

## Les points d'extension

<b>LES POINTS D'EXTENSION .....</b>	<b>1</b>
CONTENU DE LA SECTION .....	2
LA CREATION D'UN NOUVEAU POINT D'EXTENSION (1) .....	4
<i>un court rappel sur les points d'extension.....</i>	4
<i>le problème pratique à résoudre .....</i>	4
<i>les moyens offerts par Eclipse.....</i>	4
LE MODELE DE PLUG-IN (1) .....	5
<i>le modèle .....</i>	5
LE MODELE DE PLUG-IN (2) .....	6
<i>le modèle en image .....</i>	6
LE DEPOT DE PLUGINS ET LES DESCRIPTEURS DE PLUG-IN .....	7
<i>le rôle .....</i>	7
<i>l'interface IPluginRegistry .....</i>	7
<i>l'interface IPluginDescriptor.....</i>	7
LES DESCRIPTEURS D'EXTENSION ET DE POINTS D'EXTENSION .....	8
<i>l'interface IExtension .....</i>	8
<i>l'interface IExtensionPoint.....</i>	8
<i>l'interface IConfigurationElement .....</i>	8
LA CLASSE PLUGIN (1) .....	9
<i>le rôle .....</i>	9
<i>le chargement.....</i>	9
<i>l'accès aux ressources statiques .....</i>	9
LA CLASSE PLUGIN (2) .....	10
<i>l'accès aux Preferences et aux informations d'état .....</i>	10
<i>la classe AbstractUIPlugin .....</i>	10
LA CLASSE PLUGIN (3) .....	11
<i>le rôle .....</i>	11
LA DECLARATION D'UN POINT D'EXTENSION .....	12
<i>La marche à suivre .....</i>	12
LA DEFINITION D'UN SCHEMA DE POINT D'EXTENSION (1) .....	13
<i>La marche à suivre .....</i>	13
LA DEFINITION D'UN SCHEMA DE POINT D'EXTENSION (1) .....	14
<i>les propriétés des éléments et attributs.....</i>	14
<i>la grammaire des schemas .....</i>	14

## développement de plugins Eclipse

UN PREMIER EXEMPLE SIMPLE (1) .....	15
<i>un générateur périodique de ticks</i> .....	15
UN PREMIER EXEMPLE SIMPLE (2) .....	16
<i>le code générique (et généré)</i> .....	16
UN PREMIER EXEMPLE SIMPLE (3) .....	17
<i>le code correspondant au lancement et arrêt du ticker</i> .....	17
UN PREMIER EXEMPLE SIMPLE (3) .....	18
<i>le code correspondant à l'instanciation du service</i> .....	18
LES SCHEMA .EXSD .....	20
<i>Constructions autorisées</i> .....	20
L'EDITEUR INTEGRE DE SCHEMAS .....	23
<i>Constructions autorisées</i> .....	23
EXTENSION POINT SCHEMA EDITOR .....	24
<i>L'ouverture de l'éditeur</i> .....	24
<i>Ajout d'un nouvel élément</i> .....	24
<i>la documentation du schema</i> .....	24

## La création d'un nouveau point d'extension (1)

### un court rappel sur les points d'extension

- Eclipse = modèle très simple de composition
- point d'extension =
  - déclaration d'un composant disponible dans la plate-forme
  - point de communication avec d'autres plug-ins
  - vision externe d'un plugin (vision externe d'un composant)
- extension = offrir une interface de callback pour un point d'extension

plug-in = composant Eclipse

### le problème pratique à résoudre

- mettre en place :
  - un mécanisme d'instanciation (paresseux)
  - un mécanisme de communication avec les classes d'extension

### les moyens offerts par Eclipse

- un PluginRegistry contenant la représentation java des plug-in manifests (plugin.xml)
  - un modèle de plug-in → une API de découverte des informations et d'instanciation

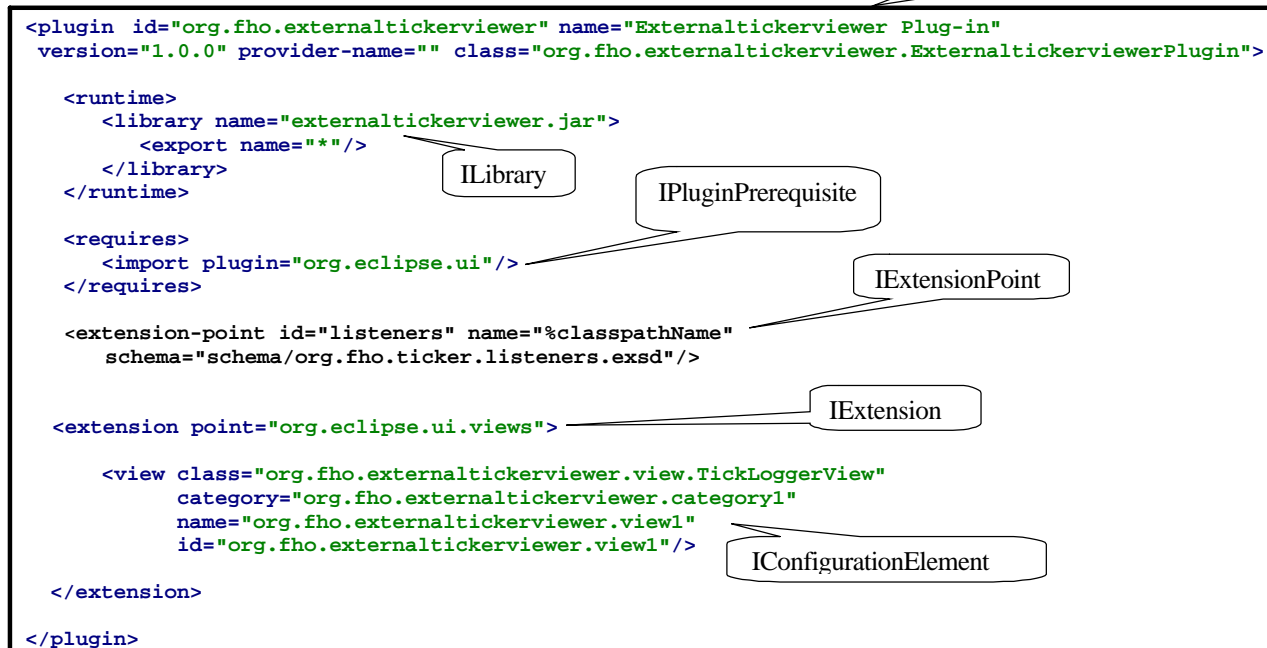
## Le modèle de plug-in (1)

### le modèle

- un descripteur de plug-in : **IPluginDescriptor**
  - indication des dépendances : **IPluginPrerequisite**
  - indication des libraries contenant les .class du plugin : **ILibrary**
- Si contribution à des extensions existantes : **IExtension**
  - + informations fournies aux extensions : **IConfigurationElement**
- Si définition d'un extension-point : **IExtensionPoint**
  - + informations propres au point d'extension : **IConfigurationElement**

## Le modèle de plug-in (2)

## le modèle en image



### le rôle

- le `PluginRegistry` contient l'ensemble des descripteurs de plug-ins de la plate-forme

### l'interface `IPluginRegistry`

- principales méthodes du registry :  
`IPluginDescriptor getPluginDescriptor(String pluginIdent)`  
`IPluginDescriptor[] getPluginDescriptors()`  
`IExtensionPoint getExtensionPoint(String extPointName)`  
`IExtensionPoint[] getExtensionPoints()`  
`IPrerequisite[] getPluginPrerequisites()`  
`IExtension getExtension(String pluginId, String extPointName)`

### l'interface `IPluginDescriptor`

- possède les accesseurs pour accéder aux éléments de description du plugin :  
`IExtensionPoint getExtensionPoint(String extPointName)`  
`IExtensionPoint[] getExtensionPoints()`  
`IPrerequisite[] getPluginPrerequisites()`  
`IExtension getExtension(String extName)`  
`IExtension[] getExtensions()`  
`URL getInstallURL()`  
`ILibrary[] getRuntimeLibraries()`  
.....

## Les descripteurs d'extension et de points d'extension

### l'interface IExtension

```
IConfigurationElement[] getConfigurationElements()  
String getExtensionPointUniquelidentifiant()  
String getUniquelidentifiant()
```

### l'interface IExtensionPoint

```
String getUniquelidentifiant()  
IConfigurationElement[] getConfigurationElements() //retourne un tableau des éléments de configuration  
IExtension[] getExtensions() //retourne un tableau des extensions (contribuées)  
  
String getSchemaReference()
```

### l'interface IConfigurationElement

```
IConfigurationElement[] getChildren()  
IConfigurationElement[] getChildren(String name)  
IConfigurationElement[] getParent()  
  
String[] getAttributeNames()  
String getAttribute(String name)  
  
String getSchemaReference()  
Object createExecutableExtension(String propName)
```

## La classe Plugin (1)

### le rôle

- **Singleton** offrant un point d'accès aux :
  - ressources statiques associées au Plug-in
  - aux préférences spécifiques et informations d'état
  - initialisation/finalisation

### le chargement

- initialisation au chargement
- finalisation au déchargement
- la méthode **startup()** est invoqué au chargement des classes (pas du Plug-in)
- chargement au démarrage obtenu par :
  - `<extension point= "org.eclipse.ui.startup">`
  - implémentation de l'interface `org.eclipse.ui.IStartup`

### l'accès aux ressources statiques

- accès aux ressources déclarées dans le plugin.xml  
**URL find(IPath resPath)**  
**InputStream openStream(IPath resPath)**

toujours invoquer la méthode parente

```

public void startup() throws CoreException {
    super.startup();
    .....
}

public void shutdown() throws CoreException {
    super.shutdown();
    .....
}
    
```

## La classe Plugin (2)

### l'accès aux Preferences et aux informations d'état

- accès aux ressources déclarées dans le plugin.xml  
**Preferences** `getPluginPreferences()`, `void savePluginPreferences()`  
`void initializeDefaultPluginPreferences()`  
**IPath** `getStateLocation()`

Attention : méthode `getPreferenceStore()` et interface `IPreferenceStore` dépréciées

### la classe `AbstractUIPlugin`

- sous-classe de `Plugin` pour les classes UI
- sauvetage automatique des préférences dans le `shutdown()`
- **ImageRegistry** `getImageRegistry()`, **IDialogSettings** `getDialogSettings()`

## La classe Plugin (3)

### le rôle

- **Singleton** offrant un point d'accès aux :
  - ressources statiques associées au Plug-in
  - préférences spécifiques
  - informations d'état
- initialisation au chargement
- finalisation au déchargement

```
.....  
public class Tp2Plugin extends AbstractUIPlugin {  
    private static Tp2Plugin plugin;  
    private ResourceBundle resourceBundle;  
  
    public Tp2Plugin() {  
        super();  
        plugin = this;  
        try {  
            resourceBundle = ResourceBundle.getBundle("org.fho.tp2.Tp2PluginResources");  
        } catch (MissingResourceException x) {  
            resourceBundle = null;  
        }  
    }  
  
    public void start(BundleContext context) throws Exception { super.start(context); }  
    public void stop(BundleContext context) throws Exception { super.stop(context); }  
  
    public static Tp2Plugin getDefault() { return plugin; }  
    .....  
}
```

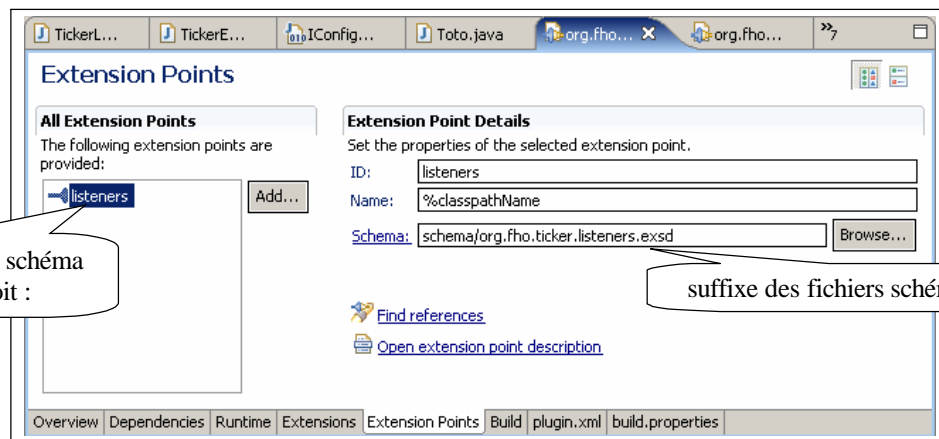
## La déclaration d'un point d'extension

### La marche à suivre

- aller dans l'onglet **Extension Points**
- cliquer sur Add...
- remplir le formulaire

L'identifiant complet du point d'extension est pluginID.ID

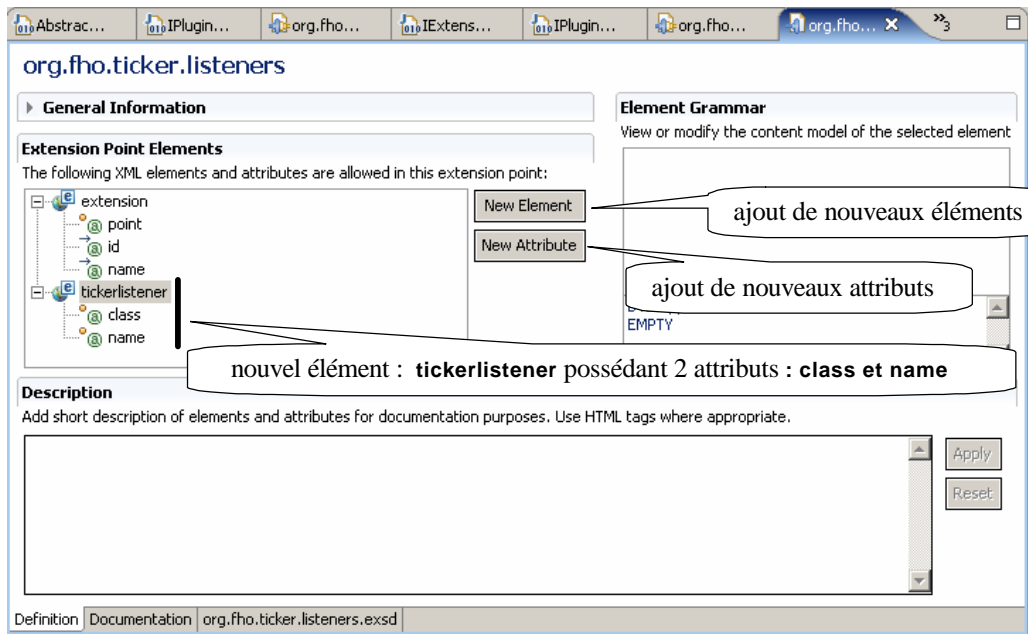
- l'ID est une string alphanumérique
- le schéma est une définition XML du point d'extension (voir suite)



## La définition d'un schéma de point d'extension (1)

### La marche à suivre

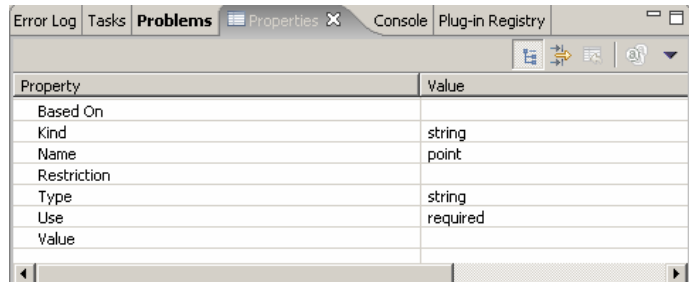
- un point d'extension possède nécessairement un élément : **extension** avec 3 attributs : **point, id, name**



## La définition d'un schéma de point d'extension (1)

### les propriétés des éléments et attributs

- les caractéristiques d'un élément sont les strings : **Icon, Label Attribute, Name**
- les caractéristiques d'un attributs sont les strings
  - **Based On :**
  - **Kind :** string, java ou resource (indique comment interpréter l'attribut : simple string, classe java, ressource)
  - **Name :** le nom de l'attribut (string)
  - **Restriction :** none ou enumeration
  - **Type :** string ou boolean
  - **Use :** required, optional ou default
  - **Value :**



The screenshot shows the Eclipse IDE's Properties window for an extension point. The window has tabs for Error Log, Tasks, Problems, Properties (selected), Console, and Plug-in Registry. The Properties window displays a table with the following data:

Property	Value
Based On	
Kind	string
Name	point
Restriction	
Type	string
Use	required
Value	

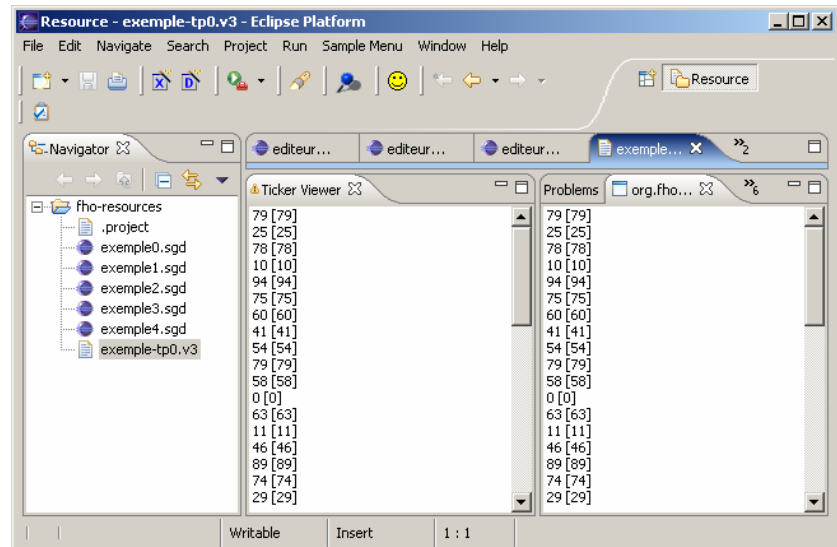
### la grammaire des schemas

- Sous ensemble des schemas XML
  - sequence [minOccurs, maxOccurs]
  - choice
  - all

## Un premier exemple simple (1)

### un générateur périodique de ticks

- génération d'un nombre aléatoire
- génération périodique
- un observateur associé au plug-in
- classe d'événement : **TickerEvent**
- interface **TickerListener**



## Un premier exemple simple (2)

### le code générique (et généré)

```
public class TickerPlugin extends AbstractUIPlugin implements Runnable {
    private static TickerPlugin plugin;
    private ResourceBundle resourceBundle;
    private static final String EXTENSION_POINT_NAME = "org.fho.ticker.listeners";
    .....

    public TickerPlugin() {
        super();
        plugin = this;
        try {
            resourceBundle = ResourceBundle.getBundle("org.fho.ticker.TickerPluginResources");
        } catch (MissingResourceException x) { resourceBundle = null; }
        .....
    }
    .....

    public static TickerPlugin getDefault() { return plugin; }

    public static String getResourceString(String key) {
        ResourceBundle bundle = TickerPlugin.getDefault().getResourceBundle();
        try {
            return (bundle != null) ? bundle.getString(key) : key;
        } catch (MissingResourceException e) { return key; }
    }

    public ResourceBundle getResourceBundle() { return resourceBundle; }
```

## Un premier exemple simple (3)

le code correspondant au lancement et arrêt du ticker

```
public class TickerPlugin extends AbstractUIPlugin implements Runnable {  
  
    public void start(BundleContext context) throws Exception {  
        super.start(context);  
        if (ticker != null) { ticker = new Ticker(); ticker.start(); }  
    }  
  
    public void stop(BundleContext context) throws Exception {  
        super.stop(context);  
        ticker.interrupt();  
        ticker = null;  
    }  
}
```

Création et lancement du ticker  
au chargement du plug-in

Arrêt du ticker au déchargement  
du plug-in (i.e. arrêt d'Eclipse)

```
public class Ticker extends Thread {  
  
    public void run() {  
        while(!stopped) {  
            TickerEvent tick = new TickerEvent((int)(Math.random() * 100), System.currentTimeMillis());  
  
            for (Iterator it = listeners.iterator(); it.hasNext(); ) {  
                TickerListener listener = (TickerListener)it.next();  
                listener.tickEmitted(tick);  
            }  
            try { Thread.sleep(2000); } catch (InterruptedException e) { stopped = true; }  
        }  
    }  
}
```

## Un premier exemple simple (3)

le code correspondant à l'instanciation du service

```

public class TickerPlugin extends AbstractUIPlugin implements Runnable {
    .....
    private LinkedList listeners;
    private Thread ticker;
    private boolean stopped = false;

    public TickerPlugin() {
        .....
        listeners = new LinkedList();
        ticker = new Thread(this);
        retrieveListeners();
    }
    .....

    private void retrieveListeners() {
        IExtensionRegistry registry = Platform.getExtensionRegistry();
        IExtensionPoint extensionPoint =
            registry.getExtensionPoint(EXTENSION_POINT_NAME);
        IExtension[] extensions = extensionPoint.getExtensions();
        for (int i = 0; i < extensions.length; i++) {
            IConfigurationElement[] elements = extensions[i].getConfigurationElements();
            for (int j = 0; j < elements.length; j++) {
                try {
                    Object listener = elements[j].createExecutableExtension("class");
                    if (listener instanceof TickerListener) {
                        listeners.add(listener);
                    } else { System.err.println("Not a Ticker Listener"); }
                } catch (CoreException e) { e.printStackTrace(); }
            }
        }
    }
}

```

## développement de plugins Eclipse

Extensions are the key mechanism that a

## Constructions autorisées

- Schema de point d'extension est un schéma XML valide (specification W3C)
- mais un sous-ensemble des constructions (conduisant à une correspondance 1-1 avec une DTD)
  - déclarations d'éléments globaux uniquement
  - déclarations d'attributs locaux uniquement
  - opérateurs de composition : **all, sequence, choice et group**
  - types d'attribut : **string, string + restriction enumeration, boolean**

```
<?xml version='1.0' encoding='UTF-8'?>
<!-- Schema file written by PDE -->
<schema targetNamespace="org.fho.ticker.org.fho.ticker">
<annotation>
  <appInfo>
    <meta.schema plugin="org.fho.ticker.org.fho.ticker" id="listeners" name="org.fho.ticker.listeners"/>
  </appInfo>
  <documentation>
    [Enter description of this extension point.]
  </documentation>
</annotation>

<element name="extension">
  <complexType>
    <sequence><element ref="tickerlistener"/></sequence>
    <attribute name="point" type="string" use="required">
      <annotation>
        <documentation></documentation>
        <appInfo><meta.attribute kind="java" /></appInfo>
      </annotation>
    </attribute>
    <attribute name="id" type="string">
```

## développement de plugins Eclipse

```
<annotation>
  <documentation>

  </documentation>
</annotation>
</attribute>
<attribute name="name" type="string">
  <annotation>
    <documentation>

    </documentation>
  </annotation>
</attribute>
</complexType>
</element>

<element name="tickerlistener">
  <complexType>
    <attribute name="class" type="string" use="required">
      <annotation>
        <documentation>

        </documentation>
      </annotation>
    </attribute>
    <attribute name="name" type="string" use="required">
      <annotation>
        <documentation>

        </documentation>
      </annotation>
    </attribute>
  </complexType>
</element>

<annotation>
  <appInfo>
    <meta.section type="since"/>
  </appInfo>
  <documentation>
    [Enter the first release in which this extension point appears.]
  </documentation>
</annotation>
```

## développement de plugins Eclipse

```
</annotation>

<annotation>
  <appInfo>
    <meta.section type="examples"/>
  </appInfo>
  <documentation>
    [Enter extension point usage example here.]
  </documentation>
</annotation>

<annotation>
  <appInfo>
    <meta.section type="apiInfo"/>
  </appInfo>
  <documentation>
    [Enter API information here.]
  </documentation>
</annotation>

<annotation>
  <appInfo>
    <meta.section type="implementation"/>
  </appInfo>
  <documentation>
    [Enter information about supplied implementation of this extension point.]
  </documentation>
</annotation>

<annotation>
  <appInfo>
    <meta.section type="copyright"/>
  </appInfo>
  <documentation>

  </documentation>
</annotation>
</schema>
```

### Constructions autorisées

- Sous-ensemble des constructions XML schema (conduisant à une correspondance 1-1 avec une DTD)

## Extension point schema editor

By convention, new schemas have the same name as the extension point id with a **.exsd** file extension. They are placed in **schema** directory in your plug-in directory tree.

### L'ouverture de l'éditeur

### Ajout d'un nouvel élément

### la documentation du schema

- Documentation about elements and attributes : definition page of the schema manifest (format HTML)
- Documentation about the extension point usage, API, etc.

