eclipse en el soporte al desarrollo de software

desarrollo de software

Abel Gómez Llana [agomez@dsic.upv.es].

15-19 de Septiembre 2008.

Universidad Politécnica de Valencia. Facultad de Informática.





Desarrollo avanzado de plug-ins

de plug-ins

18 de Septiembre de 2008







3

Contenido



- Creación de una nueva consola de eclipse.
- Creación de un editor sencillo con coloreado de sintaxis.
- Creación de una hoja de preferencias.
- Creación de un *plug-in* de ayuda.
- Creación de un punto de extensión.
- Creación de una perspectiva.







Creación de una consola de eclipse

eclipse





- Cuando eclipse se ejecuta en modo normal, las salidas estándar y de error no se muestran.
- Por tanto, ¿cuál es la interfaz estándar en Eclipse para mostrar mensajes textuales (información, depuración, etc.) al usuario? → Las vistas de consola.







- La API para el uso de una consola en Eclipse se encuentra definida en el *plug-in «org.eclipse.ui.console*».
- En primer lugar, hemos de crear una nueva, y registrarla en el gestor de consolas:

```
MessageConsole console = new MessageConsole(consoleName, null);
ConsolePlugin.getDefault().getConsoleManager().
    addConsoles(new IConsole[]{console});
```

- A continuación, debemos de crear un *stream* donde poder escribir los mensajes que se mostrarán por consola. Una consola puede tener múltiples *streams* donde escribir. MessageConsoleStream redStream = console.newMessageStream();
- Un stream puede tener asociado un determinado color, para poder diferencias distintos tipos de mensajes: redStream.setColor(Display.

getDefault().getSystemColor(SWT.COLOR_RED));







- Si estamos construyendo un conjunto de *plug-ins* que serán distribuidos de forma conjunta, puede resultar interesante crear un *plug-in* para que albergue la consola de nuestra aplicación.
- En este caso, crearemos un *plug-in* al que se acceda de la siguiente manera para registrar los mensajes en la consola:

ExampleConsoleUIPlugin.getDefault().printMessage("mensaje"); ExampleConsoleUIPlugin.getDefault().printError("mensaje");





18 Plug-in Development - es.example.console/src/es/example/console/internal/ExampleConsole.java - Eclipse SDK File Edit Source Refactor Navigate Search Project Run Window Help 前・回告日本・Q・Q・日告告 G・日息 / 日月 日、同、や ゆ・ウ・ 📳 🚸 Plug-in Devel... 😁 🗖 🗍 🗊 ExampleConsole.java 🔅 🕼 Package Explorer 💠 🛝 Plug-ins æ (+ + (-) | = (-) × package es.example.console.internal; 匙 □ □ les.example.console @import org.eclipse.swt.SWT;[] B-18 src ∃ ⊕ es.example.console public class ExampleConsole { B → B ExampleConsoleUPlugin.java ∃ Θ ExampleConsoleUtPlugin MessageConsole console; o^S plugin MessageConsoleStream stdout; ∜ PLUGIN_ID MessageConsoleStream stderr; @⁵ oetDefault() 4 console ExampleConsoleUIPlugin()
 public ExampleConsole() { Ø printError(String) console = new MessageConsole("Example console", null); Ø printMessage(String) Q. start(BundleContext) ConsolePlugin.getDefault().getConsoleManager() 0. stop(BundleContext) .addConsoles(new IConsole[]{console}); B-III es.example.consple.internal B-D ExampleConsole.java stdout = console.newMessageStream(); ∃ ⊕ ExampleConsole stderr = console.newMessageStream(); △ console 4 stderr stdout.setColor(Display.getDefault().getSystemColor(SWT.COLOR BLACX)); △ stdout stdout.setColor(Display.getDefault().getSystemColor(SWT.COLOR RED)); 0^E ExampleConsole() Ø getStderr() Ø getStdout() public MessageConsoleStream getStdout() { ③ 副, JRE System Library [jre 1.6.0_05] return stdout; B B. Plug-in Dependencies B-B-META-INF build.properties public MessageConsoleStream getStderr() { return stderr: 8 9 8 B

8





SXIO



- Eclipse proporciona una rica API (aunque compleja) para la creación de editores textuales: *JFace Text*.
- Esta API proporciona funciones para poder crear editores con coloreado de sintaxis, marcado de errores, autocompletado de código, etc.
- Eclipse proporciona un asistente que permite crear un nuevo *plug-in* que defina un nuevo editor de texto.
- Por defecto, este asistente crea un editor de texto para archivos XML.
- El ejemplo de creación de un editor de texto sencillo que mostraremos a continuación no hará uso de este asistente.





Uso del punto de extensión de un nuevo editor

un nuevo editor





 En primer lugar, crearemos un nuevo proyecto de plugin vacío.

New Plug-in Project		
Plug-in Project Create a new plug-in project		
Project name: es.example.e	editor	
Use default location		
Location: D:/Eclipse/eclipse-	SDK-3.3.2-win32/eclipse/workspace/es.example.	<u>B</u> rowse
Project Settings		
Create a Java project		
Source folder: src		
Target Platform This plug-in is targeted to rur ③ Eclipse version: ○ an OSGi framework	a with: 3.3 Equinox	
0	<back next=""> Einish</back>	Cancel

🖶 New Plug-in	Project				
Plug-in Content Enter the data required to generate the plug-in.					
Plug-in Properties					
Plug-in <u>I</u> D:	es.example.editor				
Plug-in Version:	1.0.0				
Plug-in Name:	Editor Plug-in				
Plug-in Provi <u>d</u> er:					
<u>C</u> lasspath:					
Plug-in Options					
✓ Generate an a	ctivator, a Java class that controls the plug-in's life cycle				
Ac <u>t</u> ivator: e	Activator: es.example.editor.ExampleEditorUIPlugin				
This plug-in wi	I make contributions to the UI				
Rich Client Application Would you like to create a rich dient application? O Yes					
?	< <u>B</u> ack <u>N</u> ext > <u>Finish</u>	Cancel			





 En primer lugar, crearemos un nuevo proyecto de plugin vacío.









 A continuación, registraremos nuestro editor de texto (aún sin funcionalidad) invocando el punto de extensión «org.eclipse.ui.editors».

		Extension Point Selection	
as.example.editor 🛛		Create a new Internal and External Editors extension.	=)
San Extensions			
		Extension Points Extension Wizards	
All Extensions		Extension Doint filter:	
Define extensions for this plug-in in the following section.			
type filter text		org.eclipse.ui.editorActions forg.eclipse.ui.editors	
8	10		
E	lit		
	qL		
		Show only extension points from the required plug-ins	
	N	Extension Point Description: Internal and External Editors	
		This extension point is used to add new editors to the workbench. A editor is a v	sual 🔺
		component within a workbench page. It is typically used to edit or browse a docu	iment or
		input object. To open an editor, the user will typically invoke "Open" on an IFile action is performed the workbench registry is consulted to determine an approp	. When this riate editor
	•	for the file type and then a new instance of the editor type is created. The actual	esult 🗸
		Available templates for internal and external editors:	
		the Editor	
		ad [≜] Multi-page Editor	
		Image: Constraint of the second sec	Cancel
Overview Dependencies Runtime Extensions Extension Po	ats Build MANIFEST ME build properties plugin ym		
Extensional Extensional Extensional Extension Pol	na paila marka contrai pailatpi operaco piagintxini		





<extension

```
point="org.eclipse.ui.editors">
      <editor
         class="org.eclipse.ui.editors.text.TextEditor"
         contributorClass="org.eclipse.ui.editors.text.TextEditorActionContributor"
         extensions="text"
                                                                                                                                                         🕼 es.example.editor 🔀
         icon="icons/icon.gif"
                                                         Sale Extensions
         id="es.example.editor.editor1"
                                                                                                                                                          (?)
         name="ExampleEditor">
                                                          All Extensions
                                                                                                              Extension Element Details
                                                                                                  La E
    </editor>
                                                           Define extensions for this plug-in in the following section.
                                                                                                              Set the properties of "editor". Required fields are denoted by
                                                                                                              ***.
</extension>
                                                            type filter text
                                                                                                              id*:
                                                                                                                              es.example.editor.editor1
                                                                                                  <u>A</u>dd...
                                                             Image: org.eclipse.ui.editors
                                                                                                                              ExampleEditor
                                                                                                              name*:
                                                                  ExampleEditor (editor)
                                                                                                  Edit...
                                                                                                                              icons/eclipse_icon.gif
                                                                                                                                                   Browse ...
                                                                                                              icon:
                                                                                                                              text
                                                                                                              extensions:
                                                                                                   Up
                                                                                                                              brg.eclipse.ui.editors.text
                                                                                                                                                   Browse...
                                                                                                              class:
                                                                                                  Down
                                                                                                              command:
                                                                                                              launcher:
                                                                                                                                                   Browse...
                                                                                                              contributorClass:
                                                                                                                              org.edipse.ui.editors.text
                                                                                                                                                   Browse...
                                                                                                              default:
                                                                                                              filenames:
                                                                                                              symbolicFontName:
                                                                                                              matchingStrategy:
                                                                                                                                                   Browse...
```

Overview Dependencies Runtime Extensions Extension Points Build MANIFEST.MF plugin.xml build.properties





 El plug-in que hemos creado asocia el editor de texto estándar de eclipse (org.eclipse.ui.editors.text.TextEditor) y el icono seleccionado a los archivos con extensión «*.text».

🖶 Java - Example/file.text - Eclipse SDK		- D×
<u>File Edit Navigate</u> Se <u>a</u> rch <u>Project Run Wi</u>	indow <u>H</u> elp	
	: 岱 🕆 ⓒ 🔹 🤔 🔗 🗈 🐉 Java)
Image: Package Explor X Image: Hierarchy Image: Display ↓ ↓ ↓ ↓ ↓	■ file.text 🛛	
Example file.text		
		×
i □ [⇔] Writable	Insert	@





Creación de la implementación personalizada

personalizada







- Eclipse emplea el modelo *daño, reparación, reconciliación (damage, repair, reconcile)*.
- Cada vez que el usuario cambia el documento, el reconciliador determina qué región ha sido invalidada, y cómo debe repararse:
 - *Daño* es el texto que debe redibujarse.
 - *Reparación* es el método empleado para redibujar el área dañada.
 - *Reconciliación* es el proceso de mantener la representación visual de un documento cada vez que se efectúa un cambio.
- Para personalizar el modelo de daño/reparación de nuestro editor, extenderemos la clase por defecto *TextEditor*, y la modificaremos para que emplee nuestra propia clase *SourceViewerConfiguration*.





- La clase *SourceViewerConfiguration* el clase central para configurar el editor con funcionalidad adicional (p.e., coloreado de sintaxis o autocompletado).
- Por defecto, esta clase no soporta coloreado de sintaxis, por ello, crearemos nuestra propia clase SourceViewer que herede de SourceViewerConfiguration, pudiendo sobreescribir los métodos que nos interesen.
- Nuestro SourceViewer hará uso de sus propias reglas para determinar las distintas partes del documento, y cómo deberán representarse.
- Podremos definir nuestras propias reglas de forma sencilla extendiendo la funcionalidad ofrecida por la clase *RuleBasedScanner*.









- De esta manera, nuestro plug-in requerirá de tres clases adicionales:
 - ExampleEditor Extiende la clase por defecto TextEditor. Cambiaremos el SourceViewer por defecto.
 - ExampleRuleScanner Hereda de RuleBasedScanner y define las reglas para nuestro propio editor que se emplearán en la clase de daño/reparación.
 - ExampleSourceViewerConfig Hereda de SourceViewerConfiguration, instanciará la regla de nuestro editor, y definirá la acción de reconciliación.







• En primer lugar, debemos configurar nuestro editor para que emplee nuestro *SourceViewer*:









 A continuación, debemos establecer el modelo damage/repair/reconcile sobreescribiendo el método SourceViewerConfiguration.getPresentationReconciler():







 Por último, definimos las reglas que marcarán como se coloreará el texto de nuestro editor. En este caso, definiremos un color azul para algunas palabras clave de SQL, y color verde para comentarios:





23



En este punto, podremos ejecutar nuestro editor (debemos recordar actualizar el archivo de manifiesto, para apuntar a la nueva clase del editor):

	plo.text 🛛	
-	Nombre del esquema: Model	
-	Creado en: 08/01/2008 13:16	
-		
-	Creacion de tablas	
RC	P TABLE "Model"."Solicitud";	
RE	ATE TABLE "Model"."Solicitud"	
	"fk Identificador Acta" VARCHAR NOT NULL,	
	"fk Identificador Alumno" VARCHAR NOT NULL,	
	"fk Identificador Convocatoria" VARCHAR NOT NULL,	
	"fk Identificador Examen" VARCHAR NOT NULL,	
	"Identificador Solicitud" VARCHAR NOT NULL,	
	"Están Fotos Disponibles" VARCHAR(1) NOT NULL,	
	"Está Fotocopia DNI Disponible" VARCHAR(1) NOT NULL,	
	"Está Certificado Médico Disponible" VARCHAR(1) NOT NULL,	
	"fk Identificador Resolución" VARCHAR NOT NULL.	
	PRIMARY KEY("fk Identificador Convocatoria", "fk Identificador Examen", "Identificador So	Li
;	PRIMARY KEY("fk_Identificador_Convocatoria", "fk_Identificador_Examen", "Identificador_So:	Li
;	<pre>FRIMARY KEY("fk_Identificador_Convocatoria", "fk_Identificador_Examen", "Identificador_So</pre>	Li
;	<pre>PRIMARY KEY("fk_Identificador_Convocatoria", "fk_Identificador_Examen", "Identificador_So: PP TABLE "Model"."Resolución";</pre>	Li
; ORC	PRIMARY KEY("fk_Identificador_Convocatoria", "fk_Identificador_Examen", "Identificador_So: P TABLE "Model"."Resolución"; ATE TABLE "Model"."Resolución"	Li
; RE	PRIMARY KEY("fk_Identificador_Convocatoria", "fk_Identificador_Examen", "Identificador_So: PF TABLE "Model"."Resolución"; ATE TABLE "Model"."Resolución"	Li
; RE	PRIMARY KEY("fk_Identificador_Convocatoria", "fk_Identificador_Examen", "Identificador_So: P TABLE "Model"."Resolución"; ATE TABLE "Model"."Resolución" "fk Identificador Acta" VARCHAR NOT NULL,	Li
; RE	PRIMARY KEY("fk_Identificador_Convocatoria", "fk_Identificador_Examen", "Identificador_So: PTABLE "Model"."Resolución"; ATE TABLE "Model"."Resolución" "Ik_Identificador_Acta" VARCHAR NOT NULL, "fk Identificador Convocatoria" VARCHAR NOT NULL,	Li
; RE	PRIMARY KEY("fk_Identificador_Convocatoria", "fk_Identificador_Examen", "Identificador_So: P TABLE "Model"."Resolución"; ATE TABLE "Model"."Resolución" "fk_Identificador_Acta" VARCHAR NOT NULL, "fk_Identificador_Convocatoria" VARCHAR NOT NULL, "Identificador Resolución" VARCHAR NOT NULL,	1.1
; RC	PRIMARY KEY("fk_Identificador_Convocatoria", "fk_Identificador_Examen", "Identificador_So: PF TABLE "Model"."Resolución"; "ATE TABLE "Model"."Resolución" "rk_Identificador_Acta" VARCHAR NOT NULL, "fk_Identificador_Convocatoria" VARCHAR NOT NULL, "Identificador_Resolución" VARCHAR NOT NULL, "Identificador_Resolución" VARCHAR NOT NULL,	Li
; RE	PRIMARY KEY("fk_Identificador_Convocatoria", "fk_Identificador_Examen", "Identificador_So: PTABLE "Model"."Resolución"; ATE TABLE "Model"."Resolución" "IK_Identificador_Acta" VARCHAR NOT NULL, "fk_Identificador_Convocatoria" VARCHAR NOT NULL, "Identificador_Resolución" VARCHAR NOT NULL, "Nota" VARCHAR NOT NULL, "Es Convalidación" VARCHAR(1) NOT NULL,	Li
RE	PRIMARY KEY("fk_Identificador_Convocatoria", "fk_Identificador_Examen", "Identificador_So: PTABLE "Model"."Resolución"; ATE TABLE "Model"."Resolución" "fk_Identificador_Acta" VARCHAR NOT NULL, "fk_Identificador_Convocatoria" VARCHAR NOT NULL, "Identificador_Resolución" VARCHAR NOT NULL, "Nota" VARCHAR NOT NULL, "Ba Nota" VARCHAR(1) NOT NULL, "Es Nota" VARCHAR(1) NOT NULL,	Li
; RE	PRIMARY KEY("fk_Identificador_Convocatoria", "fk_Identificador_Examen", "Identificador_So: PTABLE "Model"."Resolución"; ATE TABLE "Model"."Resolución" "fk_Identificador_Acta" VARCHAR NOT NULL, "fk_Identificador_Convocatoria" VARCHAR NOT NULL, "Identificador_Resolución" VARCHAR NOT NULL, "Nota" VARCHAR NOT NULL, "Es_Convalidación" VARCHAR(1) NOT NULL, "Es_Convalidación" VARCHAR(1) NOT NULL, "Es_Nota" VARCHAR(1) NOT NULL, "Es_Nota" VARCHAR(1) NOT NULL,	
; RC	PRIMARY KEY("fk_Identificador_Convocatoria", "fk_Identificador_Examen", "Identificador_So: PTABLE "Model"."Resolución"; ATE TABLE "Model"."Resolución" "fk_Identificador_Acta" VARCHAR NOT NULL, "fk_Identificador_Convocatoria" VARCHAR NOT NULL, "Identificador_Resolución" VARCHAR NOT NULL, "Nota" VARCHAR NOT NULL, "Sc_Convalidación" VARCHAR(1) NOT NULL, "Es_Convalidación" VARCHAR(1) NOT NULL, "Es_Nota" VARCHAR(1) NOT NULL, "FK_Identificador_Convocatoria", "Identificador_Reso:	13
; CRE	PRIMARY KEY("fk_Identificador_Convocatoria", "fk_Identificador_Examen", "Identificador_So: PTABLE "Model"."Resolución"; ATE TABLE "Model"."Resolución" "fk_Identificador_Acta" VARCHAR NOT NULL, "fk_Identificador_Convocatoria" VARCHAR NOT NULL, "Identificador_Resolución" VARCHAR NOT NULL, "Nota" VARCHAR NOT NULL, "Nota" VARCHAR NOT NULL, "Es_Convalidación" VARCHAR(1) NOT NULL, "Es_Nota" VARCHAR(1) NOT NULL, "Es_Nota" VARCHAR(1) NOT NULL, "Es_Nota" VARCHAR(1) NOT NULL, "Es_Nota" VARCHAR(1) NOT NULL, PRIMARY KEY("fk_Identificador_Acta", "fk_Identificador_Convocatoria", "Identificador_Resol	L 3
; DRC CRE (<pre>PRIMARY KEY("fk_Identificador_Convocatoria", "fk_Identificador_Examen", "Identificador_So: PTABLE "Model"."Resolución"; ATE TABLE "Model"."Resolución" "rk_Identificador_Acta" VARCHAR NOT NULL, "fk_Identificador_Convocatoria" VARCHAR NOT NULL, "Identificador_Resolución" VARCHAR NOT NULL, "Nota" VARCHAR NOT NULL, "Es_Convalidación" VARCHAR(1) NOT NULL, "Es_Nota" VARCHAR(1) NOT NULL, PRIMARY KEY("fk_Identificador_Acta", "fk_Identificador_Convocatoria", "Identificador_Resolución" PTABLE "Model"."Examen"; PT TABLE "Model"."Examen";</pre>	
); DRC CRE (); DRC CRE	PRIMARY KEY("fk_Identificador_Convocatoria", "fk_Identificador_Examen", "Identificador_So: PF TABLE "Model"."Resolución"; ATE TABLE "Model"."Resolución" "fk_Identificador_Acta" VARCHAR NOT NULL, "fk_Identificador_Convocatoria" VARCHAR NOT NULL, "Identificador_Resolución" VARCHAR NOT NULL, "Identificador_Resolución" VARCHAR NOT NULL, "Nota" VARCHAR NOT NULL, "Es_Convalidación" VARCHAR(1) NOT NULL, PES_Nota" VARCHAR(1) NOT NULL, PFIMARY KEY("fk_Identificador_Acta", "fk_Identificador_Convocatoria", "Identificador_Reso: PF TABLE "Model"."Examen"; ATE TABLE "Model"."Examen"	Li
); CRE (); DRC CRE (PRIMARY KEY("fk_Identificador_Convocatoria", "fk_Identificador_Examen", "Identificador_So: PTABLE "Model"."Resolución"; ATE TABLE "Model"."Resolución" "fk_Identificador_Acta" VARCHAR NOT NULL, "fk_Identificador_Convocatoria" VARCHAR NOT NULL, "Identificador_Resolución" VARCHAR NOT NULL, "Nota" VARCHAR NOT NULL, "Be_Convalidación" VARCHAR(1) NOT NULL, "Es_Nota" VARCHAR(1) NOT NULL, PRIMARY KEY("fk_Identificador_Acta", "fk_Identificador_Convocatoria", "Identificador_Resol PTABLE "Model"."Examen"; ATE TABLE "Model"."Examen"	Lu
); DRG CRE (); DRG CRE (PRIMARY KEY("fk_Identificador_Convocatoria", "fk_Identificador_Examen", "Identificador_So: PTABLE "Model"."Resolución"; "ATE TABLE "Model"."Resolución" "fk_Identificador_Acta" VARCHAR NOT NULL, "fk Identificador_Convocatoria" VARCHAR NOT NULL, "Identificador_Resolución" VARCHAR NOT NULL, "Nota" VARCHAR NOT NULL, "Es_Convalidación" VARCHAR(1) NOT NULL, "Es_Nota" VARCHAR(1) NOT NULL, "Es_Nota" VARCHAR(1) NOT NULL, "Fs_Nota" VARCHAR(1) NOT NULL, "Fs_Nota" VARCHAR(1) NOT NULL, "Fs_Model"."Examen"; "ATE TABLE "Model"."Examen"; "fk_Identificador_Asignatura_para_Titulación" VARCHAR NOT NULL, "The Interview of the Inter	Lu







preferencias





- Cada *plug-in*, dispone de un elemento *IPreferenceStore*, al que se accede mediante el método *AbstractUIPlugin.getPreferenceStore()*.
- IPreferenceStore es una interfaz que representa una tabla que mapea un valor (boolean, double, float, int, long, String) a una determinada clave (String).
- Una hoja de preferencias es la interfaz por defecto que Eclipse proporciona para que el desarrollador establezca los mecanismos de modificación de los valores del elemento *IPreferenceStore*, y que el usuario pueda establecer sus preferencias
- Como para los principales tipos de *plug-ins* Eclipse proporciona un asistente para crear una hoja de preferencias.
- En este caso, vamos a crear una hoja de preferencias que nos permita configurar la consola creada anteriormente.
- De esta manera, el usuario podrá determinar qué tipos de mensajes desea que se muestren por la consola (la consola muestra mensajes normales, de *debug* y de error).





- En este caso, puesto que las preferencias son de la propia consola, definiremos las clases para la hoja de preferencias dentro del mismo *plug-in* de la consola.
- Para invocar un asistente para añadir funcionalidad a un *plug-in* existente, deberemos seleccionar el botón «Add...» en la pestaña «Extensions» en el editor del archivo de manifiesto de un *plug-in*.

La, E

\dd.

Up Down

*es.example.console

*
Extensions

section.

type filter text

Define extensions for this plug-in in the following

Overview Dependencies Runtime Extensions Extension Points Bui

existence,		
botón	Extension Points Extension Wizards	
	Extension Templates	sê "Hello, World" action set
tensions»		☆ "Hello, World" command contribution
		₀≜Editor
		⊲≙File Import Wizard
		₀≙Help Content
		₀ icon Decorator
		⊲c≜Multi-page Editor
		₀≙New File Wizard
		₀≙Popup Menu
9		
		⊕ Project Builder and Nature
		₀≙Property Page
		⊕ Release Engineering Perspective
		⊲c≜Sample View
		^∲ Snlach Handler
	This template creates a page that is contribut common preference fields and how to save an select Window then Preferences from the	ed to the Preterences. It demonstrates how to create di restore values between invocations. To see the result, e menu bar.
	0	< <u>Back</u> <u>Next</u> Einish Cancel
d MANIFEST.MF build.properties plugin.	sml	

New Extension

Extension Point Selection

Create a new extension from the Preference Page template



27

Eclipse en el soporte al desarrollo de software. Universidad Politécnica de Valencia. Abel Gómez [agomez@dsic.upv.es]



 Para la creación de una hoja de preferencias, basta con indicar el nombre de la clase para la nueva hoja de preferencias, el nombre del paquete donde se colocará, y el nombre de la hoja de preferencias.

Preference Page Sample Preference Page The provided options allow you to control the preferences gava Package Name: es.example.console.preferences Page Name: ConsolePreferencePage Page Name: Example Console Page Name: Page Nam				Fill Flockage Explorer to Epstrag ins
Sample Preference Page The provided options allow you to control the preferences page that will be created. gava Package Name: es.example.console.preferences Page Name: Example Console Page Name: Example Console <th></th> <th></th> <th></th> <th>(> -> @ □ \$ ▼</th>				(> -> @ □ \$ ▼
Sample Preference Page The provided options allow you to control the preference page that will be created. Java Package Name: es.example.console.preferences Page Class Name: ConsolePreferencePage Page Name: Example Console Example Console Page Name: Example Console Page Name: Example Console Example Console Page Name: Example	🖶 Preference Pag	ge		🖃 🖅 es.example.console
Java Package Name: es.example.console.preferences Page Class Name: ConsolePreferencePage Page Name: Example Console Page Name: Page Name: Page Name: Page Name: Page Name: Page Name:	Sample Preferent	ce Page allow you to control the preference page that will be created.	=)	· · · · · · · · · · · · · · · · · · ·
	Java Package Name: Page Class Name: Page <u>N</u> ame:	es.example.console.preferences ConsolePreferencePage Example Console kext.solution.com	Einish Cancel	 e.example.console.internal ExampleConsole.java Example.console.preferences ConsolePreferencePage.java PreferenceConstants.java PreferenceInitializer.java JRE System Library [jre 1.6.0_05] Plug-in Dependencies META-INF MANIFEST.MF build,properties plugin.xml e.example.editor







- Por defecto, el asistente crea tres clases:
 - ConsolePreferencePage Es la clase que implementa la hoja de propiedades. La clase creada por defecto hereda de FieldEditorPreferencePage, que es un tipo especial de hoja de propiedades que:
 - Proporciona constructores sencillos para crear los controles SWT más comúnmente usados en hojas de propiedades.
 - Proporciona una implementación para guardar/recuperar los valores de estos controles al presionar los botones de *aplicar* y *aceptar*.
 - PreferenceConstants— En esta clase se establecen las constantes (Strings), que determinan las claves empleadas en el elemento IPreferenceStore.
 - PreferenceInitializer Es la clase que determina los valores por defecto de IPreferenceStore. Estos se establecen la primera vez que se ejecuta el plug-in, o cuando el usuario pulsa el botón «Restore Defaults».







 Si ejecutamos el *plug-in* con la implementación por defecto, tendremos la siguiente hoja de ejemplo:

e Preferences				- ox
type filter text	Example Console			\$ • \$ -
 General Ant Ecore Diagram EMFT JET Transformations Example Console Help Icon Style Install/Update Java Model Validation Plug-in Development Run/Debug Team UML Cass Diagram UML Composite Structures Diag UML Profile Definition Diagram UML Statemachine Diagram UML Use Case Diagram 	A demonstration of a p Directory preference: An example of a multip Choice 1 Choice 2 A text preference:	oreference page implementatio volean preference ole-choice preference Default value	n	Browse
			Restore <u>D</u> efaults	Apply
0			ОК	Cancel





 Para personalizar los controles que queremos emplear en nuestra hoja de ejemplo, en primer lugar deberemos determinar cuáles serán las preferencias que vamos a guardar, y actualizar la clase *PreferenceConstants* con los valores apropiados.







Después, deberemos modificar el método *ConsolePreferencePage.createFieldEditors()*, creando los controles adecuados <u>dependiendo</u> del tipo de valor que deseemos guardar, y asociándolo a una clave definida entre las constantes:







También, se debe ajustar la clase *Preferencelnitializer* para que establezca los valores iniciales de forma correcta.







 Por último, se deben modificar los métodos que escriben en la consola para que consulten si se debe escribir o no según las preferencias.

<pre>public void printError(String message) { if (getPreferenceStore().getBoolean(PreferenceConstants.P_ENABLE_STDERR)) console.getStderr().println(message); } public void printMessage(String message) { if (getPreferenceStore().getBoolean(PreferenceConstants.P_ENABLE_STDOUT)) console.getStdout().println(message); } public void printDebugMessage(String message) { if (getPreferenceStore().getBoolean(PreferenceConstants.P_ENABLE_STDOUT)) console.getStdout().println(message); } public void printDebugMessage(String message) { if (getPreferenceStore().getBoolean(PreferenceConstants.P_ENABLE_STDDBG)) console.getStddebg().println(message); } </pre>	ava 🕖 ExampleConsoleUIPlugin.java 🛛 🖓 🗖 🗖	D PreferenceInitializer.java	D PreferenceConstants.java	cePage.java	nsolePreference	Do
<pre> public void printMessage(String message) { if (getPreferenceStore().getBoolean(PreferenceConstants.P_ENABLE_STDOUT)) console.getStdout().println(message); } public void printDebugMessage(String message) { if (getPreferenceStore().getBoolean(PreferenceConstants.P_ENABLE_STDDBG)) console.getStddebg().println(message); } } </pre>	tants.P_ENABLE_STDERR))	{ an(PreferenceConstant essage);	ntError(String message erenceStore().getBoole .getStderr().println(m	c void prin f (getPrefe console,	public if	6
<pre>public void printDebugMessage(String message) { if (getPreferenceStore().getBoolean(PreferenceConstants.P_ENABLE_STDDBG)) console.getStddebg().println(message);</pre>	tants.P_ENABLE_STDOUT))	<pre>ye) { an(PreferenceConstant essage);</pre>	ntMessage(String messa erenceStore().getBoole .getStdout().println(m	c void prin f (getPrefe console	, public if }	6
}	tants.P_ENABLE_STDDBG))	nessage) { in(PreferenceConstant nessage);	ntDebugMessage(String) erenceStore().getBoole .getStddebg().println()	c void prin f (getPrefe console.	public if	•





 Podemos probar la funcionalidad de las preferencias de la consola, si por ejemplo modificamos la clase del editor para que nos avise cuando el editor se abre, se cierra, y cuando un archivo es salvado:





Eclipse en el soporte al desarrollo de software. Universidad Politécnica de Valencia. Abel Gómez [agomez@dsic.upv.es]



Creación de una hoja de preferencias con controles avanzados




Creación de una hoja de preferencias

- En el ejemplo visto anteriormente, una hoja de preferencias extiende a la clase *FieldEditorPreferencePage* porque esta proporciona constructores para crear los controles más comunes (casillas de texto, botones de radio, *checkboxes*, selectores de directorio, etc.). ¿Pero qué ocurre si necesitamos crear una hoja con controles avanzados (tablas, *layouts* en columnas, etc.)
- En ese caso, la clase de nuestra hoja de preferencias debe implementar la interfaz IWorkbenchPreferencePage. Para simplificar la implementación, nuestra clase puede heredar directamente de la clase *PreferencePage* (o sobreescribir los métodos necesarios de *FieldEditorPreferencePage* y heredar de ésta).
- Si se hereda de PreferencePage perderemos toda esa funcionalidad que añade FieldEditorPreferencePage.







Creación de una hoja de preferencias

- Cuando se crea una hoja de propiedades que extiende *PreferencePages*, la clase hija:
 - Debe implementar el método abstracto createControl().
 - Se recomienda implementar el método *doComputeSize()*.
 - Puede (y debería para que la página funcione correctamente, ya que en estos métodos es donde se carga/resetea/salva el estado de las preferencias de/hacia los controles) sobreescribir los métodos performOk(), performApply(), performDefaults(), performCancel(), y performHelp().
 - Las subclases también pueden emplear el método noDefaultAndApplyButton() permite eliminar los botones «Default» y «Apply».
- En caso de que la modificación de un valor deba reflejarse inmediatamente en el estado de otro control (puede ser la propia hoja de preferencia, u otro control completamente diferente, como una vista, o un editor) se puede implementar un listener de la siguiente manera:
 - El segundo control debe tener una clase que implemente la interfaz
 IPropertyChangeListener y el método *propertyChange(...)*. Este será el método que será invocado al detectar un cambio en las preferencias.
 - El segundo control debe registrar el *listener* en el *IPreferenceStore* correspondiente:

PluginClass.getDefault().getPreferenceStore.addPropertyChangeListener(miListener).





Creación de un *plug-in* de ayuda

ayuda





- Eclipse proporciona también un sistema centralizado y extensible de ayuda por medio de puntos de extensión.
- Cada *plug-in* puede contribuir al centro de ayuda de Eclipse con su propio contenido.
- El centro de ayuda se ejecuta en eclipse dentro del navegador web del sistema, realizando las peticiones sobre un servidor del propio Eclipse (un servidor *Apache Tomcat* incrustado en Eclipse).









- Crear un nuevo *plug-in* de ayuda para Eclipse es extremadamente sencillo gracias a los habituales asistentes de creación de nuevo *plug-in*, aunque el proceso de preparación puede resultar tedioso.
- Los documentos de ayuda son documentos HTML ordinarios, siendo un sistema simple y fácilmente entendible.
- Además, al tratarse de HTML estándar, se puede emplear cualquier otro editor externo a gusto del desarrollador.
- Con referencia a este tipo de *plug-ins* se recomienda que estén aislados del resto de *plug-ins* que conformen nuestra herramienta, no incluyendo mas que la funcionalidad necesaria.
- Para la creación de un nuevo proyecto de ayuda, se procede de la forma habitual ($New \rightarrow Project; Plug-in Project$).





				🗧 New plug-in project with sample help content
New Plug-in Plug-in Conter Enter the data re	I Project nt quired to generate the plug⊣n.			Sample Help Table of Contents Create a standalone or integrated table of contents.
Plug-in Propertie Plug-in ID: Plug-in Name: Plug-in Name: Plug-in Provider: Clacepath: Plug-in Options Clacepath: Plug-in Options Clacepath: This plug-in w Rich Client Appli Would you like to	es.example.help 1.0.0 Example.Help Plug-in ctivator, a Java dass that controls the plug-in's life cyde es.example.help.ExampleHelpPlugin dil make contributions to the UI cation cation	New Plug-in Project Templates Select one of the available templates to general Select one of the available templates to general Custom plug-in using one of the templates Available Templates: Custom plug-in wizard Figure definitions converter Hello, World Flug-in with a popur menu Plug-in with a popur page Plug-in with a property page Plug-in with an editor Plug-in with an eltor Plug-in with an incramental project builder	te a fully-funct This wizard c standaione o Extensions • org.eclips 	Label for table of contents: Sample Table of Contents ✓ Primary Generate a grimary table of contents for testing ✓ Generate a 'Getting Started' category ✓ Generate a 'Concepts' category ✓ Generate a 'Tasks' category ✓ Generate a 'Reference' category ✓ Generate a 'Samples' category





Por defecto, al crear un plug-in de ayuda de ejemplo, se crean los siguientes elementos:

- Archivos de manifiesto (*plugin.xml*, *MANIFEST.MF*)
- Carpeta «src/», con la clase activadora por defecto.
- Carpeta «*html/*», donde se albergan los documentos HTML de cada una de las páginas de nuestro *plug-in* de ayuda.
- Archivos «toc*.xml», archivos XML con la tabla de contenido del plug-in.









- Como se observa en el archivo de manifiesto, de los archivos «toc*.xml», el fichero «toc.xml» es el denominado «primario».
- Este es el que se mostrará en primer nivel de la ventana de ayuda de Eclipse.
 - Sample Table of Contents
 Getting Started
 Concepts
 Tasks
 Reference



🕼 es.e	example.help 🛛							
1<	?xml version=	="1.0"	' encodi	ng="UTF-8"?	>			~
2<	eclipse vers?	sion='	'3.2"?>					
3<	plugin>							
4								
5	<extension< td=""><th></th><td></td><td></td><td></td><td></td><td></td><td></td></extension<>							
6	point	t="org	g.eclips	e.help.toc"	>			
7	<toc< td=""><th></th><td></td><td></td><td></td><td></td><td></td><td></td></toc<>							
8	fi	ile="t	coc.xml"					
9	pi	rimary	/="true"	>				
10								
11	<toc< td=""><th></th><td></td><td></td><td></td><td></td><td></td><td></td></toc<>							
12	fi	ile="t	cocconce	pts.xml">				
13								
14	<toc< td=""><th></th><td></td><td></td><td></td><td></td><td></td><td></td></toc<>							
15	fi	ile="t	cocgetti	ngstarted.x	ml":	>		
16								
17	<toc< td=""><th></th><td></td><td></td><td></td><td></td><td></td><td></td></toc<>							
18	fi	ile="t	cocrefer	ence.xml">				
19								
20	<toc< td=""><th></th><td></td><td></td><td></td><td></td><td></td><td></td></toc<>							
21	fi	ile="t	cocsampl	es.xml">				
22								
23	<toc< td=""><th></th><td></td><td></td><td></td><td></td><td></td><td></td></toc<>							
24	fi	ile="t	toctasks	.xml">				
25								
26	<th>n></th> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>	n>						
27								
28<	/plugin>							
29								1
<								>
Overvie	ew Dependencies R	Runtime	Extensions	Extension Points	Build	MANIFEST.MF	plugin.xml	»,





- La etiqueta <*toc*> representa el elemento raíz de una tabla de contenido.
- El atributo *topic* es un enlace a un archivo HTML que describe el contenido del elemento TOC.
- El atributo *label* es un título breve para el elemento TOC, que sea legible para los humanos.
 - Sample Table of Contents
 Sample Table of Contents
 - Concepts
 - 🗄 🛄 Tasks
 - 🗄 💷 Reference
 - 🗉 💷 Samples

toc	.xml 🛛	
1<	<pre>?xml version="1.0" encoding="UTF-8"?></pre>	~
2<	<pre>?NLS TYPE="org.eclipse.help.toc"?></pre>	
3		
4<	toc label="Sample Table of Contents" topic="html/toc.html">	
5	<topic label="Getting Started"></topic>	
6	<pre><anchor id="gettingstarted"></anchor></pre>	
7		
8	<topic label="Concepts"></topic>	
9	<anchor id="concepts"></anchor>	
10		
11	<topic label="Tasks"></topic>	
12	<anchor id="tasks"></anchor>	
13		
14	<topic label="Reference"></topic>	
15	<anchor id="reference"></anchor>	
16		
17	<topic label="Samples"></topic>	
18	<anchor id="samples"></anchor>	
19		
20 ≲	/toc>	
21		
		-





- La etiqueta *<topic>* marca un determinado elemento del *toc*.
- Al igual que el elemento *<toc>*, dispone de una etiqueta.
- El elemento <*anchor*> define un identificador que será enlazado desde otro elemento <*toc*> definido en un archivo separado. Esto permite jerarquizar y anidar distintas tablas de contenido en diferentes ficheros.

	📄 toc.xml 😣				- E		
	1 xml vers:</th <th>ion="1.0"</th> <th>encoding="UTF-8"</th> <th>?></th> <th>^</th> <th></th> <th></th>	ion="1.0"	encoding="UTF-8"	?>	^		
	2 NLS TYPE=</th <th>="org.ecl:</th> <th>pse.help.toc"?></th> <th></th> <th></th> <th></th> <th></th>	="org.ecl:	pse.help.toc"?>				
	3						
	4 <toc label="</th"><th>="Sample 1</th><th>Table of Contents</th><th>" topic="html/t</th><th>toc.html"></th><th></th><th></th></toc>	="Sample 1	Table of Contents	" topic="html/t	toc.html">		
	5 <topic 1<="" th=""><th>label="Get</th><th>ting Started"></th><th></th><th></th><th></th><th></th></topic>	label="Get	ting Started">				
	6 <anch< th=""><th>hor id="gg</th><th>ttingstarted"/></th><th></th><th></th><th></th><th></th></anch<>	hor id="gg	ttingstarted"/>				
	7 <th>></th> <th></th> <th></th> <th></th> <th></th> <th></th>	>					
	8 <topic 3<="" th=""><th>label="Com</th><th>ncepts"></th><th></th><th></th><th></th><th></th></topic>	label="Com	ncepts">				
Sample Table of Contents		hor id="co	oncepts"/>		1.0		
Getting Started		> itoc.xm	l 📄 tocconcepts.xml	ille es.example.help	org.eclipse.help.toc	tocgettingstarted.xml 🛛	
		labe 1 x</th <th>ml version="1.0" en</th> <th>ncoding="UTF-8"?></th> <th></th> <th></th> <th><u>~</u></th>	ml version="1.0" en	ncoding="UTF-8"?>			<u>~</u>
Main Topic		2 N</th <th>LS TYPE="org.eclip:</th> <th>se.help.toc"?></th> <th></th> <th></th> <th></th>	LS TYPE="org.eclip:	se.help.toc"?>			
Sub Topic		4410	c label="Getting St	arted" link to="	toc xml#gettingst	arted">	
🗆 💷 Main Topic 2		5	<topic href="h</th><th>tml/gettingstarte</th><th>d/maintopic.html" label="Main</th><th>n Topic"></topic>				
Sub Topic 2		6	<topic <="" label='</th><th>"Sub Topic" href=</th><th>"html/gettingstar</th><th>ted/subtopic.html" /></th><th></th></tr><tr><th>E Concepts</th><th></th><th>7</th><th></topic></th><th></th><th></th><th></th><th></th></tr><tr><th>E D Tacks</th><th></th><th>8</th><th><topic label="Main</th><th>n Topic 2"></th><th></th><th></th><th></th></tr><tr><th></th><th></th><th>9</th><th><topic label=' th=""><th>"Sub Topic 2" hre</th><th>i="html/gettingst</th><th>arted/subtopic2.html"</th><th>/></th></topic>	"Sub Topic 2" hre	i="html/gettingst	arted/subtopic2.html"	/>
		11 <th><!-- CODICS</th--><th></th><th></th><th></th><th></th></th>	CODICS</th <th></th> <th></th> <th></th> <th></th>				
🖽 💷 Samples		12					
							~
		<					>





- Como ejemplo, el archivo «tocgettingstarted.xml» es enlazado por la tabla de contenido «toc.xml».
- Este archivo define una estructura similar al archivo anterior.
- A resaltar encontramos los atributos *href*, que enlazan un determinado *topic* con un archivo HTML existente, y el atributo *link_to*, que enlaza con el *anchor* definido en la tabla de contenido de primer nivel. De esta manera se establece que esta tabla se anida dentro de la de primer nivel, tal y como se observa en el menú de ayuda.

Sample Table of Contents	📄 toc.xml 🛛 📄 tocconcepts.xml 🕼 es.example.help 🖓 org.eclipse.help.toc 📄 tocgettingstarted.xml 🛛
E 🖬 Main Topic	<pre>1<?xml version="1.0" encoding="UTF-8"?> 2<?NLS TYPE="org.eclipse.help.toc"?> 3</pre>
Sub Topic Main Topic 2 Sub Topic 2	<pre>4<toc label="Getting Started" link_to="toc.xml#gettingstarted"> 5 <topic href="html/gettingstarted/maintopic.html" label="Main Topic"> 6 <topic href="html/gettingstarted/subtopic.html" label="Sub Topic"></topic></topic></toc></pre>
 E Concepts E III Tasks 	<pre>7 8 <topic label="Main Topic 2"> 9 <topic <="" href="html/gettingstarted/subtopic2.html" label="Sub Topic 2" pre=""></topic></topic></pre>
 □ Reference □ Samples 	10 11 12



Eclipse en el soporte al desarrollo de software. Universidad Politécnica de Valencia. Abel Gómez [agomez@dsic.upv.es]



/>

Ξ

Ejecución de un Infocenter

Flecociou de ou unocemei





Ejecución de un Infocenter

El sistema de ayuda de Eclipse puede accederse desde una intranet o incluso Internet.

- Para ello, puede configurarse Eclipse únicamente como servidor web, sin que sea necesario lanzar el entorno por completo.
- Este servidor web se denomina *Infocenter* en la documentación de eclipse.
- En su invocación, permite especificar numerosos argumentos, como el puerto donde se desean escuchar las peticiones. La lista completa de opciones se encuentra en la documentación de eclipse

http://help.eclipse.org/help33/topic/org.eclipse.platform.doc.isv/g uide/ua_help_setup.htm).





Ejecución de un Infocenter

- La creación de un *Infocenter* es sencilla:
 - Se ha de descargar la plataforma básica de eclipse (Eclipse Platform Runtime Binary).
 - Se debe descomprimir la plataforma básica de eclipse. Esto proporcionará todos los binarios mínimos necesarios, incluido el sistema de ayuda. (El conjunto mínimo de *plug-ins* requeridos es: *org.apache.lucene, org.eclipse.core.runtime, org.eclipse.help, org.eclipse.help.appserver, org.eclipse.help.base, org.eclipse.help.webapp, org.eclipse.osgi, org.eclipse.tomcat, org.eclipse.update.configurator*)
 - Se deben de incluir los *plug-ins* propios de ayuda en la carpeta de plug-ins.
- El Infocenter se ejecuta mediante el comando

java -classpath d:\myApp\eclipse\plugins\org.eclipse.help.base_[version].jar org.eclipse.help.standalone.Infocenter -command start -eclipsehome d:\myApp\eclipse -port 8081

Para detener el Infocenter se ejecutará:





Creación de un punto de extensión

extensión





- Como se ha visto anteriormente, un *punto de* extensión es una forma de comunicación entre dos plug-ins.
- Fundamentalmente, permite que un plugin A, haga pública una interfaz a la que se conectará cualquier otro plug-in (B).
- A diferencia de la mera dependencia entre *plug-ins*, donde el *plug-in* B hace uso del código de A, sin que este último tenga control sobre éste, los puntos de extensión permiten al *plug-in* A conocer a B, y ejecutar código de este (puesto que A determina la interfaz que éste debe implementar).
- A continuación veremos dos ejemplos de puntos de extensión.







- En primer lugar, vamos a crear un sencillo *plug-in* de ejemplo empleando el asistente de creación de un proyecto «Hola mundo!».
- Este tipo de *plug-ins* simplemente contribuyen un botón en la barra de iconos de eclipse, que al ser pulsado, ejecuta un método *run()*. Esto nos permitirá invocar la funcionalidad del *plug-in* de forma sencilla.







 Para añadir un punto de extensión, empleamos el botón «Add...» de la pestaña «Extension Points» en el editor de archivos de manifiesto.

ga es.example.extension ≥	3	
44 Extension Poin	nts	?
All Extension Points		
Edit extension points defin plug-in in the following sec	ied by this tion.	
	<u>A</u> dd	
	•	

🖶 New Extension Poi	nt	
Extension Point Pro	perties	0
Specify properties of the	new extension point.	
Extension Point ID:	es.example.extension	
Extension Point Name:	Example Extension Point	
Extension Point Schema:	schema/es.example.extension.exsd	
Edit extension point sc	hema when done	
0	Einish	Cancel



54



	f 12	
eneral In	formation	Schema Inclusions
rnis secuori	describes general information about this schema.	specify schemas to be included with this schema.
lug-in ID:	es.example.extension	Add
oint ID:	es.example.extension	Remov
oint Name:	Example Extension Point	
ocumenta elect the se extension po B Descrip	ation ection from the list and enter text in the editor below. T oint. Use HTML tags where needed. tion)	This text will be used to compose the reference HTML document for this → Supplied Implementation
elect the se stension po	ation ection from the list and enter text in the editor below, T oint. Use HTML tags where needed. tion 🚔 Since 🚔 Examples 🚔 API Information	his text will be used to compose the reference HTML document for this 을 Supplied Implementation 믙 Copyright
ecumenta elect the se extension po Descrip Enter (ation ection from the list and enter text in the editor below. T pint. Use HTML tags where needed. tion 🖶 Since 🚔 Examples 🚔 API Information description of this extension poin	'his text will be used to compose the reference HTML document for this 을 Supplied Implementation 盛 Copyright nた.]
ocumenta elect the se xtension po Descrip Enter (ation ection from the list and enter text in the editor below. T oint. Use HTML tags where needed. Ition 🖶 Since 🖨 Examples 🖨 API Information 着 description of this extension poin	his text will be used to compose the reference HTML document for this Supplied Implementation 🖶 Copyright Int.]
ocumenta elect the sa xtension po Descrip Enter o	ation ection from the list and enter text in the editor below. T oint. Use HTML tags where needed. Ition 🖶 Since 🖶 Examples 🖶 API Information description of this extension poin	his text will be used to compose the reference HTML document for this Supplied Implementation 🖶 Copyright nt.]
ocumenta elect the se xtension po Descrip Enter o	ation ection from the list and enter text in the editor below. T point. Use HTML tags where needed. tion 🖶 Since 🚔 Examples 🚔 API Information description of this extension poin	his text will be used to compose the reference HTML document for this Supplied Implementation Copyright nt.]
ecumenta elect the sector extension po E Descrip	ation ection from the list and enter text in the editor below. T oint. Use HTML tags where needed. tion 🖶 Since 🖶 Examples 🖶 API Information description of this extension poin	his text will be used to compose the reference HTML document for this Supplied Implementation 플 Copyright nt .]



55



Example Extension Point Preview Reference Extension Point Elements Element Details Specify the XML elements and attributes which are allowed in this extension point. Properties for the "extension" element. Image: Specify the XML elements and attributes which are allowed in this extension point. Name: extension Image: Specify the XML element point. New Element Image: Specify the XML element point. New Attribute Image: Specify the XML element point. Name: extension Image: Specify the XML element point. Name: extension Image: Specify the XML element point. Name: Deprecated: Image: Difference Image: Specify the XML element point
Extension Point Elements Element Details Specify the XML elements and attributes which are allowed in this extension point. Properties for the "extension" element. Image: Content of the sectors of the sector



56





- Al esquema por defecto, vamos a añadir inicialmente un nuevo elemento, *Message*, que tendrá dos atributos, *plugin_id* y *message*.
- Cada *plug-in* que se conecte al *extension point*, notificará un identificador y un mensaje que quiera mandar al mundo.
- Nuestro *plug-in* del punto de extensión, será capaz de consultar la lista de *plug-ins* que están conectados, y cuando pulsemos el botón de «Hola mundo!», nos dirá todos los mensajes que hayan indicado los *plug-ins* conectados.





• En primer lugar, el nuevo esquema del punto de extensión queda como el siguiente:

Extension Found Liennends		LICITETI DECOIS	
Specify the XML elements and attributes which are allowed i point. extension General Message General Message	New Element New Attribute	Properties for the "Message" element. Name: Message Deprecated: true false Label Property: Icon: Translatable: true Translatable: Translatable: EMPTY	







• Y por su parte, el método *run()* de nuestro botón, de la siguiente manera:

```
public void run(IAction action) {
    StringBuffer buffer = new StringBuffer();
    IExtensionRegistry reg = Platform.getExtensionRegistry();
    IConfigurationElement[] extensions = reg
                          .getConfigurationElementsFor("es.example.extension");
    for (int i = 0; i < extensions.length; i++) {
        IConfigurationElement element = extensions[i];
        buffer.append(element.getAttribute("plugin_id"));
        buffer.append(": \"");
        buffer.append(element.getAttribute("message"));
        buffer.append("\"\n");
    }
    MessageDialog.openInformation(
        window.getShell(),
    </pre>
```

"Plug-ins using the extension point", buffer.toString());





- En este punto, está todo listo para que nuestros plug-ins se conecten al punto de extensión.
- Para ello podemos crear un proyecto de *plug-in* vacío, y editaremos el archivo de manifiesto:







60



 Y seleccionaremos el punto de extensión recién creado.
 Si nuestro *plug-in* con el punto de extensión de ejemplo no está entre las dependencias, deberemos desmarcar el *checkbox*.

E New Extension				- DX
Extension Point Selection Select an extension point from the	nose available in	the list.		
Extension Points Extension Wiz Extension Point filter: *exal e.e.xample.extension org.eclipse.emf.common.ui.	examples rom the required o see its descr	plug-ins iption)		
0	< <u>B</u> ack	<u>N</u> ext >	Einish	Cancel





- Una vez se ha incluido el punto de extensión, aparecen en el editor los dos campos que hemos indicado que se pueden establecer.
- Basta rellenar su contenido, y podemos lanzar el conjunto de plug-ins a ejecución.

All Extensions Define extensions for this plug-in in th	$\downarrow^{a}_{\mathbf{Z}} \square$ e following section.	Extension Element Details Set the properties of "Message". Require
type filter text	Add Edit Up Down	heids are denoted by ~~. message*: Hola mundo! plugin_id*: es.example.dummy





- Una vez se ha incluido el punto de extensión, aparecen en el editor los dos campos que hemos indicado que se pueden establecer.
- Basta rellenar su contenido, y podemos lanzar el conjunto de plug-ins a ejecución.







- Hemos visto un ejemplo de cómo el *plug-in* que ofrece el punto de extensión es capaz de consultar los atributos rellenados por aquellos *plug-ins* que importan el punto de extensión.
- Pero, ¿cómo hacer para ejecutar código de un *plug-in* que importa el punto de extensión?
- Tomemos el siguiente caso, y supongamos que nuestro plugin de «Hola mundo!» quiere que cada plug-in salude por sí mismo cuando pulsemos el botón, en lugar de solicitar el mensaje a cada plug-in que emplea el punto de extensión...







- En primer lugar, hemos de definir la interfaz que debe implementar cada *plug-in* para que se pueda conectar al punto de extensión.
- Llamaremos a esta interfaz ISaludador:

package es.example.extension;

```
public interface ISaludador {
   public void saludar();
}
```





Igualmente, deberemos modificar la definición del punto de extensión, indicando que necesitaremos un nuevo atributo de tipo *Java*, y que implementará la interfaz *ISaludador*. Debemos asegurarnos de que nuestro exporta el paquete donde se define la interfaz *ISaludador*.

xample Extension Point			Preview Reference	<u>: Document</u> ?
xtension Point Elements	Attribute Deta	ails		
Specify the XML elements and attributes which are allowed in this extension	Properties for the	he "saludador" attribu	te.	
	Name:	saludador		\leftarrow
extension New Element	Deprecated:	🔿 true 🛛 💿 fals	e	
Message New Attribute	Use:	required		~
©® point	Default Value:			
aname	Type:	java		K~ /
E- (Message	Extender			Browse
© nessage	<u>Extends.</u>			browse
(a) plugin_la	Implements:	es.example.extens	sion.ISaludador	Browse
-				





Y debemos modificar nuestro método *run()*, para que consulte el nuevo atributo «*saludador*», que será un nombre de clase, y cree una instancia de ésta.

```
public void run(IAction action) {
   StringBuffer buffer = new StringBuffer();
   IExtensionRegistry reg = Platform.getExtensionRegistry();
   IConfigurationElement[] extensions = reg
      .getConfigurationElementsFor("es.example.extension");
   for (int i = 0; i < extensions.length; i++) {
      IConfigurationElement element = extensions[i];
      ISaludador saludador = null;
      try {
         saludador = (ISaludador)
            element.createExecutableExtension("saludador");
      } catch (CoreException e) { e.printStackTrace(); }
      saludador.saludar();
   }
}</pre>
```



}

}

67



- Por último, modificamos nuestro plug-in dummy:
 - Creamos la nueva clase CustomSaludador, que implementa la interfaz ISaludador.







- Por último, modificamos nuestro *plug-in dummy*:
 - Actualizamos el archivo de manifiesto, para indicar la clase que implementa la interfaz *ISaludador*.

🚺 CustomSaludador.java 🕼 SampleAction.java	🚯 es.exa	mple.dummy 🖾	DefaultClassLoader	.c [»] 2	
Extensions					?
All Extensions	↓ ^a z ⊑	Extension	Element Details		
Define extensions for this plug-in in the following section.		Set the pro	perties of "Message". Re	quired fields are deno	oted by "*".
type filter text		message*:	Hola mundo!		
□ ⊕= es example extension	<u>A</u> dd	plugin_id*;	es.example.dummy		
(Message)	Edit	saludador*:	es.example.dummy.C	CustomSaludador	Browse
		1			
	Up	J			\
	Down	J			\mathbf{N}
					١
Overview Dependencies Runtime (Extensions) Extension R	oints Build	MANIFEST.MF plug	jin.xmi build.properties		





- En este punto, ya podemos ejecutar nuestros *plug-ins* en una nueva instancia de eclipse:

🖨 Java - Eclipse SDK		
<u>Eile E</u> dit <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject	Sample Menu Run Window Help	
i 🗈 - 🗟 📄 🕸 - 🛛)•Q₄• @₩@• @% 2 -₽-+++	Java
🛱 Packag 😕 🖹 Hierac 🗖 🗖	🗖 🖬 Outine 🗙	_
- ↔ @ □ ♥ ♥	An outline is not available es.example.dummy K Hola desde el plug-in dummy! OK	
	Problems @ Javadoc 😥 Declaration 📮 Console 🐹	
	No consoles to display at this time.	





Creación de una nueva perspectiva

perspectiva





Creación de una perspectiva

- Eclipse proporciona un asistente de ejemplo para crear una nueva perspectiva.
- Éste se lanza desde la localización habitual, en el asistente de creación de un nuevo plug-in.

/allable Templates:		
Name	Extension Point	Select A
Project Builder and Nature	- org.eclipse.core.resources.buil	Deselect
Icon Decorator	- org.eclipse.ui.decorators	
□ = XML Editor	- org.eclipse.ui.editors	
Hello world command contri	- org.eclipse.ui.commands	
I = "Hello world" Action Set	- org.eclipse.ui.actionSets	
□ = Help Table of Contents	- org.eclipse.help.toc	
File Import Wizard	- org.eclipse.ui.importWizards	
Multi-page Editor	- org.eclipse.ui.editors	
New File Wizard	org.eclipse.ui.newWizards	
Release Engineering Perspec	org.eclipse.ui.perspectives	
	- org.eclipse.ui.popupMenus	
Preference Page	org.eclipse.ui.preferencePages	
Property Page	- org.edipse.ui.propertyPages	
Splash Handler		
Universal Welcome Contribut	- org.eclipse.ui.intro.configExten	
©=View	- org.eclipse.ui.views	
out of 16 selected.		
his template creates a release enginee Hease engineering, the theme allows for emonstrates how to access the various	ring themed perspective. While most users will not be doing any or a very logical layout of the perspective elements and s team and CVS components from within a perspective.	

E New plug-in project with custom templates					- D X
Perspective Options Choose the options for the release engineering perspective.					
Java Package:	gtrg.perspectives				
Perspective <u>C</u> lass Name:	RelEngPerspective				
Perspective Name:	Release Engineering				
Add <u>R</u> elated Perspecti	ve Shortcuts				
Add Show View Shortcuts					
Add New Wizard Shortcuts					
☑ Add Menu and Toolbar Contributions (Action Sets)					
?		< <u>B</u> ack	Next >	Einish	Cancel



72


Creación de una perspectiva

- El asistente crea un proyecto típico.
- La configuración de la perspectiva se encuentra toda en una única clase, en este caso, *RelEngPerspective*.

📱 Package Explorer 🖾 🌋 Plug-ins 📃 🗖	🗊 SampleAction.java 🕼 es.example.perspecti 🛛 🕽 Rel	lEngPerspective.ja	- 0
← ← @ E & ▼ = E es.example.console = es.example.dummy	Extensions	Extension Element Details	0
 es.example.editor es.example.extension es.example.help es.example.perspective es.example.perspective es.example.perspective es.example.perspective.perspectives RelEngPerspective.java JRE System Library [jre 1.6.0_05] Plug-in Dependencies icons META-INF MANIFEST.MF build.properties plugin.xml 	Define extensions for this plug-in in the following section. type filter text Image: Org.eclipse.ui.perspectives Image: Org.eclipse.ui.perspeclipse.ui.perspectives	Set the properties of "perspective". Required f denoted by "*". id*: es.example.perspective.perspective name*: Release Engineering dass*: espectives.RelEngPerspective icon: icons/releng_gears.gif fixed:	ields are s.RelEngPers Browse Browse



73

Eclipse en el soporte al desarrollo de software. Universidad Politécnica de Valencia. Abel Gómez [agomez@dsic.upv.es]



Creación de una perspectiva

- Todo elemento del *workbench* tiene un identificador (id) asociado.
- Este ide es el que se emplea para poder determinar cuáles serán los elementos visibles por defecto en la perspectiva.
- La clase *RelEngPerspective* define los siguientes métodos, cada uno de ellos mostrando un grupo distinto de elementos del *workbench*:
 - createInitialLayout(IPageLayout) Este método invoca a todos los demás. Inicializa la perspectiva.
 - addViews() Define las vistas activas del workbench, así como su posición en él.
 - addActionSets() Añade los grupos de iconos que estarán disponibles en la barra de herramientas.
 - addPerspectiveShortcuts() Determina para qué perspectivas se monstrarán los iconos en la zona de atajos.
 - addNewWizardShortcuts() Indica qué Asistentes se mostrarán en el menú contextual de crear un nuevo elemento.
 - addViewShortcuts() Determina cuáles serán los atajos para activar una vista que se mostrarán.







Ejercicio de evaluación





Ejercicio de evaluación

- Modificar el *plug-in* de consola para que ofrezca un punto de extensión que solicitará dos parámetros:
 - plugin_id el id del plugin que importe el punto de extensión.
 - enabled si la consola se encuentra activa por defecto para ese plug-in.
- Modificar la hoja de propiedades de la consola para que consulte todos los *plug-ins* que importan el punto de extensión.
 - Debe definir un control que permita, para cada uno de los *plug-ins*, definir si la consola está activa o no.
- Modificar los métodos printError, printMessage, y printDebugMessage para que reciban dos argumentos: el identificador del plug-in que lo invoca, y el mensaje a mostrar.
 - El plug-in de consola sólo mostrará el mensaje si el plug-in ha definido la consola como activa en las preferencias.





