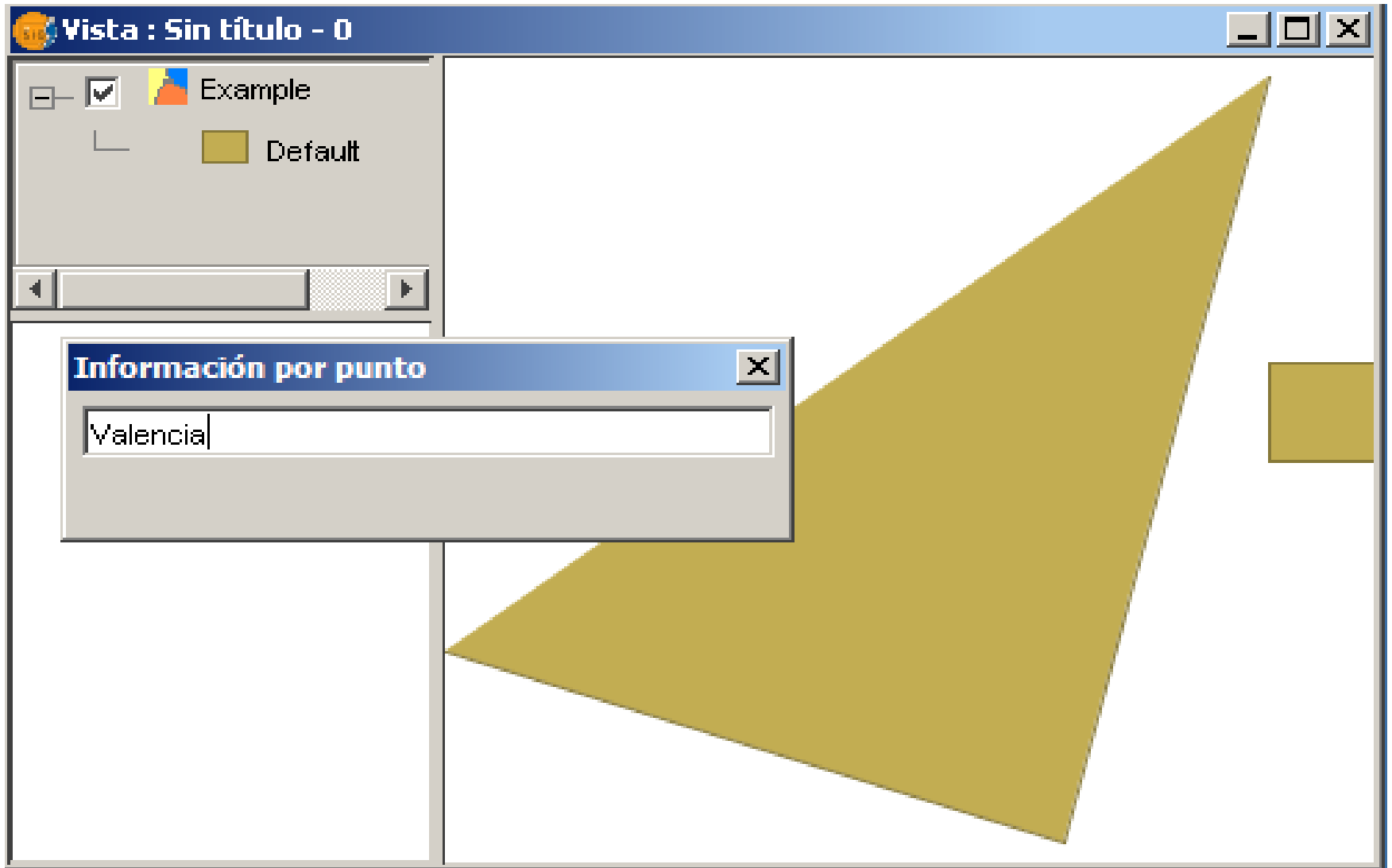




Crear una extensión desde 0 en gvSIG

Jorge Piera (piera_jor@gva.es)

- Vamos a crear una extensión desde 0
- La extensión creará una capa vectorial donde dibujaremos polígonos por código
- Los polígonos se corresponderán con provincias, y tendrán un nombre y un número de habitantes
- Crearemos una herramienta que devolverá el nombre de la provincia al seleccionarla en la vista





Crear proyecto

The screenshot displays the Eclipse IDE environment. The Package Explorer on the left shows a project named 'fwAndami' with the following structure:

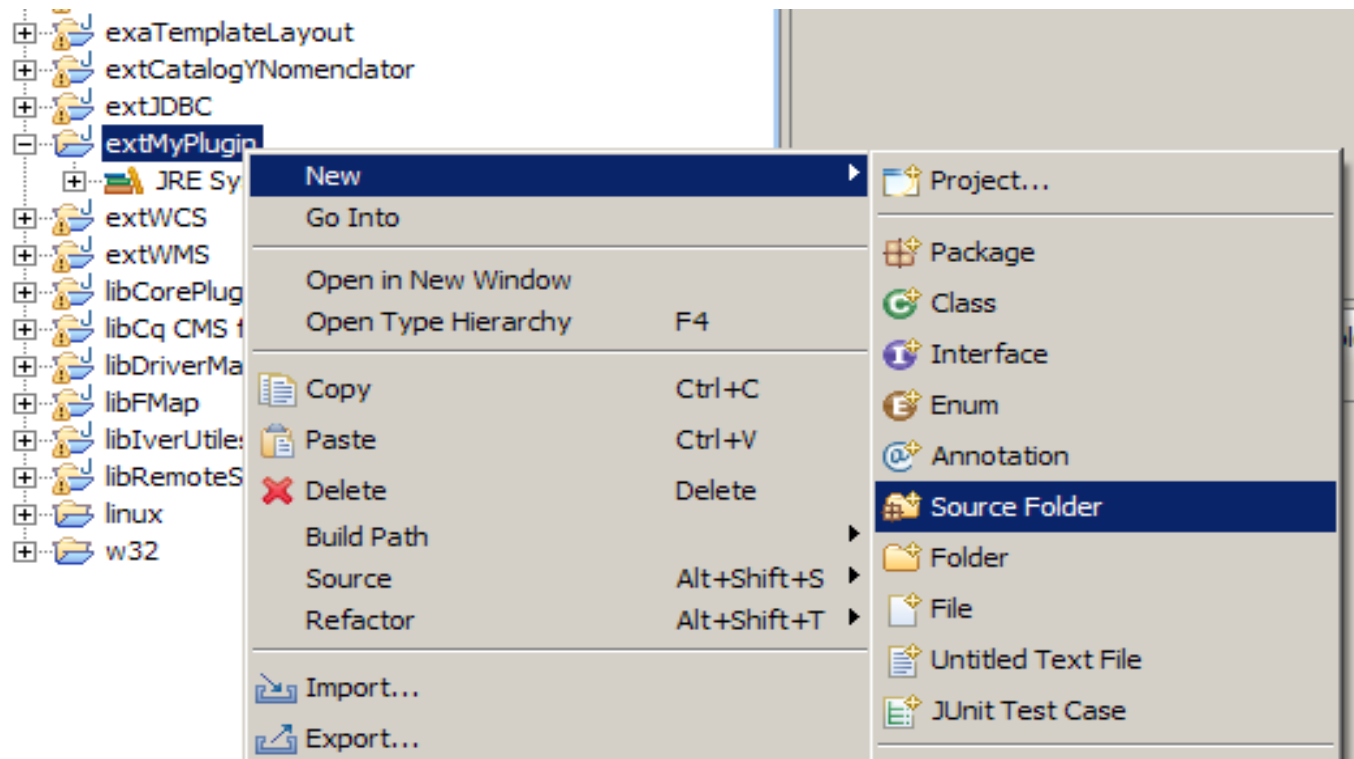
- fwAndami
 - appCatalogYNomenclatorClient
 - appgvSIG
 - exaCreateTable
 - exaExample1
 - exaLoadLayer
 - exaTemplateLayout
 - extCatalogYNomenclator
 - extJDBC
 - extWCS
 - extWMS
 - libCorePlugin
 - libCq CMS for java
 - libDriverManager
 - libFMap
 - libIverUtiles
 - libRemoteServices
 - linux
 - w32

The 'New' context menu is open, and the 'Project...' option is selected, showing a sub-menu with the following items:

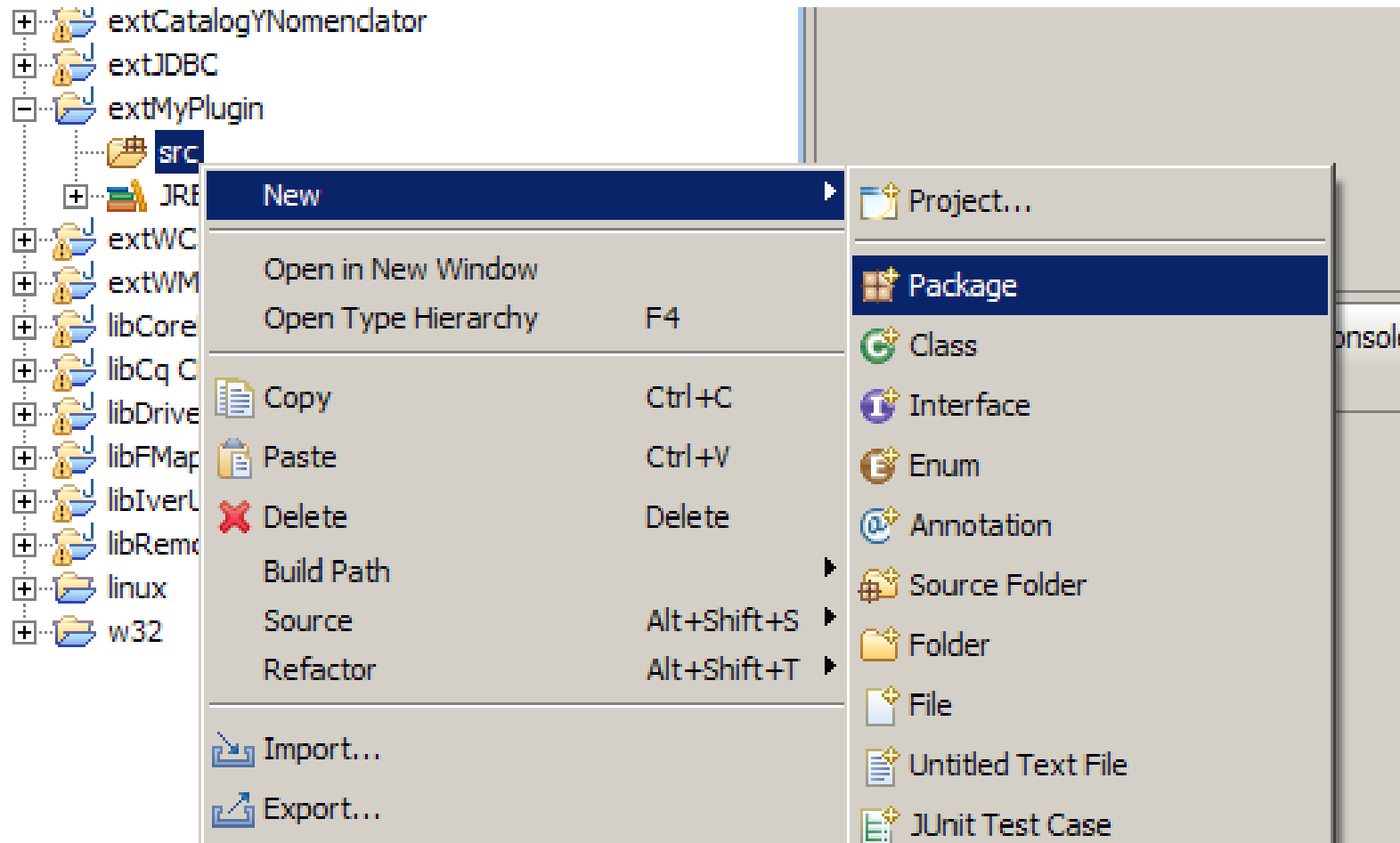
- Project...
- Package
- Class
- Interface
- Enum
- Annotation
- Source Folder
- Folder
- File
- Untitled Text File
- JUnit Test Case
- Example...
- Other... (Ctrl+N)

Crear carpeta fuentes

- Seleccionar “Java Project” y darle un nombre al proyecto (p.e. ExtMyPlugin)
- Crear una carpeta llamada “src” para poner los fuentes



- Hay que crear un paquete llamado “com.iver.cit.gvsig.myplugin”



The screenshot shows an IDE interface. On the left, a project tree is visible with the following structure:

- extCatalogYNomendator
- extJDBC
- extMyPlugin
 - src
- JRE
- extWC
- extWM
- libCore
- libCq C
- libDrive
- libFMap
- libIverL
- libRemo
- linux
- w32

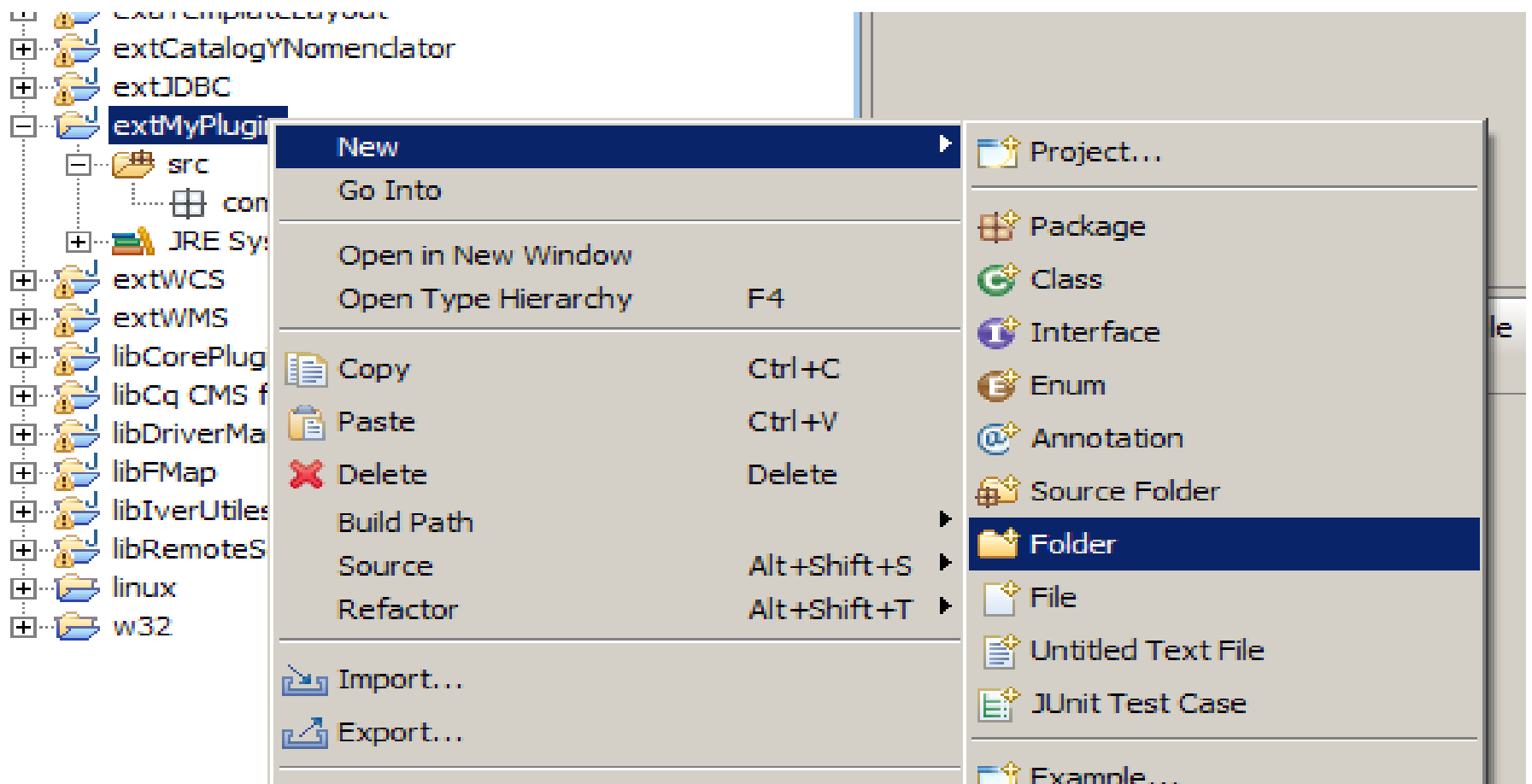
A context menu is open over the 'src' folder, showing the following options:

- New
- Open in New Window
- Open Type Hierarchy F4
- Copy Ctrl+C
- Paste Ctrl+V
- Delete Delete
- Build Path
- Source Alt+Shift+S
- Refactor Alt+Shift+T
- Import...
- Export...

The 'New' menu is expanded, showing the following options:

- Project...
- Package
- Class
- Interface
- Enum
- Annotation
- Source Folder
- Folder
- File
- Untitled Text File
- JUnit Test Case

- Crear las carpetas config e images



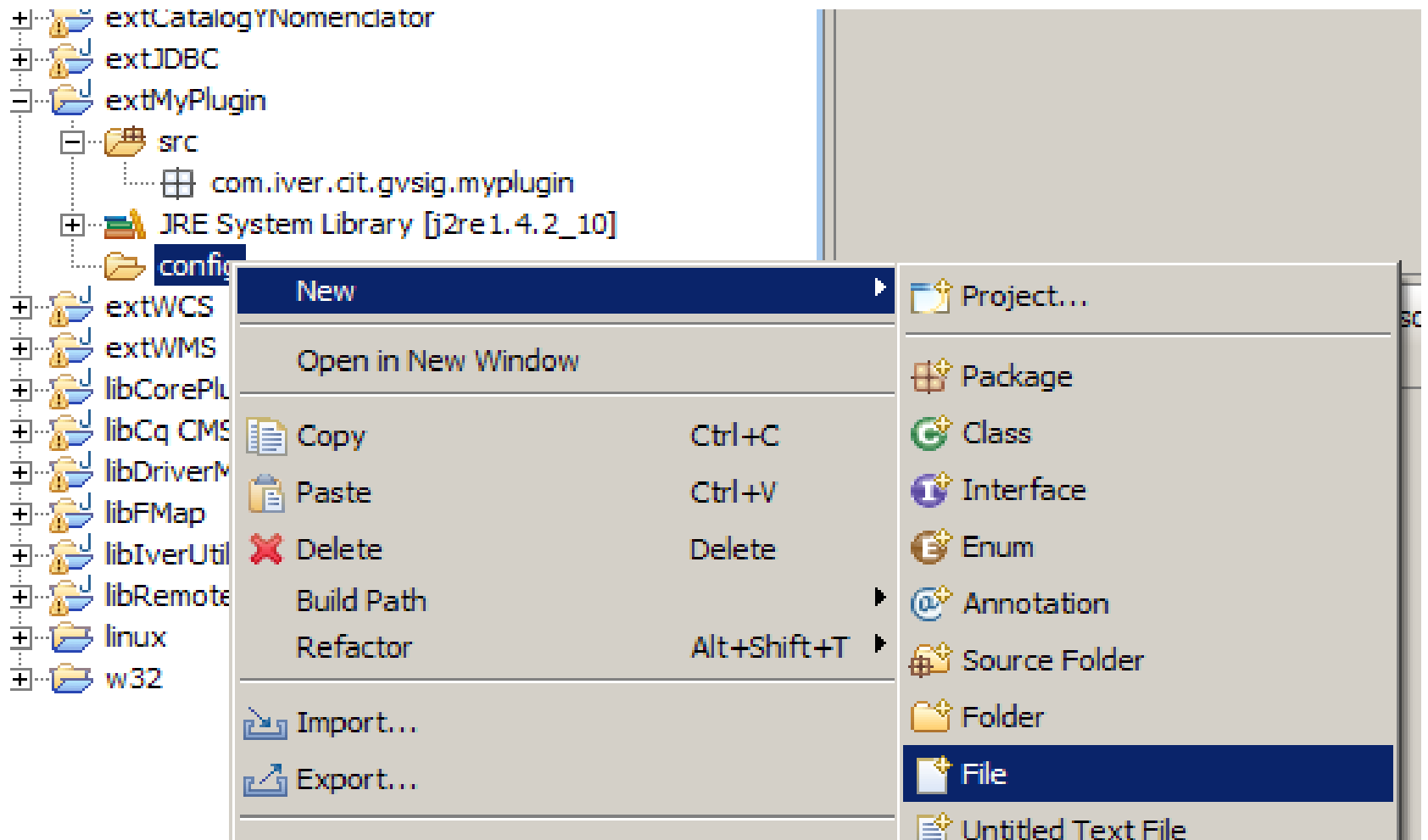
The screenshot shows an IDE interface with a project tree on the left and a 'New' context menu open over the 'src' folder of the 'extMyPlugin' project. The 'New' menu is expanded, showing various options for creating new elements. The 'Folder' option is highlighted in blue.

Option	Shortcut
New	
Go Into	
Open in New Window	
Open Type Hierarchy	F4
Copy	Ctrl+C
Paste	Ctrl+V
Delete	Delete
Build Path	
Source	Alt+Shift+S
Refactor	Alt+Shift+T
Import...	
Export...	

The 'New' menu options include:

- Project...
- Package
- Class
- Interface
- Enum
- Annotation
- Source Folder
- Folder**
- File
- Untitled Text File
- JUnit Test Case
- Example...

- Crear el fichero “config.xml”



The screenshot shows a project tree on the left with the following structure:

- extCatalogYNomenclator
- extJDBC
- extMyPlugin
 - src
 - com.iver.cit.gvsig.myplugin
 - JRE System Library [j2re 1.4.2_10]
 - config** (highlighted)
- extWCS
- extWMS
- libCorePlu
- libCq CMS
- libDriverM
- libFMap
- libIverUtil
- libRemote
- linux
- w32

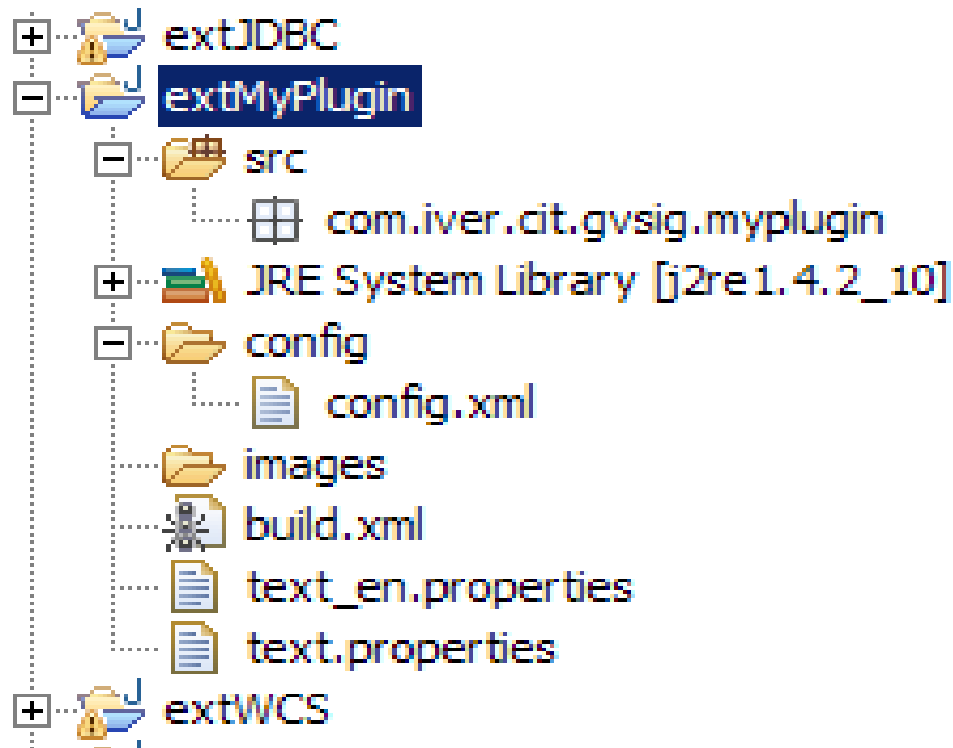
A context menu is open over the 'config' folder, showing the following options:

- New (highlighted)
- Open in New Window
- Copy (Ctrl+C)
- Paste (Ctrl+V)
- Delete (Delete)
- Build Path
- Refactor (Alt+Shift+T)
- Import...
- Export...

The 'New' submenu is open, showing the following options:

- Project...
- Package
- Class
- Interface
- Enum
- Annotation
- Source Folder
- Folder
- File** (highlighted)
- Untitled Text File

- Crear los ficheros para las traducciones en castellano y en inglés (text.properties y text_en.properties)
- Crear el fichero build.xml



Build.xml

- Copiar el contenido del build.xml del proyecto exaExample1
- Cambiar la línea

```
<property name="plugin" value="com.iver.ejemplo"/>
```

Por

```
<property name="plugin" value="com.iver.cit.gvsig.myplugin"/>
```

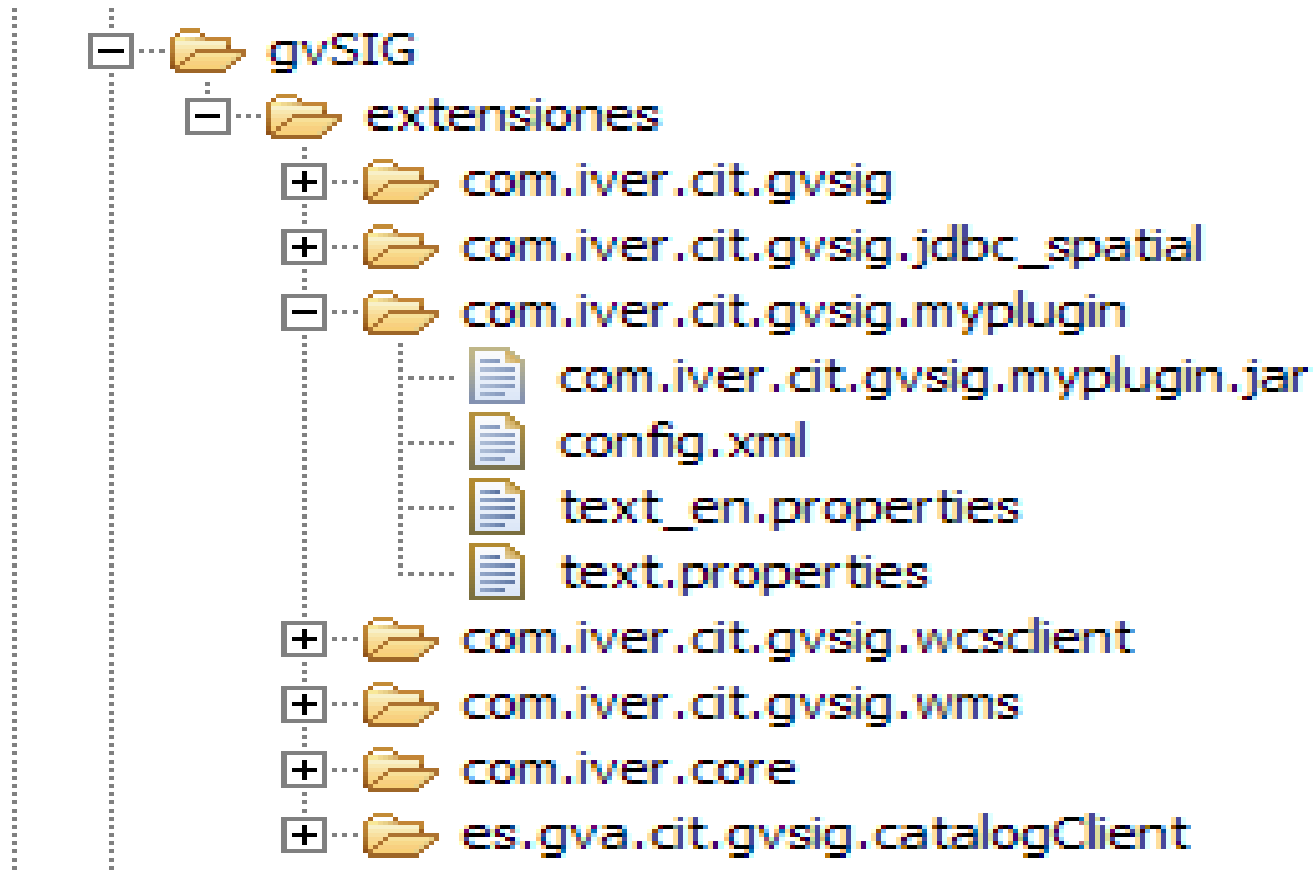
- Eliminar la línea

```
<copy file="config/about.htm" todir="${dist}"/>
```

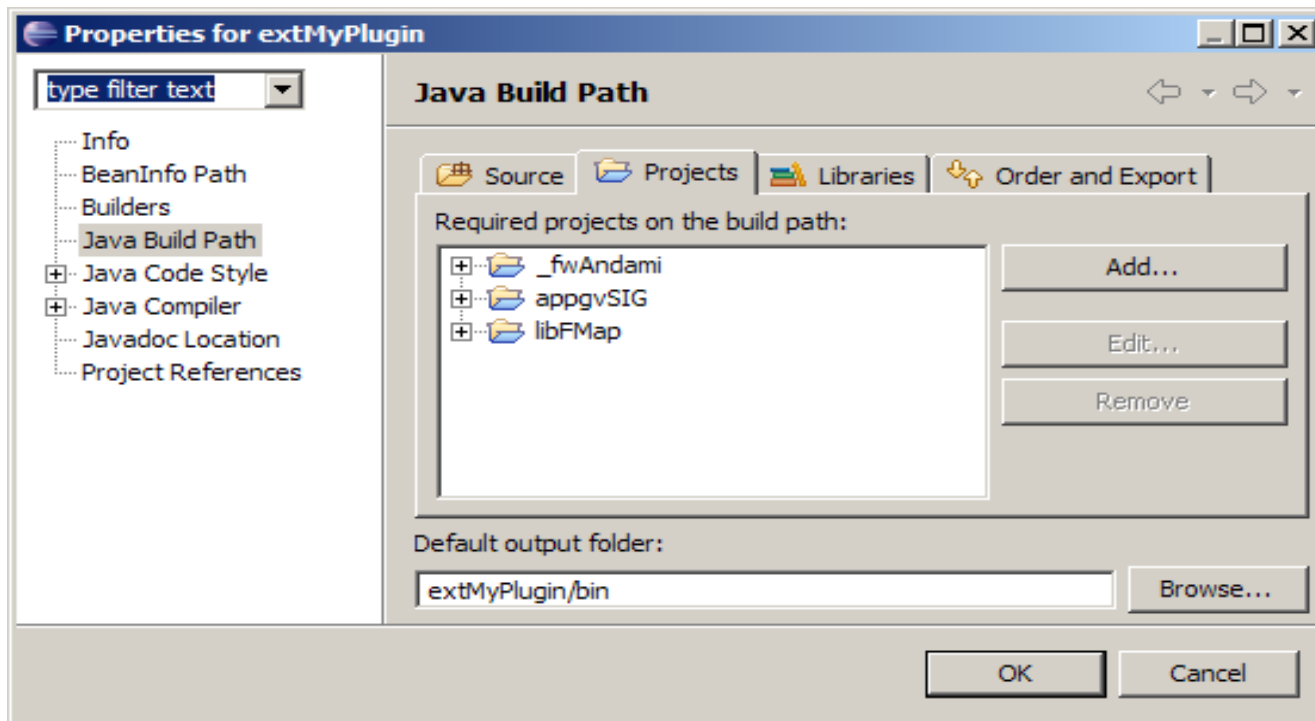
- Ejecutar el build

Comprobaciones

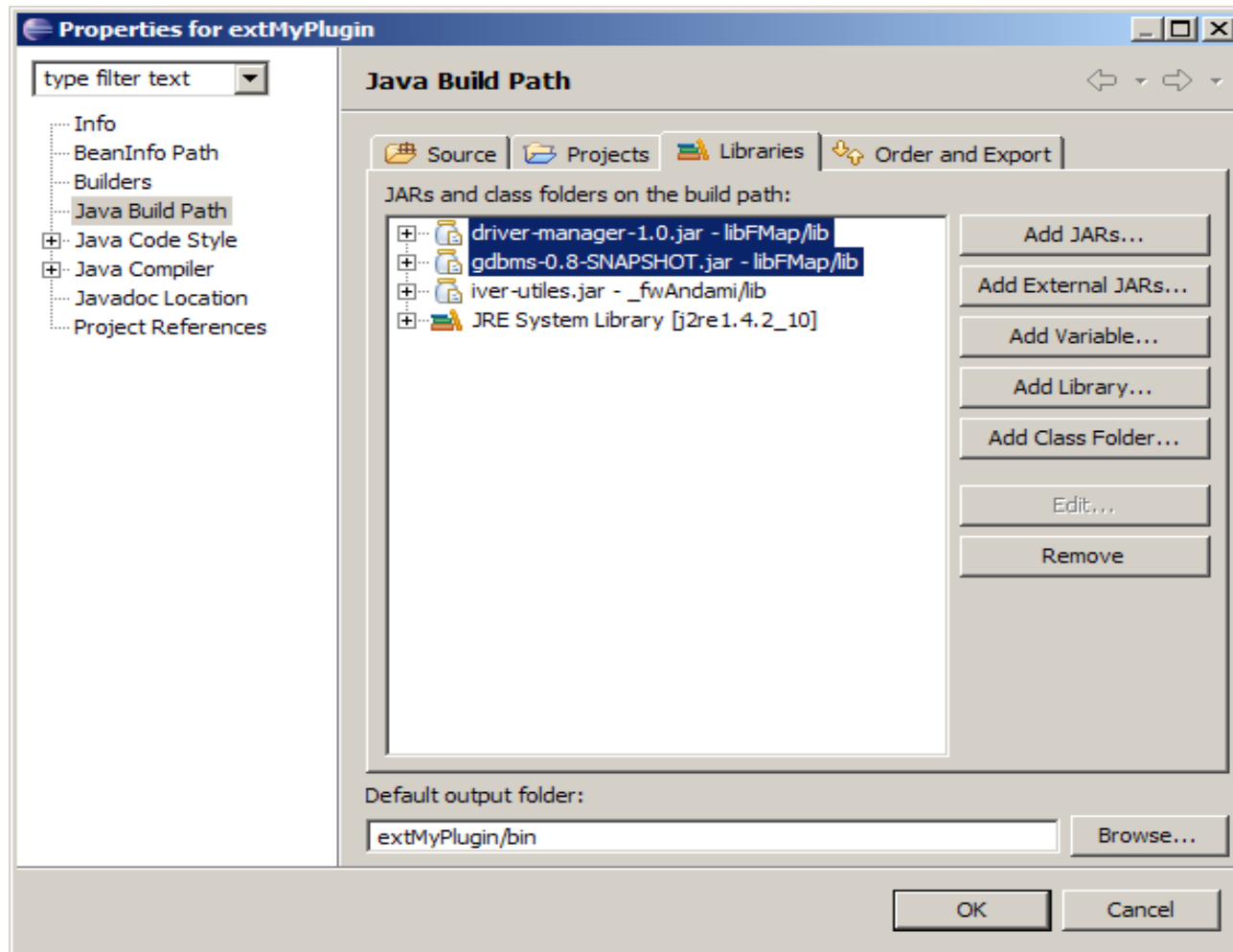
- Comprobar que en la carpeta “_fwAndami->gvSIG->extensiones” aparece esto



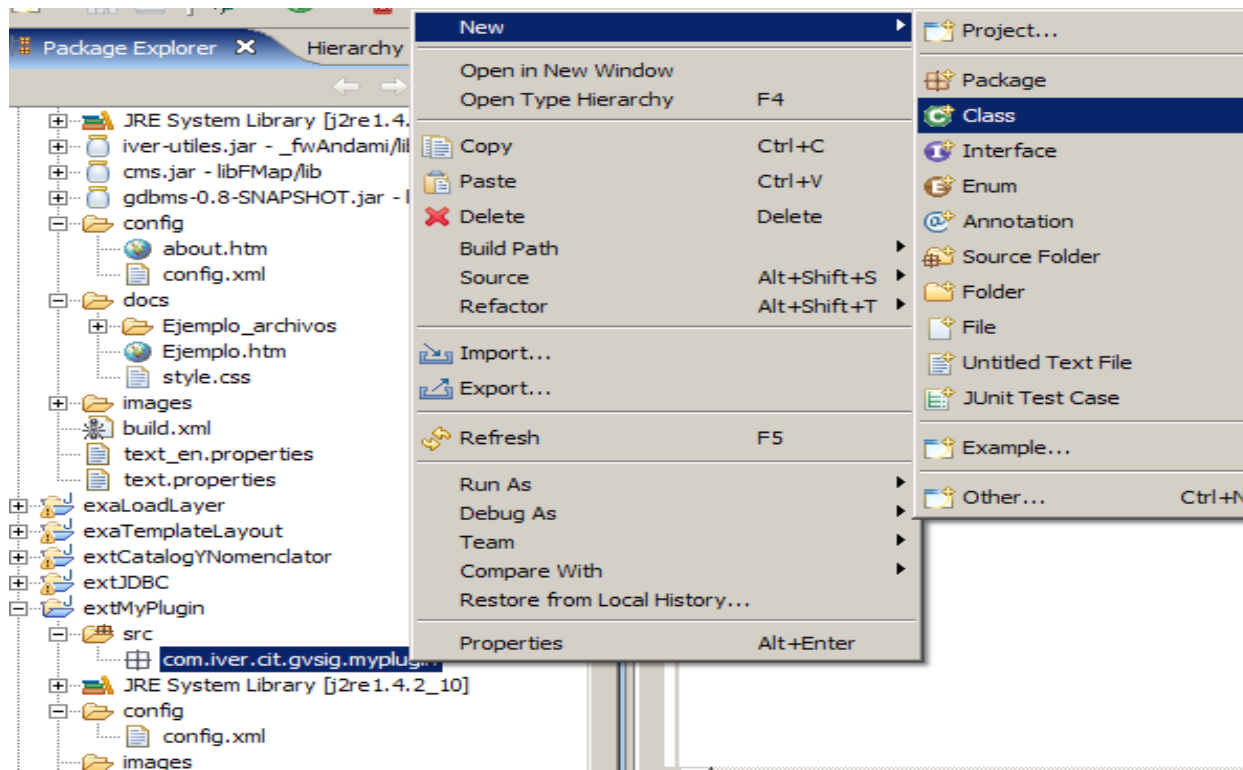
- Hay que indicarle al plugin los proyectos de los que depende
- Pinchamos sobre el proyecto con el botón derecho y seleccionamos la opción “Properties”
- Seleccionamos “Java Build Path”
- Añadimos los proyectos en “Projects”:



- En la pestaña “libraries”, añadimos las librerías:
 - `_fwAndami/lib/iver-utiles.jar`
 - `libFMap/lib/cms.jar`
 - `libFMap/lib/driver-manager-1.0.jar`
 - `libFMap/lib/gdbms-0.8-SNAPSHOT.jar`



- Vamos a crear nuestra primera extensión
- Para ello tenemos que crear una clase, que herede de la clase “Extension”
- La podemos llamar “LoadProvinciasExtension”



- Escribir el “extends Extension”
- Aparecerá una “bombilla amarilla” que indica un error. Pulsar el ella y elegir la opción “Import 'Extension' (com.iver.andami.plugins)”

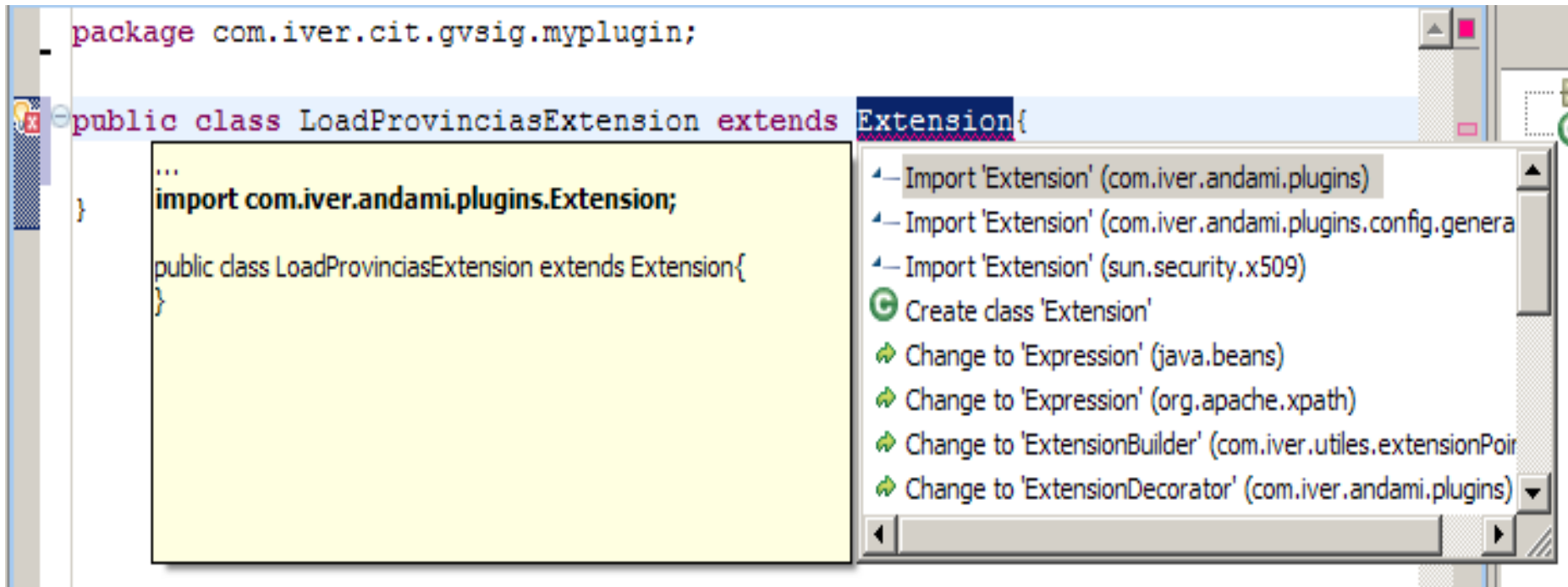
```

package com.iver.cit.gvsig.myplugin;

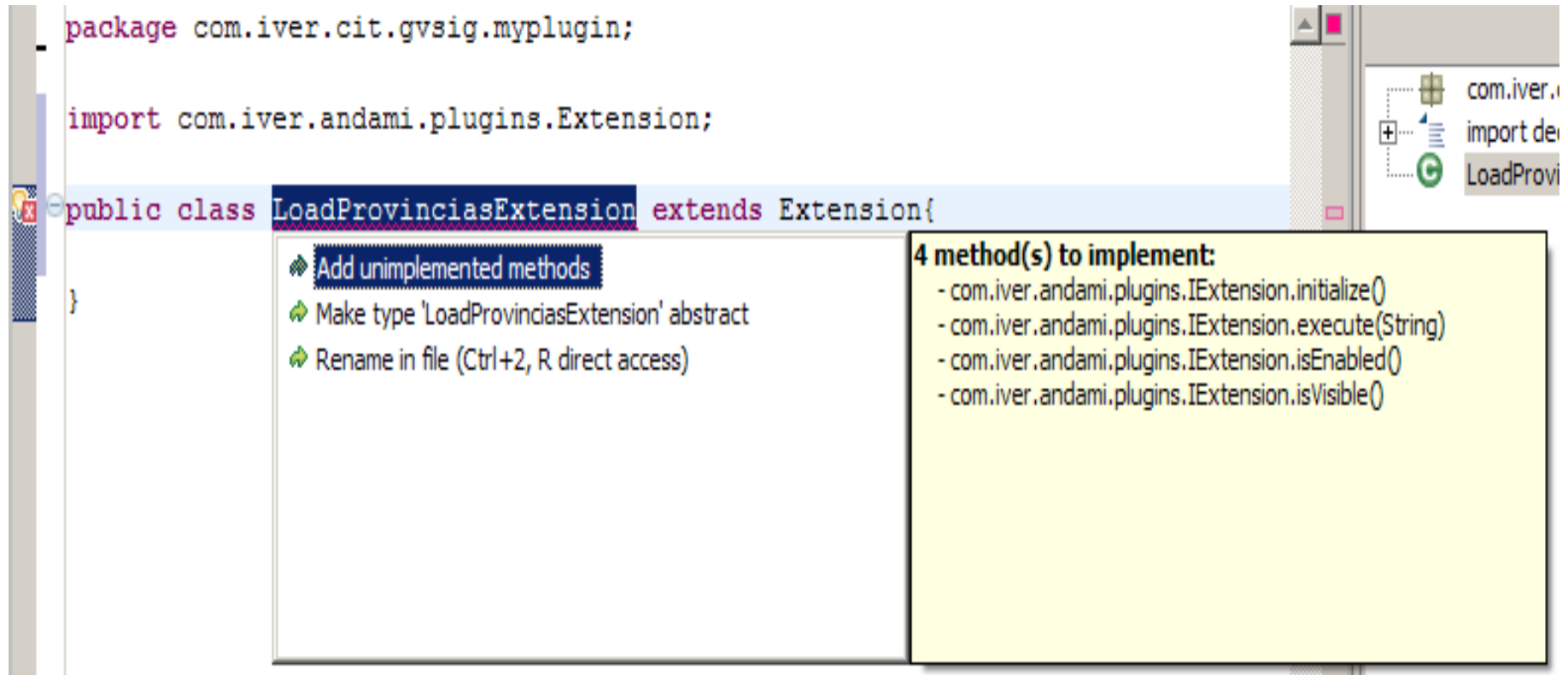
public class LoadProvinciasExtension extends Extension{
    ...
    import com.iver.andami.plugins.Extension;

    public class LoadProvinciasExtension extends Extension{
    }
}

```



- La “bombilla” sigue estando. Volver a pulsarla y seleccionar la opción “add unimplemented methods”



```

package com.iver.cit.gvsig.myplugin;

import com.iver.andami.plugins.Extension;

public class LoadProvinciasExtension extends Extension{
}
    
```

Add unimplemented methods

- Make type 'LoadProvinciasExtension' abstract
- Rename in file (Ctrl+2, R direct access)

4 method(s) to implement:

- com.iver.andami.plugins.IExtension.initialize()
- com.iver.andami.plugins.IExtension.execute(String)
- com.iver.andami.plugins.IExtension.isEnabled()
- com.iver.andami.plugins.IExtension.isVisible()

- Modificar los métodos `isEnabled` e `isVisible` para que se habilite la extensión y sea visible. Para ello tiene que devolver “true”.

```
public boolean isVisible() {  
    return true;  
}
```

```
public boolean isEnabled() {  
    return true;  
}
```

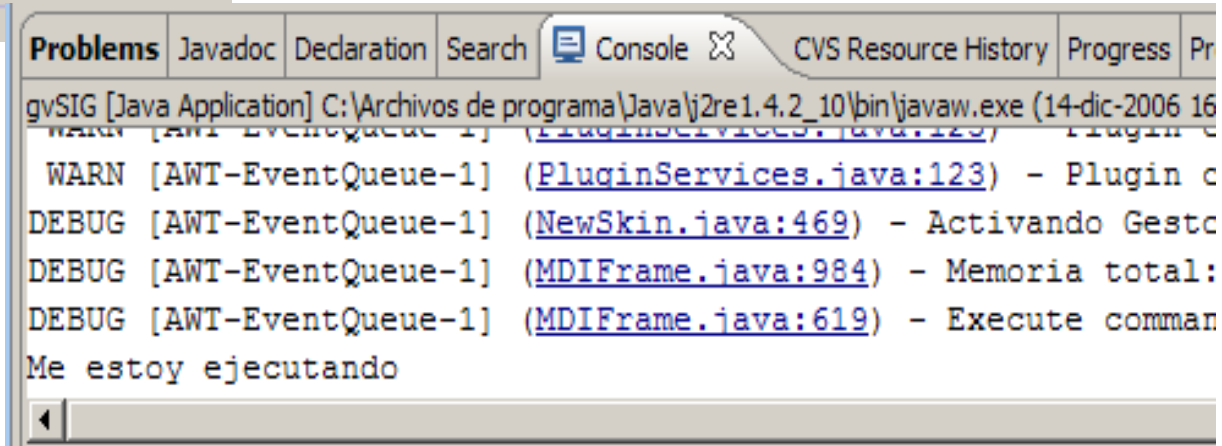
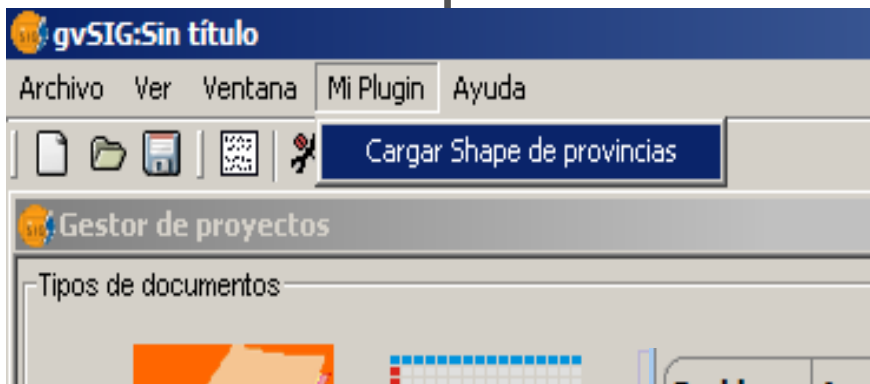
- Vamos a ponerle algo de código a la extensión para ver que se ejecuta
- Un `System.out.println("Me estoy ejecutando")` en el `execute` será suficiente (de momento)

```
public void execute(String actionCommand){  
    System.out.println  
        ("Me estoy ejecutando");  
}
```

- Hay que configurar el “config.xml” para asociarle una opción de menú a la extensión
- Copiamos el “config.xml” de la extensión exaExample1 para usarlo de plantilla.
- Creamos una única extensión:
 - El class-name= “com.iver.cit.gvsig.myplugin.LoadProvincias Extension”
 - La opción de menú es “Mi Plugin/Cargar Shape de provincias”
 - Poner las opciones de menú en castellano y en inglés

Primera ejecución

- Ya podemos ejecutar el “build.xml” y ejecutar gvSIG. Debería aparecernos la opción de menú que hemos añadido
- Al pulsar en ella veremos el mensaje que hemos escrito por la consola del eclipse





isEnabled

- Antes de incorporar el comportamiento para añadir una capa, debemos deshabilitar la opción del menú, para que solo se vea cuando haya una vista activa
- Tenemos que implementar el método isEnabled de la Extension

```
public boolean isEnabled() {  
    IWindow v = PluginServices.  
        getMDIManager().getActiveWindow(); if (v  
instanceof View){  
        return true;  
    }else{  
        return false;  
    }  
}
```

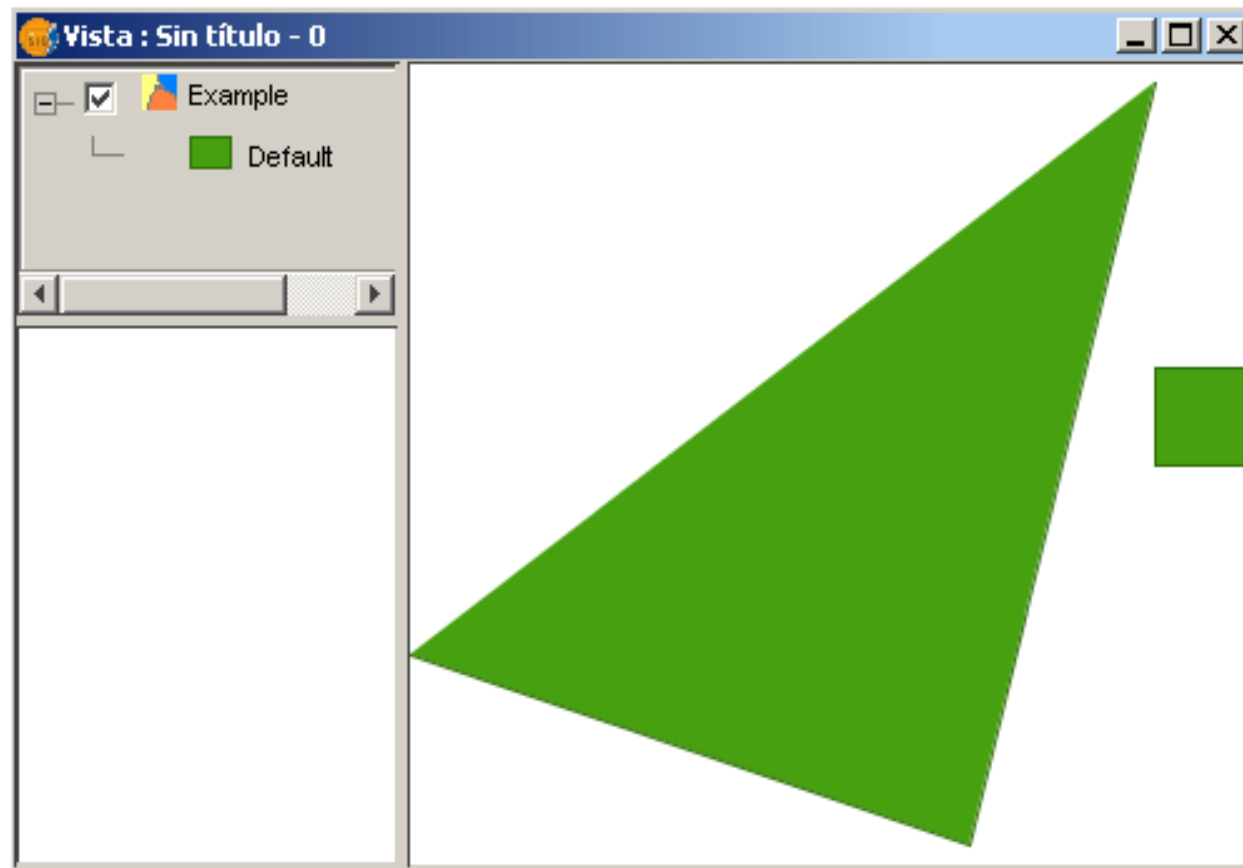


Cargar Shape



- Vamos a sustituir el código del “execute” por un código que cree una capa de provincias.
- Partir del código del “CreateMemoryLinesLayer” (copiar el execute) y crear un tema de polígonos que representarán las provincias.
- Usar “Fpoligon2D”, en lugar de “Fpolyline2D”
- Los polígonos se cierran solos
- Comprobar que la capa se carga

- Se pueden dibujar los polígonos todo lo completos que queramos



- Añadirle a los polígonos los atributos:
 - ID: Identificador de la provincia
 - NOMBRE: Nombre
 - POBLACIÓN: Número de habitantes

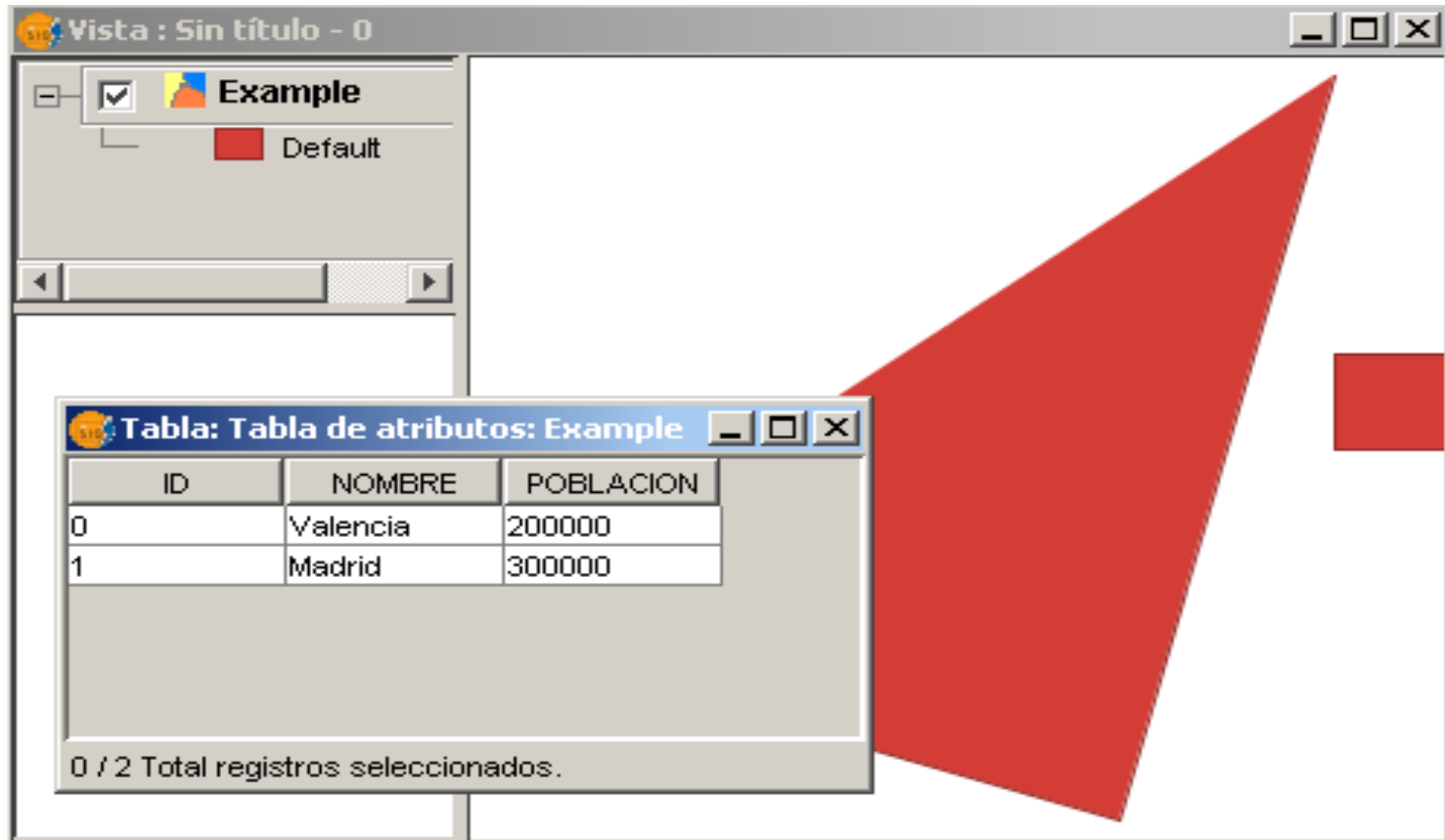
```
ArrayList arrayFields = new ArrayList();  
arrayFields.add("ID");  
arrayFields.add("NOMBRE");  
arrayFields.add("POBLACION");  
Value[] auxRow = new Value[3];
```

- Insertar los polígonos con sus respectivos atributos
- Para ello simplemente hay que darle los valores correctos a “auxRow” antes de introducir el polígono en la capa.

```
auxRow[0] = ValueFactory.createValue(0);  
auxRow[1] = ValueFactory.createValue("Valencia");  
auxRow[2] = ValueFactory.createValue(200000);
```

Comprobar que funciona

- Ejecutar el buid y abrir gvSIG para comprobar que funciona
- Ver la tabla asociada a la capa



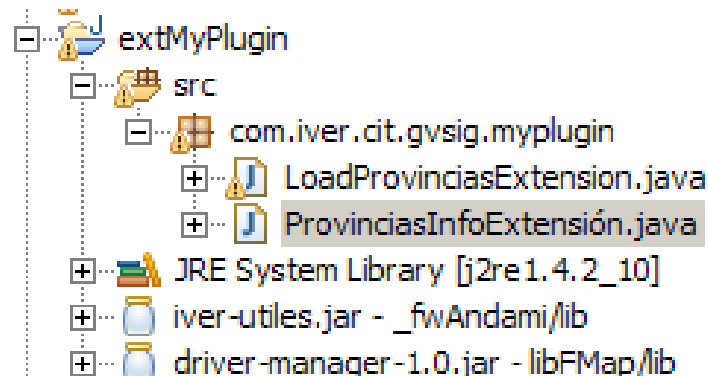
The screenshot shows the gvSIG interface. The main window is titled 'Vista : Sin título - 0'. On the left, there is a layer list with a checked box for 'Example' and a red square labeled 'Default'. The main map area displays a large red polygon and a smaller red square. A table window titled 'Tabla: Tabla de atributos: Example' is open, showing the following data:

ID	NOMBRE	POBLACION
0	Valencia	200000
1	Madrid	300000

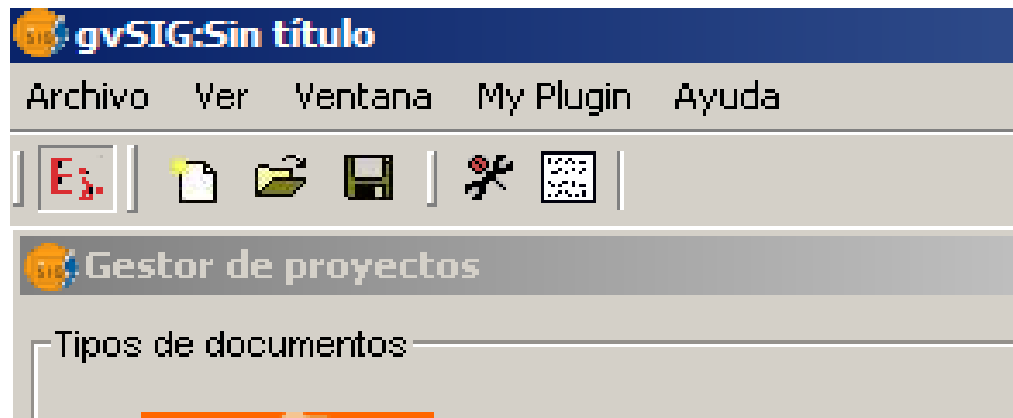
At the bottom of the table window, it says '0 / 2 Total registros seleccionados.'

Control sobre la vista

- Vamos a crear un control que nos mostrará en una ventana personalizada la información de una provincia al pulsar sobre ella
- Primero deberemos copiar la imagen “ejemplo.png” del proyecto “exaExample1” a la carpeta images de nuestro plugin
- A continuación deberemos crear otra extensión llamada “ProvinciasInfoExtensión”

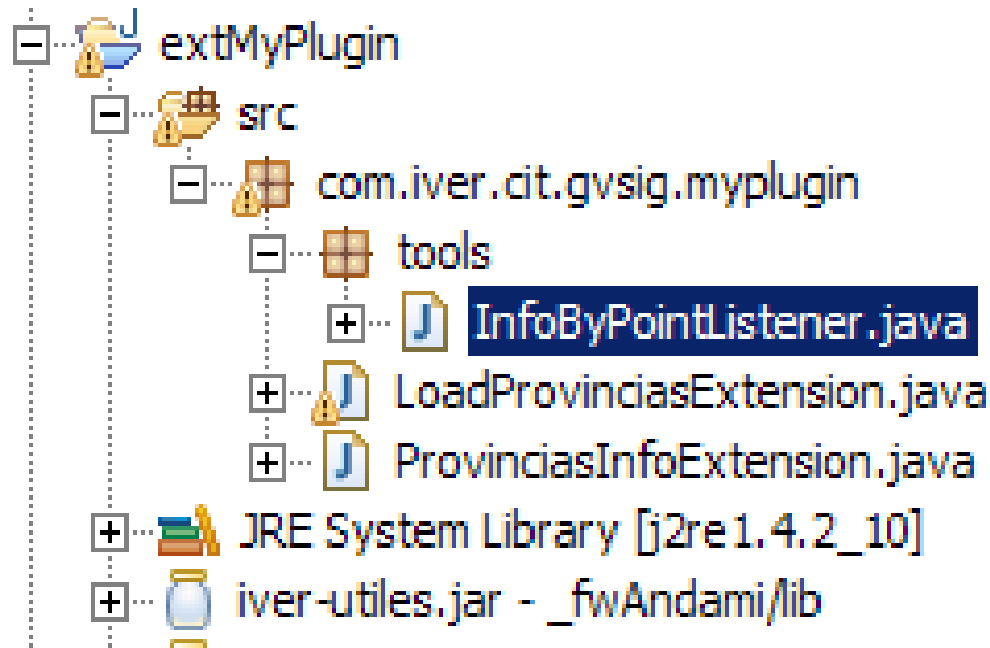


- Añadir en el config.xml otra entrada para la nueva extensión.
 - class-name= “com.iver.cit.gvsig.myplugin.ProvinciasInfoExtension”
 - Añadir un botón en la barra de herramientas: en lugar de “menu” hay que usar la etiqueta “toolbar”
 - Se pueden encontrar ejemplos de toolbars en los config's de los otros proyectos (p.e: appgvSIG)



InfoByPointListener

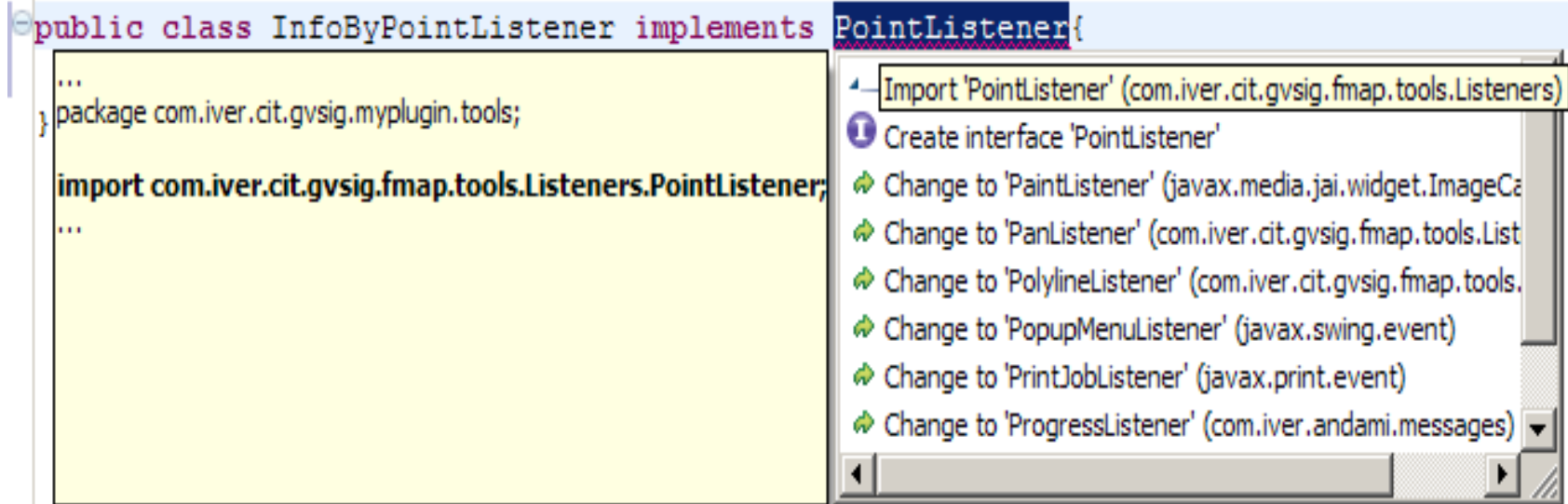
- Vamos a crear una herramienta que sea capaz de interactuar sobre la vista al hacer un “click” en ella
- Creamos el paquete “tools”
- Creamos la clase “InfoByPointListener”



InfoByPointListener

- El listener debe de implementar la interfaz “PointListener”
- Igual que antes, pulsando dos veces la “bombilla amarilla” solucionamos los problemas

```
public class InfoByPointListener implements PointListener{
    ...
    package com.iver.cit.gvsig.myplugin.tools;
    import com.iver.cit.gvsig.fmap.tools.Listeners.PointListener;
    ...
}
```



The screenshot shows an IDE window with a code editor on the left and a dropdown menu on the right. The code editor contains the following code:

```
public class InfoByPointListener implements PointListener{
    ...
    package com.iver.cit.gvsig.myplugin.tools;
    import com.iver.cit.gvsig.fmap.tools.Listeners.PointListener;
    ...
}
```

The dropdown menu on the right lists several suggestions for the interface to implement:

- Import 'PointListener' (com.iver.cit.gvsig.fmap.tools.Listeners)
- Create interface 'PointListener'
- Change to 'PaintListener' (javax.media.jai.widget.ImageCa)
- Change to 'PanListener' (com.iver.cit.gvsig.fmap.tools.List)
- Change to 'PolylineListener' (com.iver.cit.gvsig.fmap.tools.
- Change to 'PopupMenuListener' (javax.swing.event)
- Change to 'PrintJobListener' (javax.print.event)
- Change to 'ProgressListener' (com.iver.andami.messages)

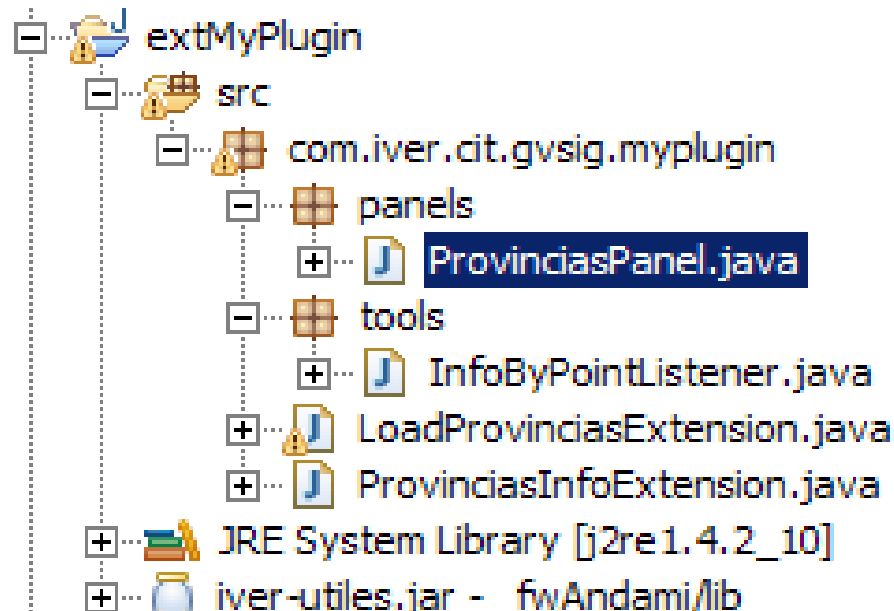
- Hay que añadirle un atributo de tipo MapControl al listener
- Hay que definir un constructor que tenga como argumento de entrada ese parámetro

```
private MapControl mapControl = null;
```

```
public InfoByPointListener(MapControl mapControl){  
    this.mapControl = mapControl;  
}
```


Ventana de Información

- Vamos a crear la ventana que va ha mostrarnos la información de las provincias
- Creamos el paquete “panels”
- Creamos dentro de él la clase “ProvinciasPanel”



- La ventana tiene que heredar de “Jpanel” e implementar la interfaz “IWindow”
- Un click en la “bombilla”, y los errores desaparecerán

```
public class ProvinciasPanel extends JPanel implements IWindow{
```

- ⚙ Add unimplemented methods
- + Add default serial version ID
- + Add generated serial version ID
- 🌱 Make type 'ProvinciasPanel' abstract
- 🌱 Rename in file (Ctrl+2, R direct access)

1 method(s) to implement:

```
- com.iver.andami.ui.mdiManager.IWindow.  
getWindowInfo()
```

blems Javadoc Deck

ninated> gvSIG [Java

RN [AWT-EventO

RN [AWT-EventO

RN [AWT-EventO

- Completar el getViewInfo con el código:

```
public WindowInfo getWindowInfo() {  
    WindowInfo m_viewInfo = new  
    WindowInfo(WindowInfo.MODALDIALOG);  
    m_viewInfo.setWidth(265);  
    m_viewInfo.setHeight(10);  
    m_viewInfo.setTitle(PluginServices.getText(this,  
        "title"));  
    return m_viewInfo;  
}
```

- La cadena “title” debe añadirse en los ficheros de traducciones
- El `setHeight` y el `setWidth` especifican las dimensiones que tendrá la ventana resultante
- `WindowInfo.MODALDIALOG` indica el comportamiento de la ventana

- Completar el constructor de la ventana para que dibuje un cuadro de texto donde aparecerá el nombre de la provincia seleccionada
- Cada uno puede crear una ventana más compleja



Ventana de Información

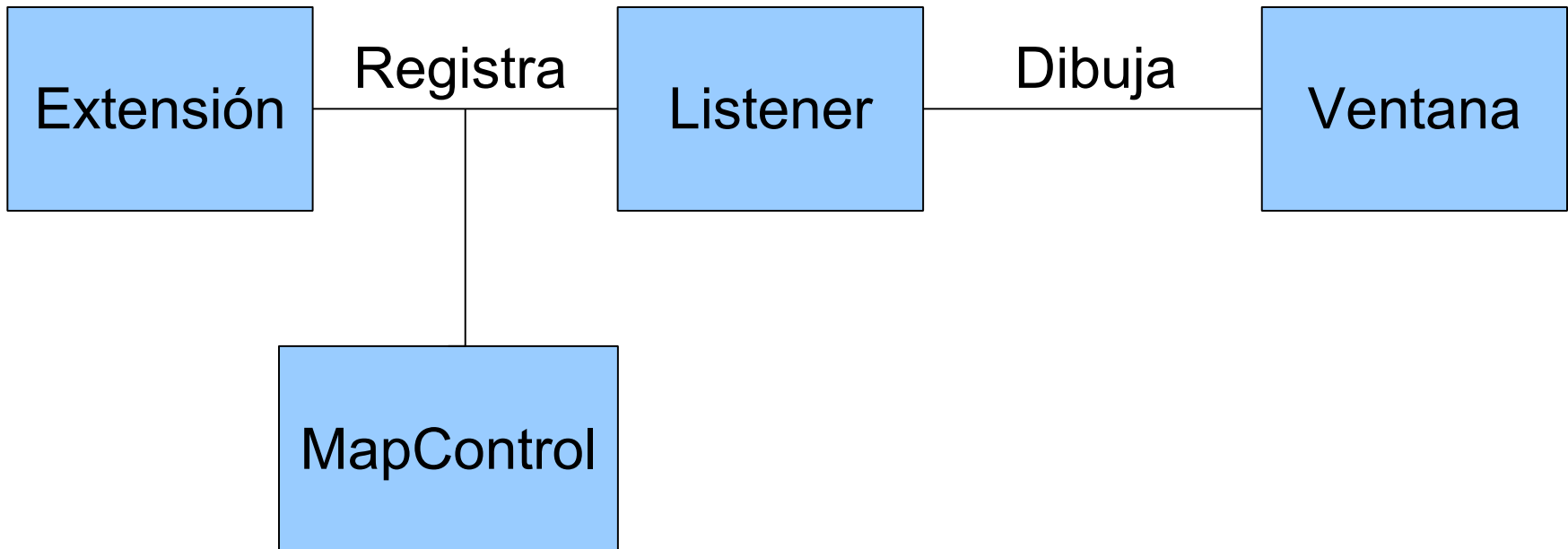


```
public ProvinciasPanel(String provinciasName){  
    super();  
    JTextField tfProvincias = new JTextField();  
    tfProvincias.setBounds(5,5,250,20);  
    tfProvincias.setText(provinciasName);  
    this.setLayout(null);  
    this.add(tfProvincias);  
}
```

- Componente que controla lo que se puede hacer sobre una vista
- Se le pueden añadir nuevas herramientas
 - En nuestro caso vamos a añadirle una herramienta para seleccionar geometrías
- Siempre tiene una herramienta seleccionada (zoom, desplazamiento, etc)

Unión de los componentes

- La extensión deberá crear un listener al ser ejecutada y le asociará la vista correspondiente
- El listener deberá ser capaz de abrir una ventana y escribir el nombre de la provincia





Unión Extensión-Listener



- Crear en la extensión un listener
- Obtener el MapControl y registrarlo
- Cargar la herramienta
- Implementar el isEnabled igual que antes



Unión Extensión-Listener



```
private InfoByPointListener
    listener = null;
public void execute(String actionCommand){
    View view = (View)PluginServices.
        getMDIManager().getActiveWindow();
    MapControl mc = view.getMapControl();
    if (listener == null){
        listener =
            new InfoByPointListener(mc);
        mc.addMapTool("provinciasInfo", new
            PointBehavior(listener));
    }
    mc.setTool("provinciasInfo");
}
```

```
public void point(PointEvent event) throws  
    BehaviorException {  
    Point2D pReal = event.getPoint();  
    Point2D mapPoint =  
        mapControl.getViewPort().  
        toMapPoint((int)pReal.getX(),  
        (int)pReal.getY());  
    System.out.println("El punto seleccionado es: " +  
        mapPoint.getX() + " " +  
        mapPoint.getY());  
}
```

- Vamos a comprobar que el listener funciona correctamente
- Para ello vamos a modificar el método point del InfoByPointListener para que muestre por la consola el punto de la vista “seleccionado”

```

Problems Javadoc Declaration Console X Search
<terminated> gvSIG [Java Application] C:\Archivos de programa\Java\j2re1.4.2_10\bin\javaw.exe (
El punto es46.059701492537314,111.01492537313433
El punto es69.70149253731344,82.26865671641792
El punto es44.17910447761194,120.95522388059702

```

- Tenemos que obtener una referencia a la capa de provincias sobre la que se ejecutará la consulta

```
double tol = mapControl.getViewPort().
    toMapDistance(1);
FLayers lyrs =
    mapControl.getMapContext().getLayers();
FLyrVect lyrProvincias =
    (FLyrVect)lyrs.getLayer("Example");
```

- Creamos la consulta


```
FBitSet selection = lyrProvincias.
    queryByPoint(mapPoint,tol);
```

Recuperar los atributos

- Recuperamos el ID, Nombre y la Población de las provincias

```

if (!selection.isEmpty()){
    DataSource ds = ((AlphanumericData)lyrProvincias).
        getRecordset();
    ds.start();
    int idField = ds.getFieldIndexByName("ID");
    int idNombre = ds.getFieldIndexByName("NOMBRE");
    int idPoblacion =
        ds.getFieldIndexByName("POBLACION");
    Value strID = ds.getFieldValue(
        selection.nextSetBit(0),idField);
    Value strNombre = ds.getFieldValue(
        selection.nextSetBit(0),idNombre);
    Value strPoblacion = ds.getFieldValue(
        selection.nextSetBit(0),idPoblacion);
    ds.stop();
    
```



Creamos la ventana de resultados



- Tenemos que crear una ventana de resultados y pasarle en nombre de la provincia
- Hay que introducir esta ventana en Andami

```
if (strNombre != null){  
    ProvinciasPanel panel = new  
        ProvinciasPanel(strNombre.toString());  
    PluginServices.getMdiManager().  
        addWindow(panel);  
}
```



Info by Point



The screenshot shows a GIS application window titled "Vista : Sin título - 0". The window contains a map area with a large olive-green polygon and a smaller square. A legend in the top-left corner lists "Example" with a multi-colored square and "Default" with an olive-green square. A pop-up window titled "Información por punto" is open, displaying the text "Valencia" in a text box.



Gracias por su atención