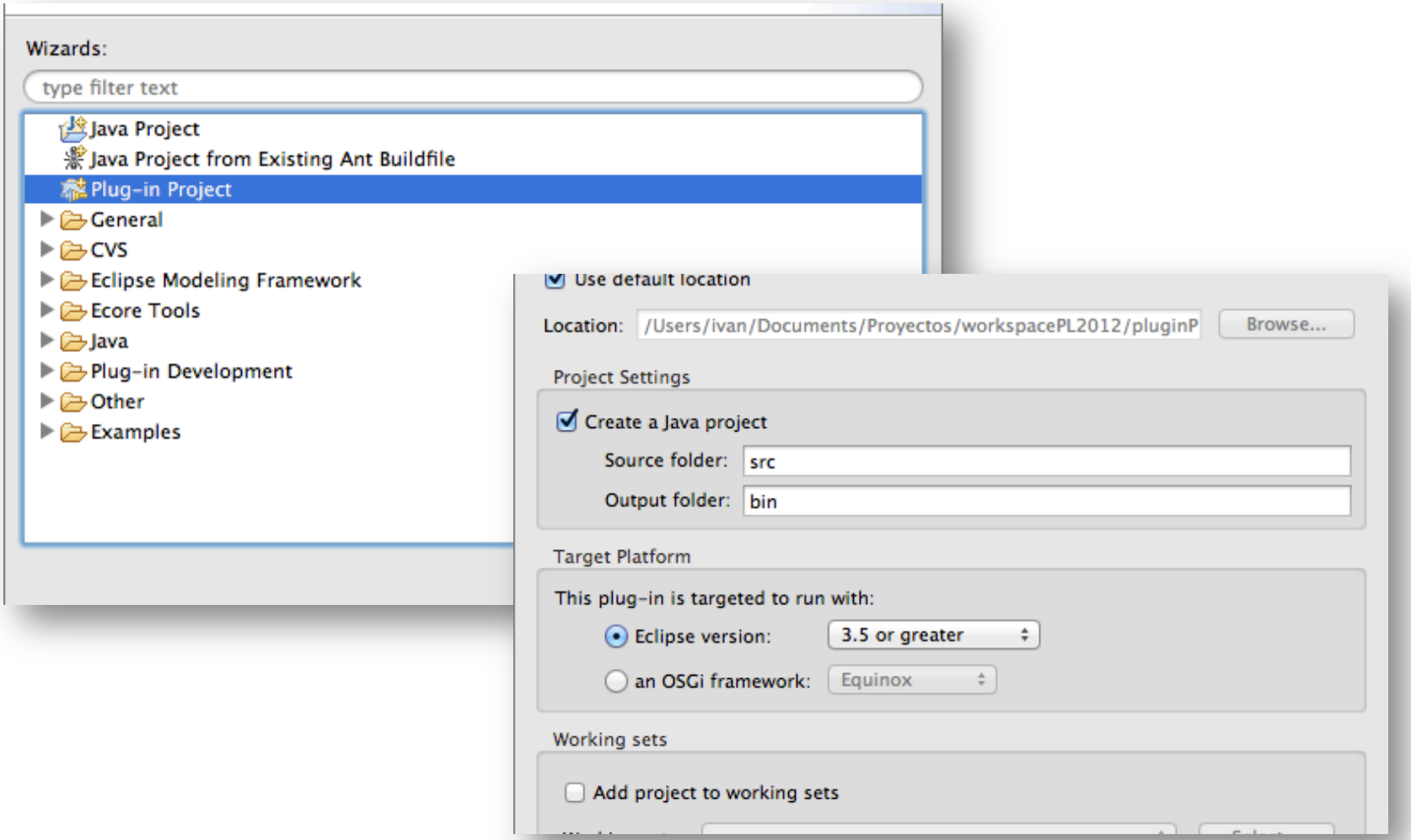
Depuración de código 

- Eclipse ofrece una completa perspectiva de depuración, incluyendo:
  - Sesión de depuración
  - Puntos de ruptura (breakpoints)
  - Inspección de variables/expresiones
- Control de la ejecución
  - *Debug:* inicia la depuración
  - *Step into*: ejecuta la instrucción, entrando en métodos
  - *Step over*: ejecuta la instrucción, sin entrar en métodos
  - *Step return*: ejecuta hasta final del método actual
  - *Pause*: detiene la ejecución
  - *Resume*: continúa con la ejecución de la aplicación
  - *Terminate*: finaliza el proceso

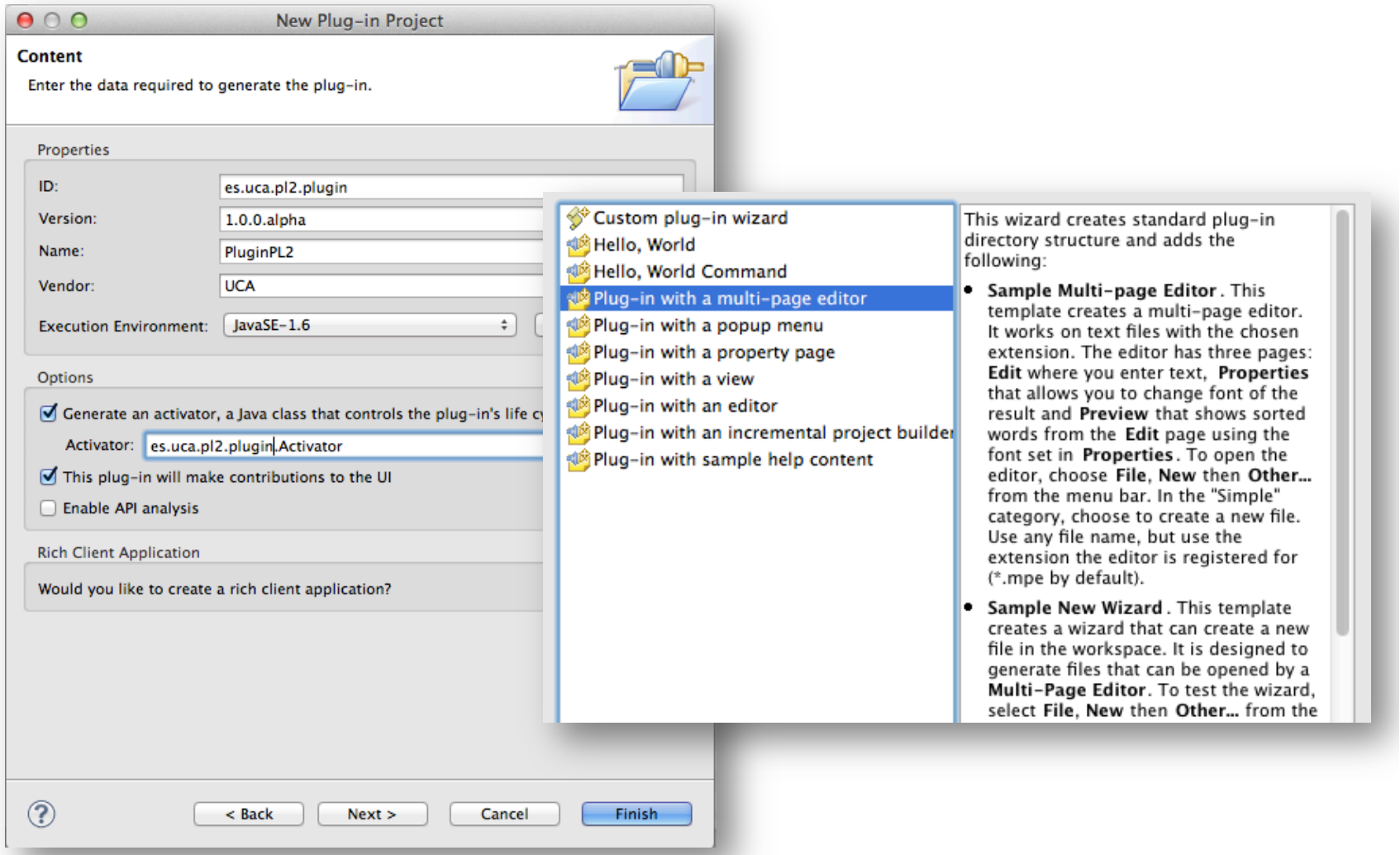
LA PLATAFORMA ECLIPSE

DESARROLLO DE UN PLUG-IN

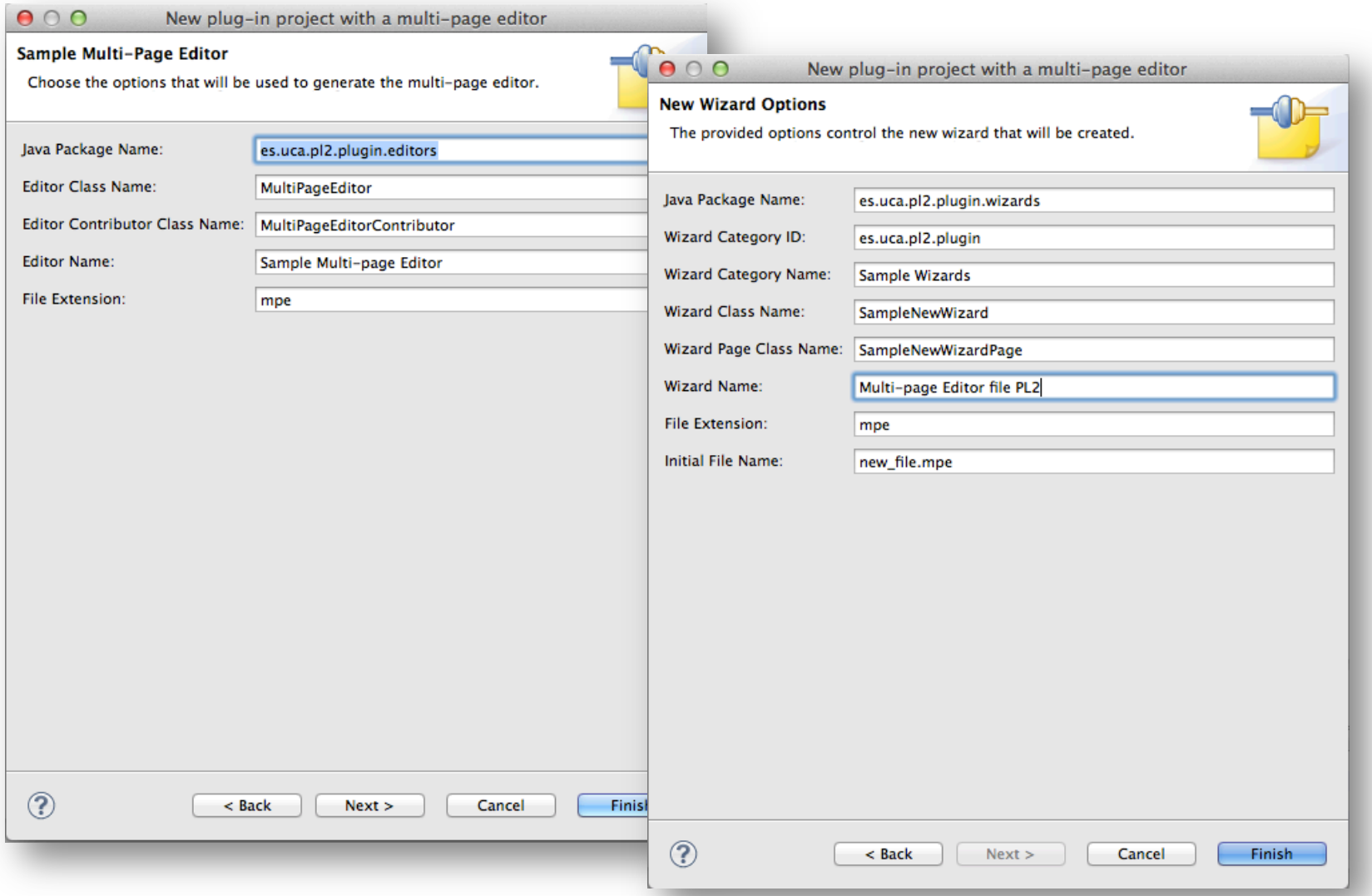
Creación de un plug-in



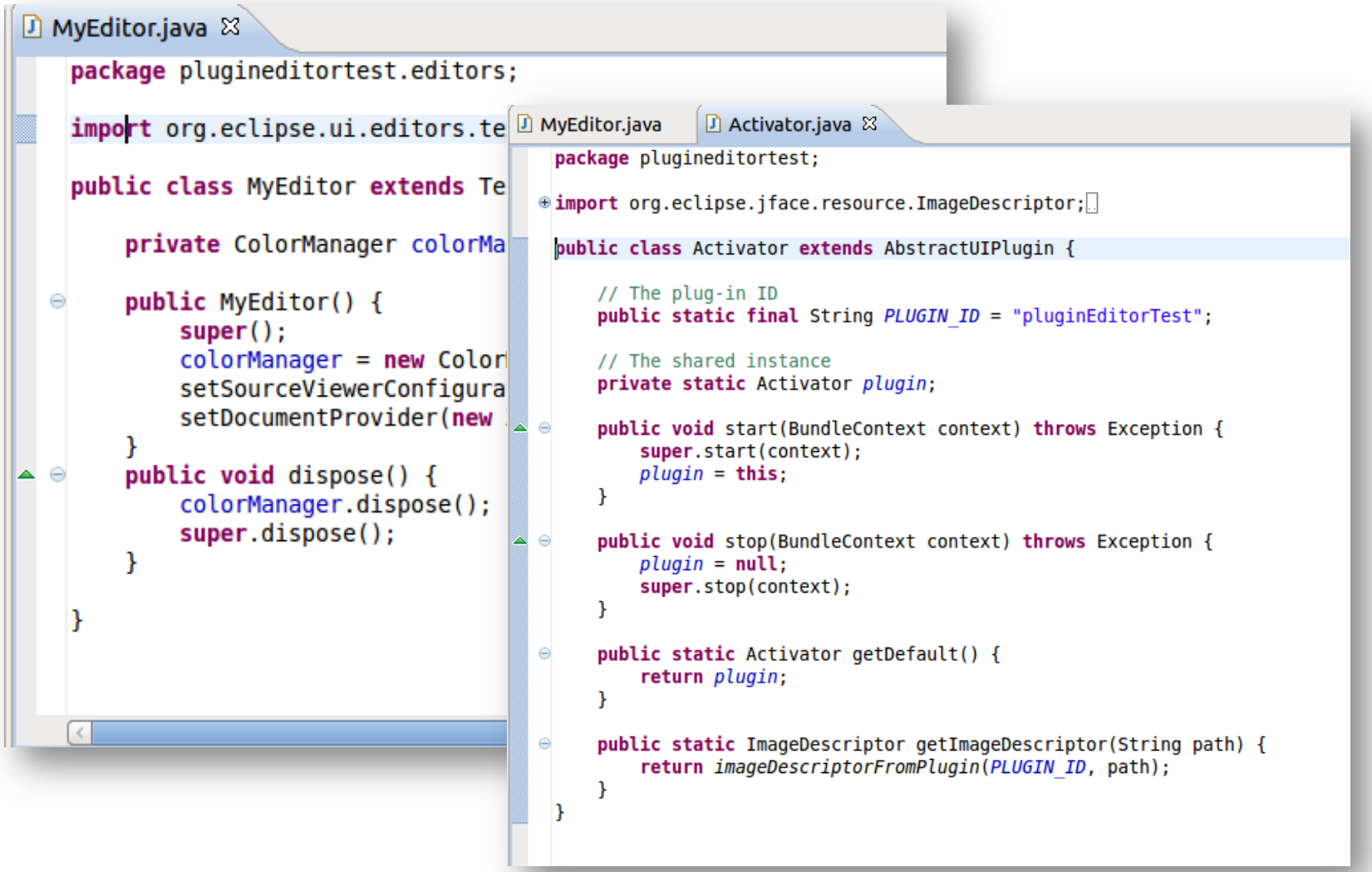
Creación de un plug-in (II)



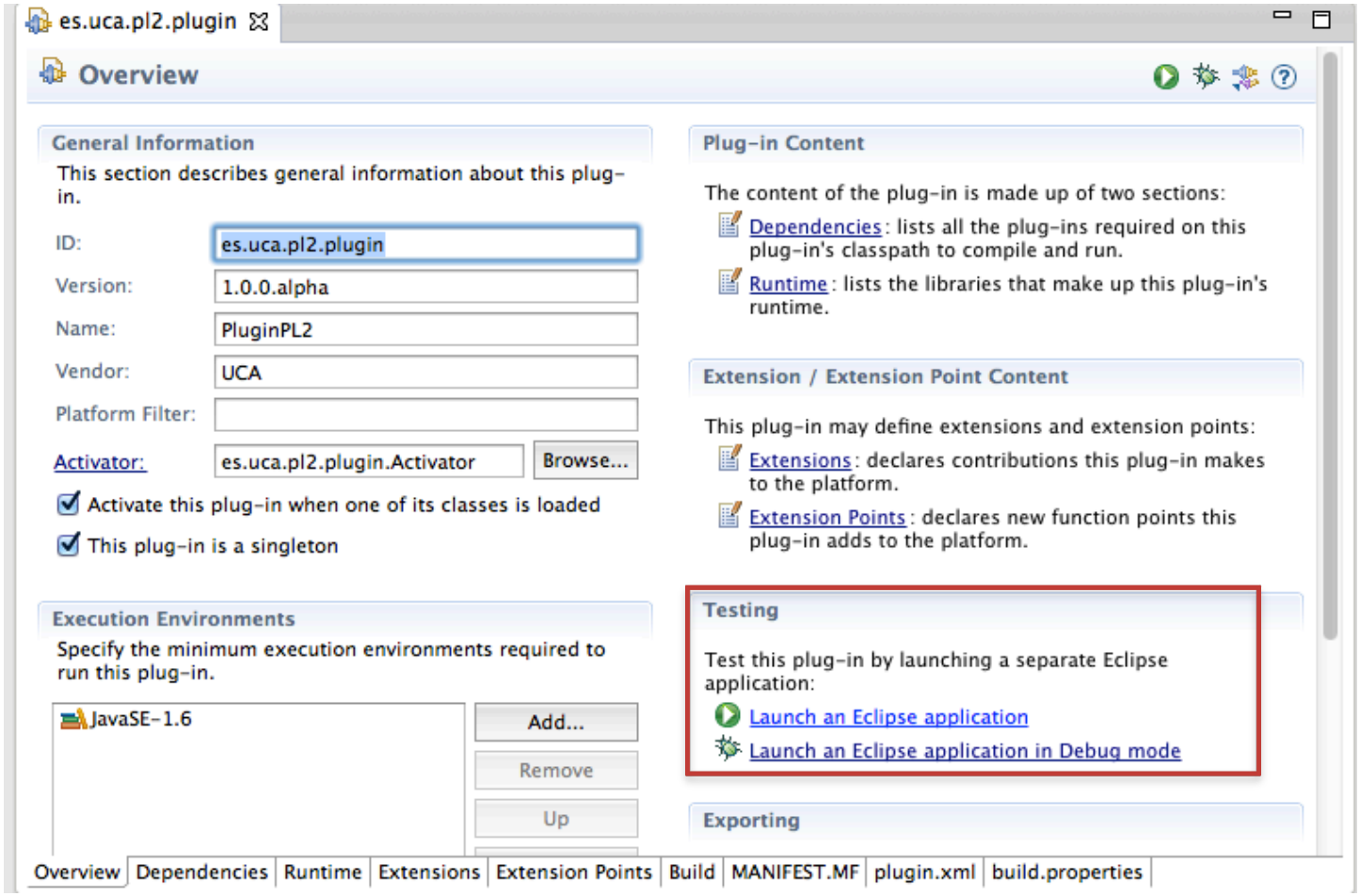
Creación de un plug-in (III)



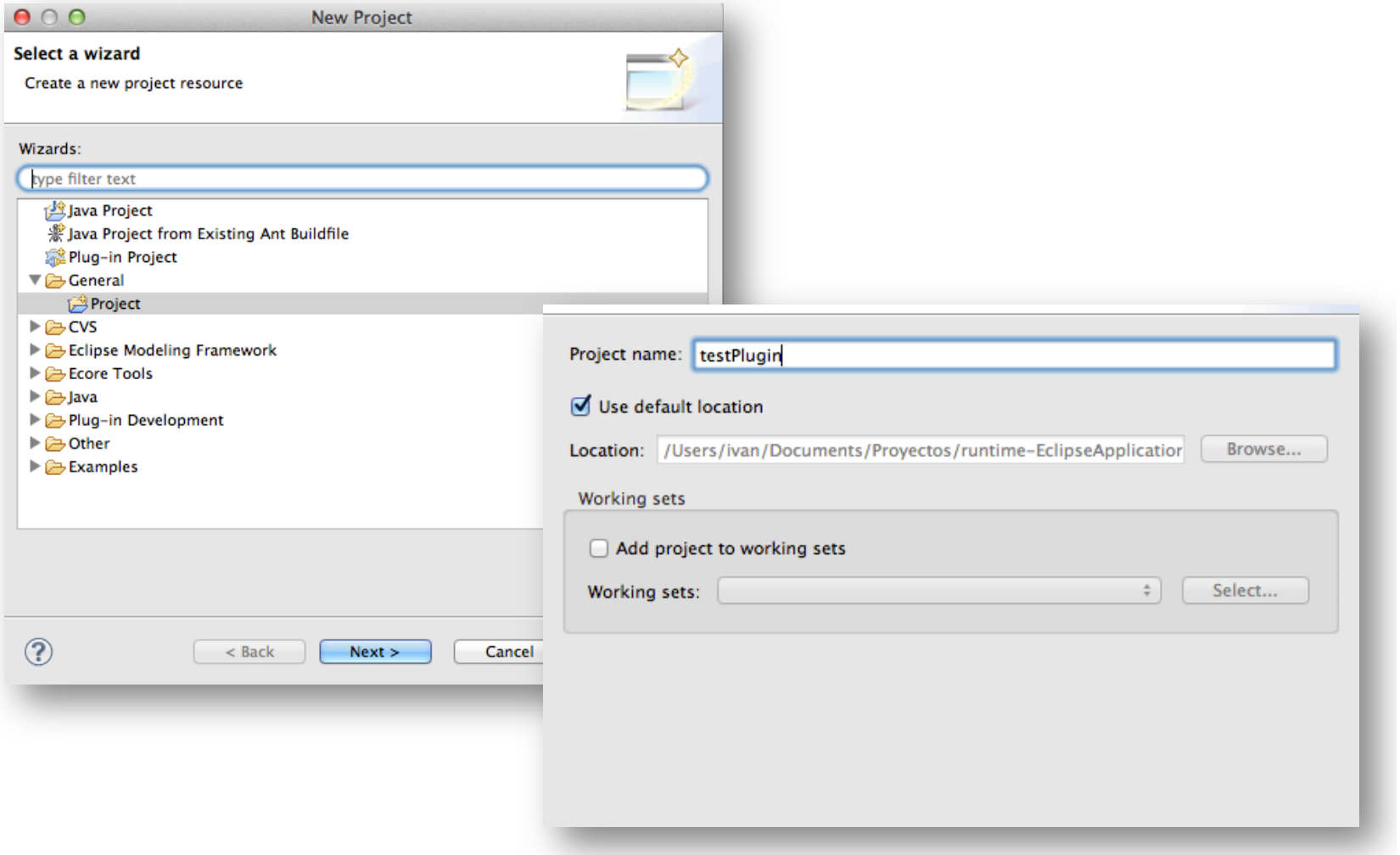
Creación de un plug-in (IV)



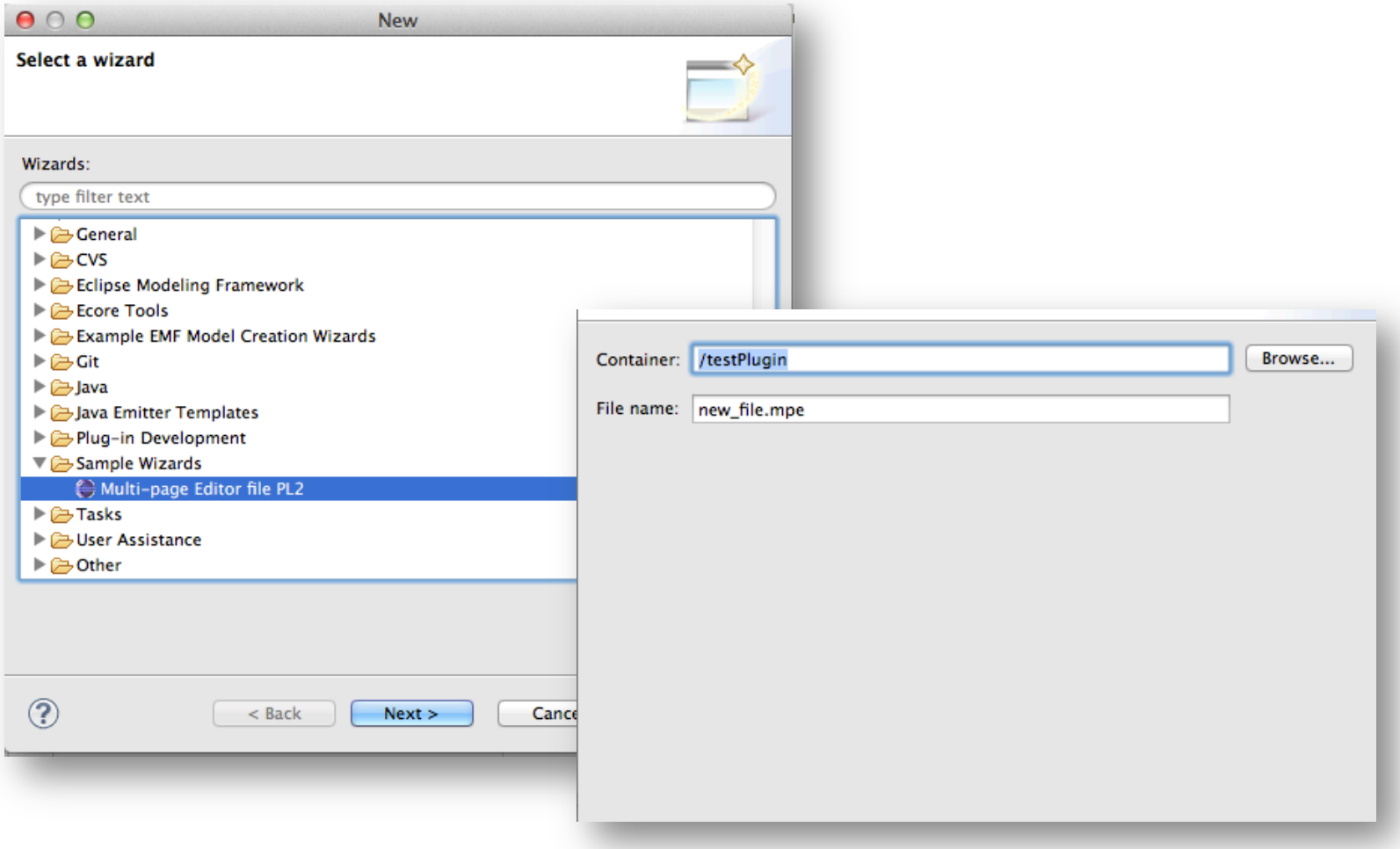
Test del plug-in



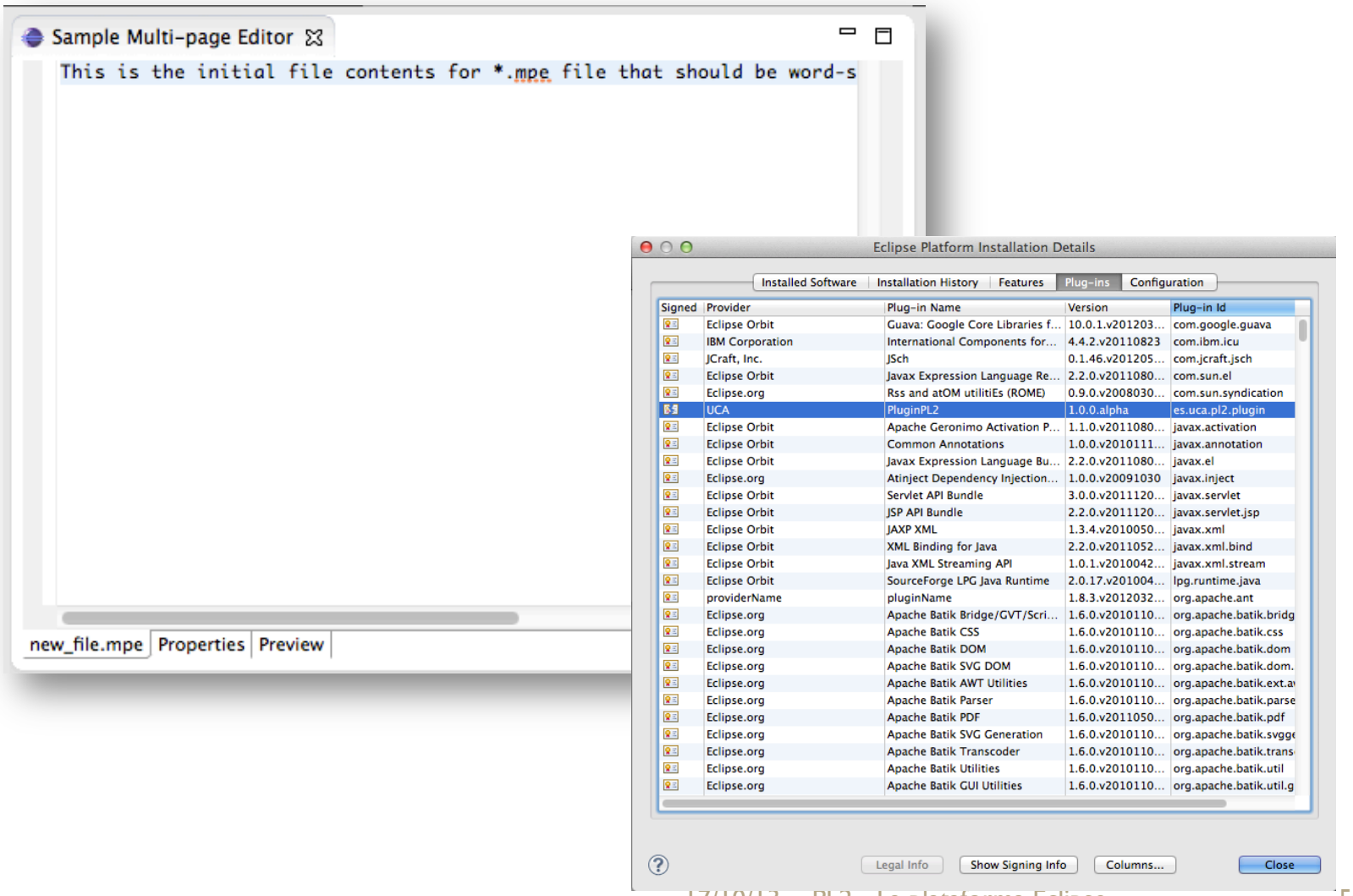
Test del plug-in (II)



Test del plug-in (III)



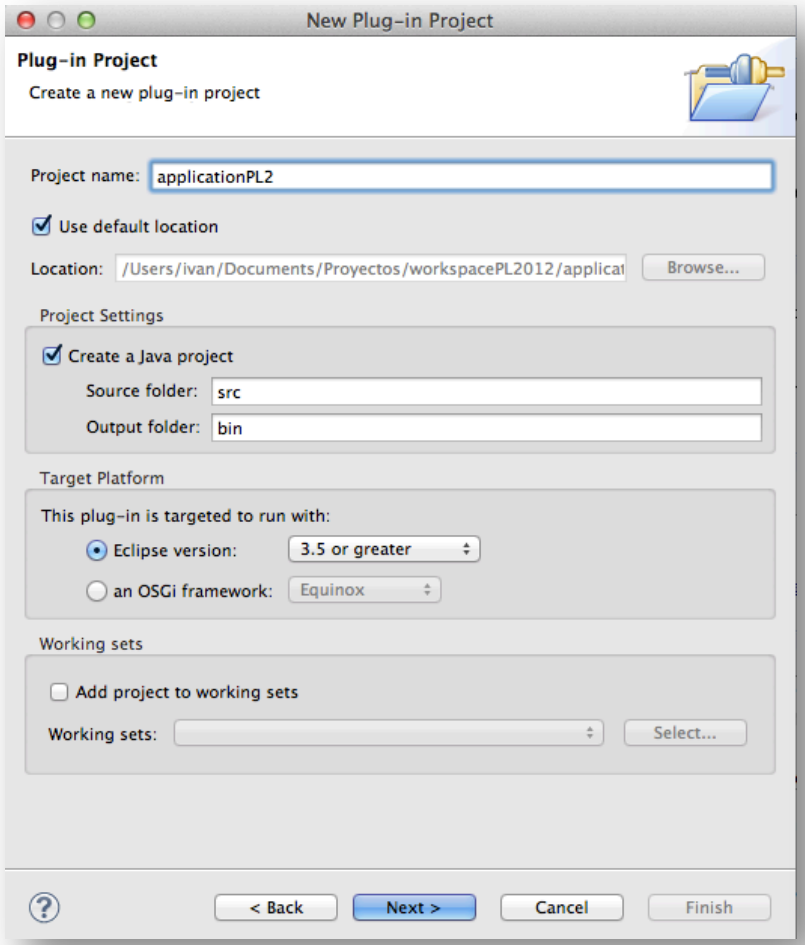
Test del plug-in (IV)

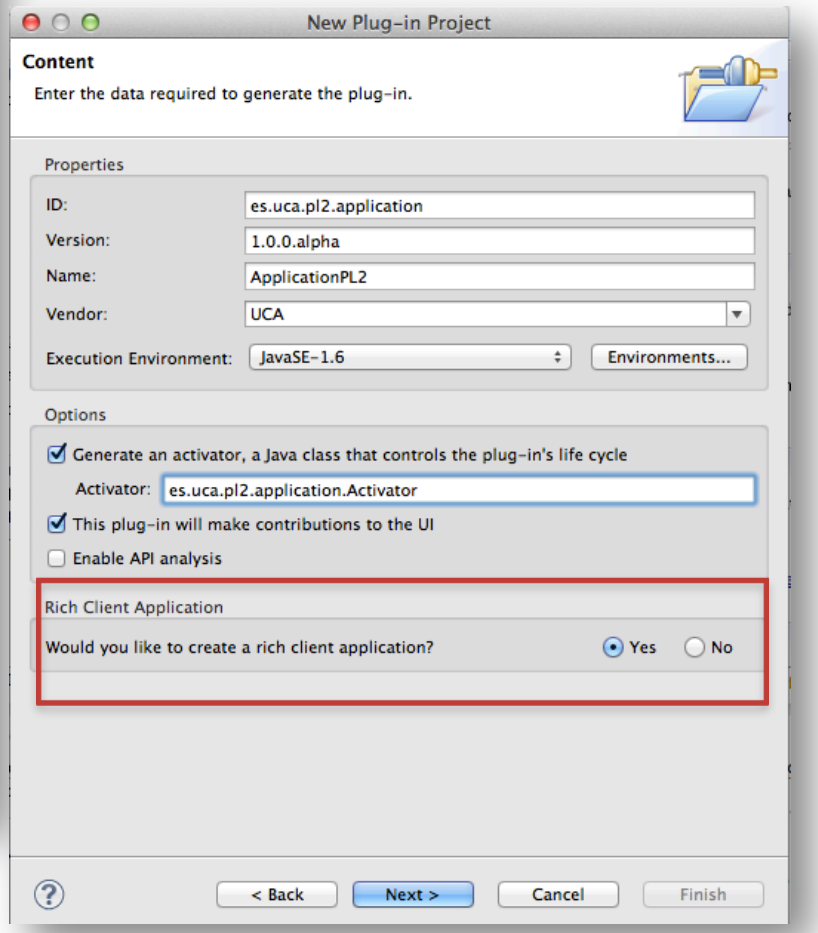


LA PLATAFORMA ECLIPSE

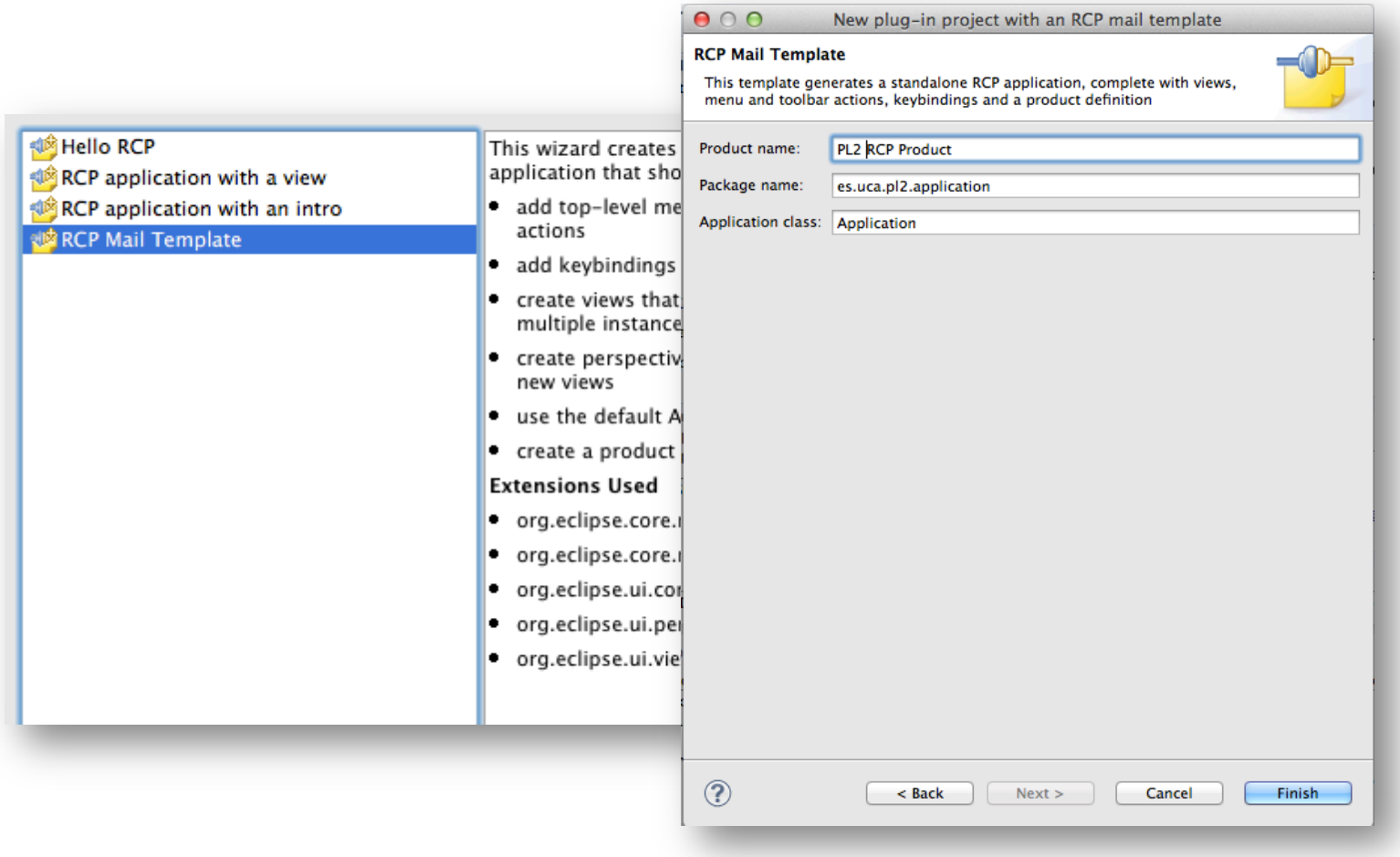
DESARROLLO DE UNA APLICACIÓN RCP

Creación de una aplicación RCP

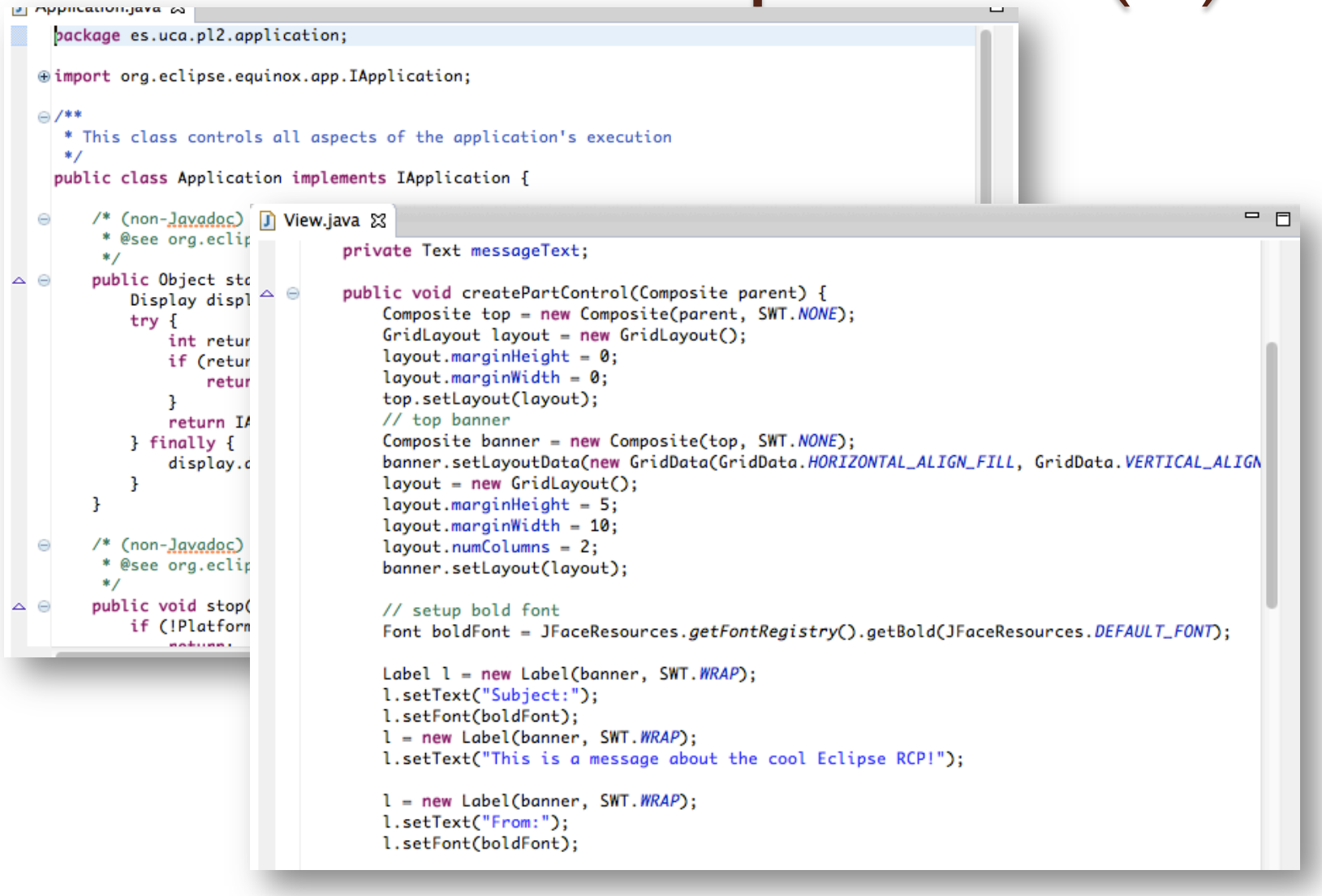




Creación de una aplicación RCP (II)



Creación de una aplicación (III)

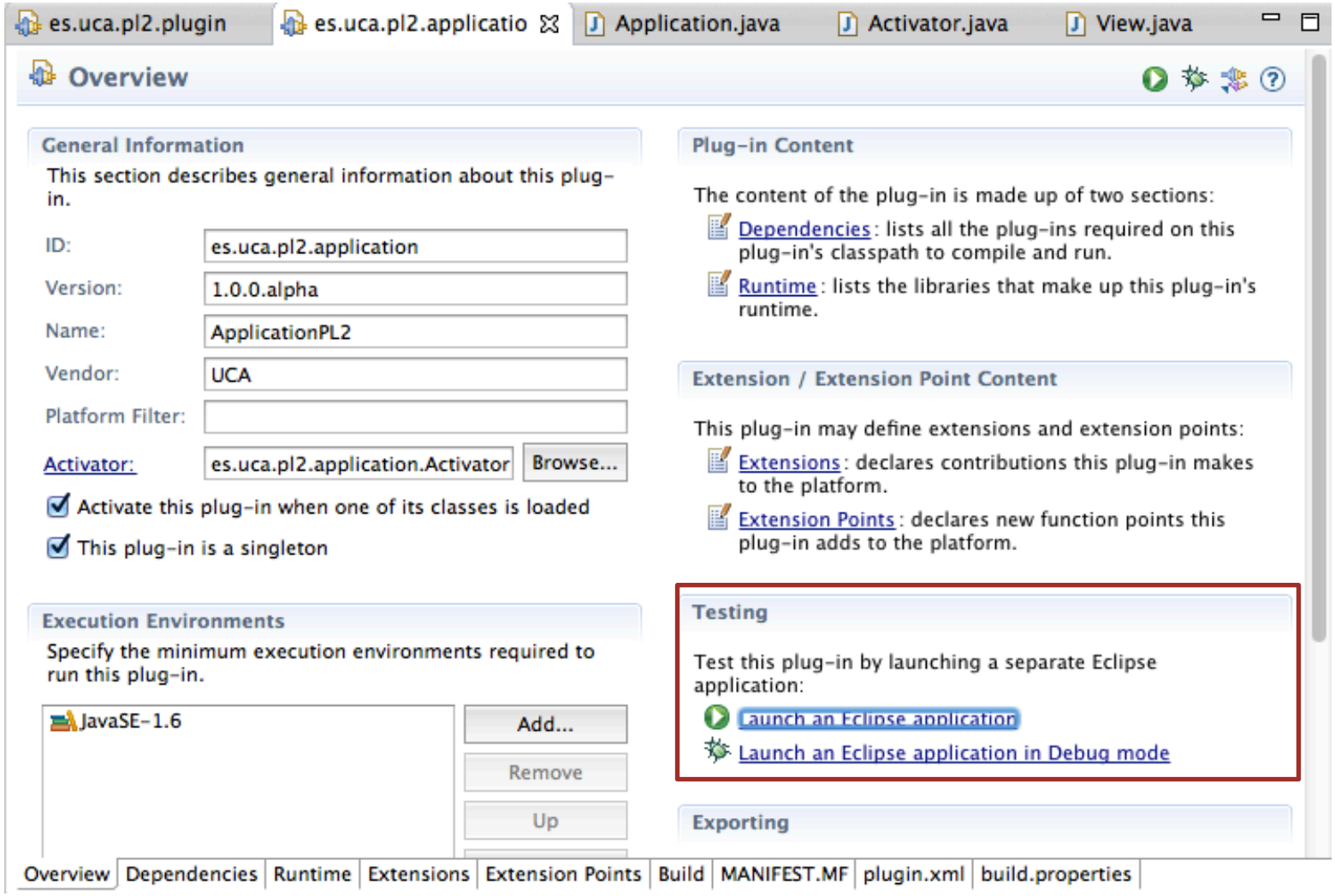


```
package es.uca.pl2.application;

import org.eclipse.equinox.app.IApplication;

/**
 * This class controls all aspects of the application's execution
 */
public class Application implements IApplication {

    /* (non-Javadoc)
     * @see org.eclip
     */
    public Object sta
        Display displ
        try {
            int retur
            if (retur
                retur
            }
            return IA
        } finally {
            display.d
        }
    }

    /* (non-Javadoc)
     * @see org.eclip
     */
    public void stop(
        if (!Platform
```

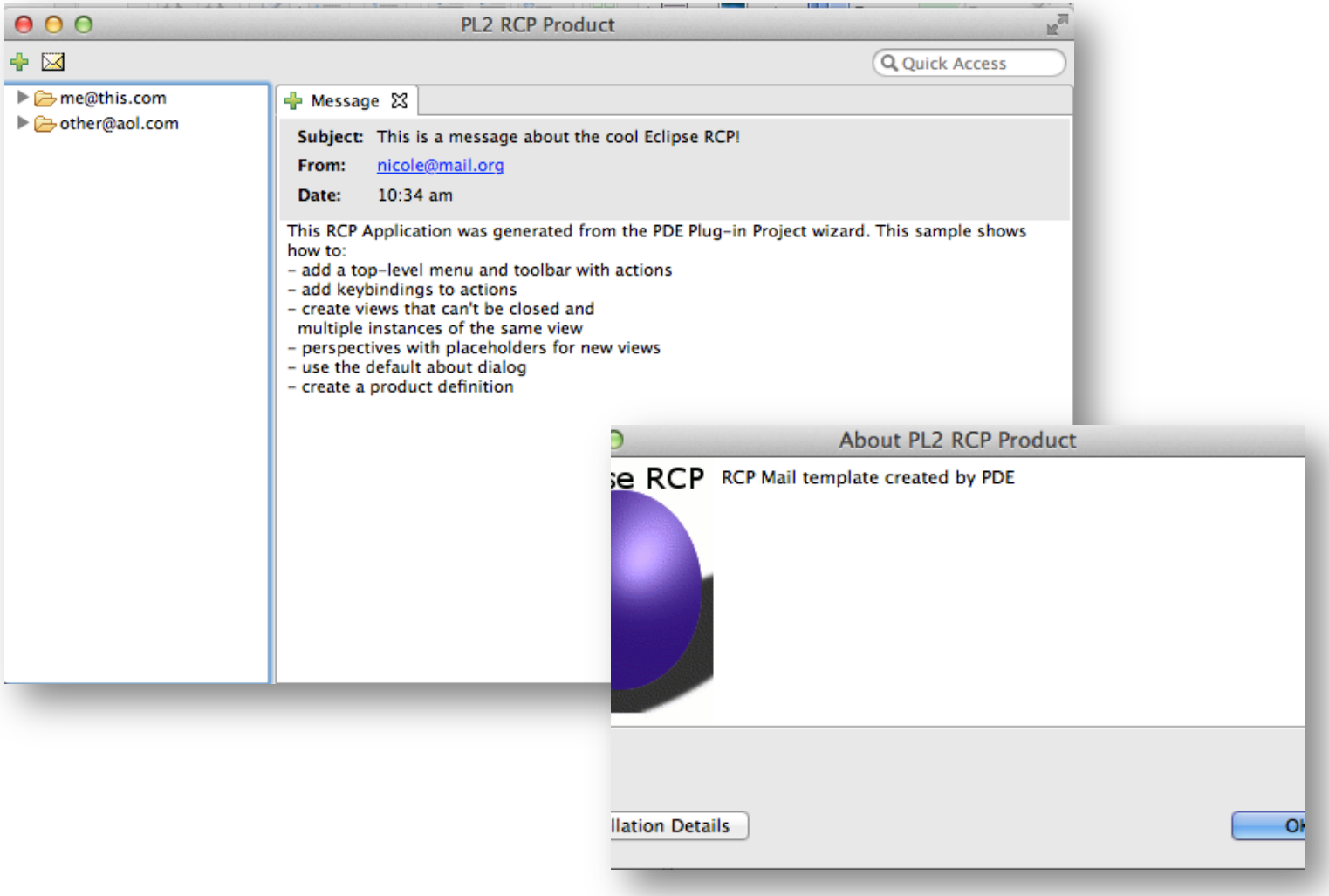
View.java

```
    private Text messageText;

    public void createPartControl(Composite parent) {
        Composite top = new Composite(parent, SWT.NONE);
        GridLayout layout = new GridLayout();
        layout.marginHeight = 0;
        layout.marginWidth = 0;
        top.setLayout(layout);
        // top banner
        Composite banner = new Composite(top, SWT.NONE);
        banner.setLayoutData(new GridData(GridData.HORIZONTAL_ALIGN_FILL, GridData.VERTICAL_ALIGN
        layout = new GridLayout();
        layout.marginHeight = 5;
        layout.marginWidth = 10;
        layout.numColumns = 2;
        banner.setLayout(layout);

        // setup bold font
        Font boldFont = JFaceResources.getFontRegistry().getBold(JFaceResources.DEFAULT_FONT);

        Label l = new Label(banner, SWT.WRAP);
        l.setText("Subject:");
        l.setFont(boldFont);
        l = new Label(banner, SWT.WRAP);
        l.setText("This is a message about the cool Eclipse RCP!");

        l = new Label(banner, SWT.WRAP);
        l.setText("From:");
        l.setFont(boldFont);
```

Test de una aplicación RCP



Test de una aplicación RCP (II)



DESARROLLO DE EDITORES CON ECLIPSE

RESUMEN

¿Qué hemos aprendido hoy?

- Conocer Eclipse y sus proyectos principales
- Componentes principales del IDE
- Arquitectura modular basado en un kernel y un conjunto de plugins.
- Desarrollar plugins dentro del propio IDE o como aplicaciones independientes a partir de unas plantillas.

Procesadores de Lenguajes 2

La plataforma Eclipse

Curso 2013-2014

Iván Ruiz Rube
ivan.ruiz@uca.es