

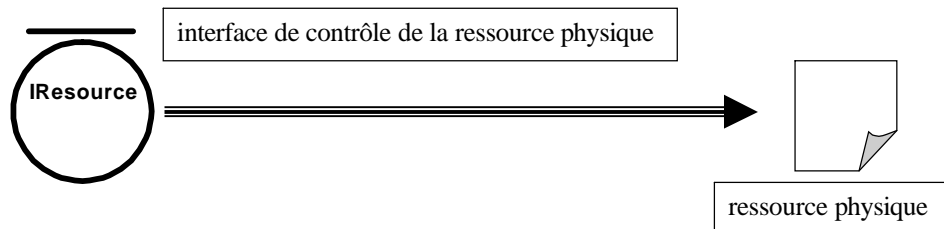
La gestion des ressources

LA GESTION DES RESSOURCES	1
CONTENU DE LA SECTION	2
QUELQUES GENERALITES (1).....	3
<i>quelques concepts</i>	3
<i>la vue physique</i>	3
QUELQUES GENERALITES (2).....	4
<i>la vue logique : interfaces et classes associées</i>	4
QUELQUES GENERALITES (3).....	5
<i>les interactions entre la vue physique et la vue logique</i>	5
<i>la désignation (l'interface IPath et la classe Path)</i>	5
L'ESPACE DE TRAVAIL	6
<i>la classe Workspace</i>	6
<i>quelques méthodes</i>	6
LE PROJET.....	7
<i>l'interface IProject et la classe Project</i>	7
<i>quelques méthodes</i>	7
<i>un exemple simple</i>	7
LES FICHIERS ET LES DOSSIERS	8
<i>quelques méthodes</i>	8
<i>exemples simples</i>	8

Quelques généralités (1)

quelques concepts

- le workspace : le conteneur **logique** des ressources accessibles aux plug-ins
- le project : unité de gestion du workspace
- les ressources : entités du système de fichiers associées à un modèle d'événement et de marqueurs
- les propriétés : meta-données associées à une ressource
- les markers : méta-données particulières pour le marquage des ressources



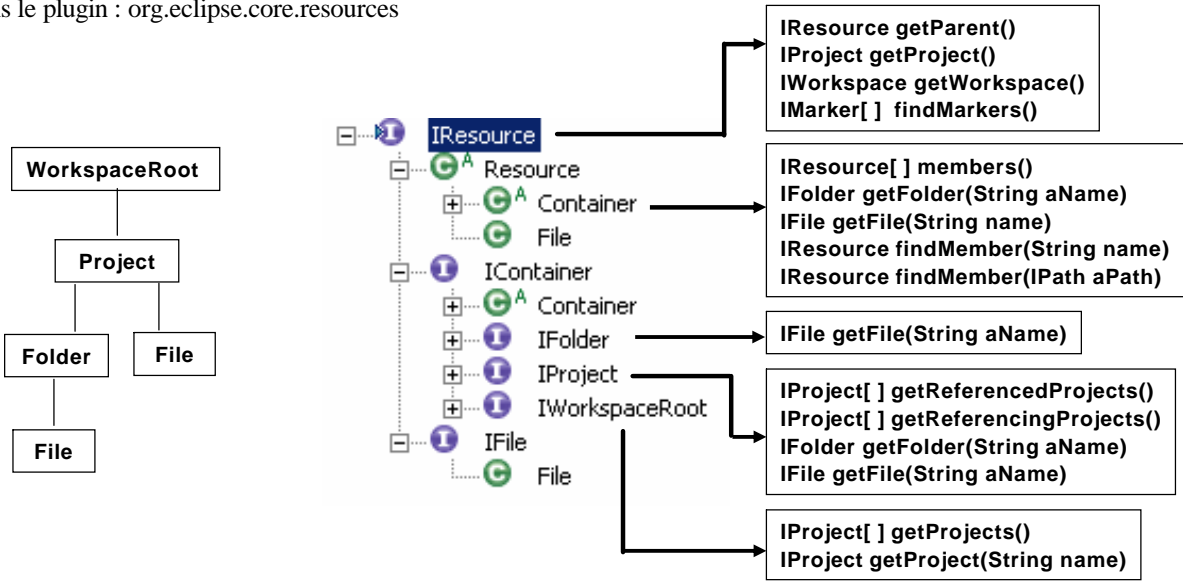
la vue physique

- un workspace, un project : une arborescence du système de fichiers définie par un répertoire et son contenu
- le répertoire du workspace contient les répertoires associés aux projets
- mécanisme de liens permettant d'accéder aux projets situés dans d'autres répertoires

Quelques généralités (2)

la vue logique : interfaces et classes associées

- définie dans le plugin : org.eclipse.core.resources



Quelques généralités (3)

les interactions entre la vue physique et la vue logique

- les vues ne sont pas systématiquement synchronisées
 - ➔ aucune garantie que la ressource désignée par un handle existe effectivement [test]
- chargement de la vue logique à partir de l'état sauvegardé
- mise à jour des modifications externes du système de fichier par `IResource.refreshLocal()`
 - ➔ **Window > Preferences > Workbench > Refresh Workspace on startup**
- le répertoire `.metadata` contient les données associées aux préférences et propriétés des projets
 - il contient un répertoire `.plugin` : les données associées aux plug-ins
 - il contient un répertoire `.config` : les données de configuration associées au workspace
- le fichier `.project` contient les informations concernant les ressources

la désignation (l'interface `IPath` et la classe `Path`)

- `full path` : désignation relative au workspace
- `project relative path` : relative au projet
- `location` : relative au système de fichier (désignation absolue)
- nombreuses méthodes de manipulation : test, analyse, modification

L'espace de travail

la classe `Workspace`

- points d'entrée aux ressources via `WorkspaceRoot`
- accès à la description du workspace
- gestion de listeners pour les modifications des ressources associées au workspace (`ResourceChangeListener`, `LifeCycleListener`)

Workspace n'implémente PAS **IResource** contrairement à **WorkspaceRoot**

quelques méthodes

```
getDescription() //retourne la description du workspace
IStatus validateName(String name, type) //retourne true si le nom est valide pour le workspace
addResourceChangeListener(ResourceChangeListener In) //enregistre un listener de modification de ressources
```

l'interface IProject et la classe Project

- accès à la description du projet (sauvegardée dans le fichier .project)
- accès à la nature du projet

quelques méthodes

```
getDescription() //retourne la description du workspace
IStatus validateName(String name, type) //retourne true si le nom est valide pour le workspace
addResourceChangeListener(ResourceChangeListener ln) //enregistre un listener de modification de ressources
```

un exemple simple

```
IWorkspace wspace = ResourcesPlugin.getWorkspace() ;
IWorkspaceRoot wspaceRoot = wspace.getRoot() ;
IProject project = wspaceRoot.getProject("myProject") ;
IPath projectPath = new Path("c:/workspace" + project.getName());

final IProjectDescription description = wspace.newProjectDescription(project.getName());
try {
    project.create(description, null);
    project.open(null);
} catch(CoreException ex) { ..... }
```

Création dans le répertoire par défaut associé au workspace

quelques méthodes

- méthodes génériques
- famille de méthodes : `void setContents(InputStream stream,)` associant/synchronisant un handle avec la ressource

exemples simples

```
private void createFolderInProject(IProject proj, String fName) {
    IFolder nFolder = proj.getFolder(fName);
    if (!nFolder.exists()) {
        try {
            nFolder.create(true, true, null);
        } catch(CoreException ex) { ..... }
    }
}

private void createFileInFolder(IFolder folder, String fName, InputStream stream) {
    IFile nFile = folder.getFile(fName);
    try {
        if (nFile.exists()) nFile.setContents(stream, true, false, null);
        else {
            java.io.File sFile = nFile.getLocation().toFile();
            if (sFile.exists()) sFile.create(stream, false, null);
        }
    } catch(CoreException ex) { ..... }
}
```