12 - Eclipse Builders, Natures and Markers

In this chapter we will talk about builders, natures and markers.

Builders: If you are using Eclipse IDE for java development then you are not new to Builders. Inside Eclipse builders are used to take java source as input and generate corresponding class files. So you can define builders as an engine which generates some output based on your input. Incremental Builders - These contribute to performance improvement by rebuilding only the delta since last full build.

Markers: Markers are used mark locations in editor. Java editor uses these markers to mark error locations, warnings, book marks etc.

Natures: Natures are used to link project with various functionalities such as builders in a project. For ex: if we add java nature to a project inside eclipse. It associates project with incremental java compiler.

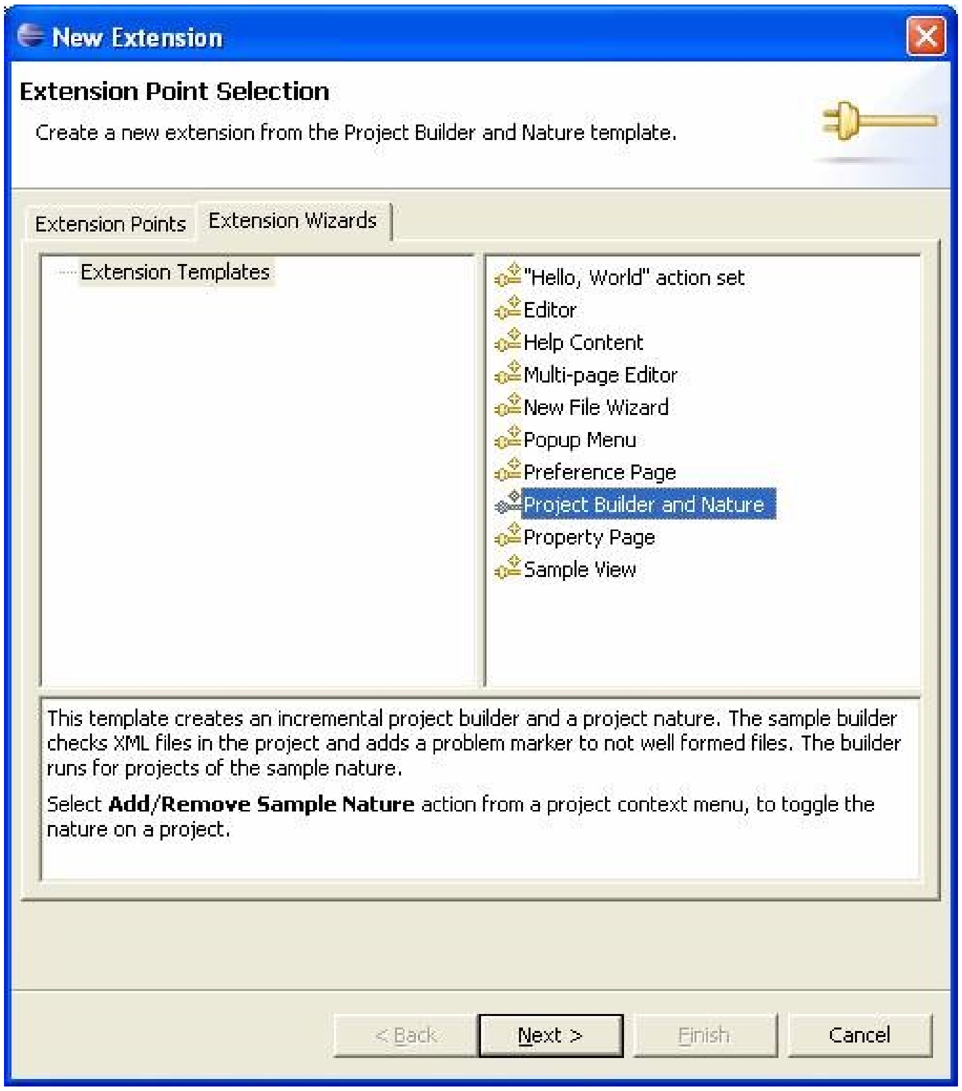
Creating Builders Nature using existing templates

In order to understand builder nature and marker concepts we will use one of the templates available in eclipse itself. This template will create an builder (incremental). In addition, it will also create a project nature. Name of this builder is "sample builder" and it basically validates XML file and if required adds a problem marker to problematic xml files. The builder will run only for projects which have declared themselves as having sample nature projects. Select Add/Remove Sample Nature action from a project context menu, to toggle the nature on a project.

C H A P T E R S

O T H E R

S U B J E C T S



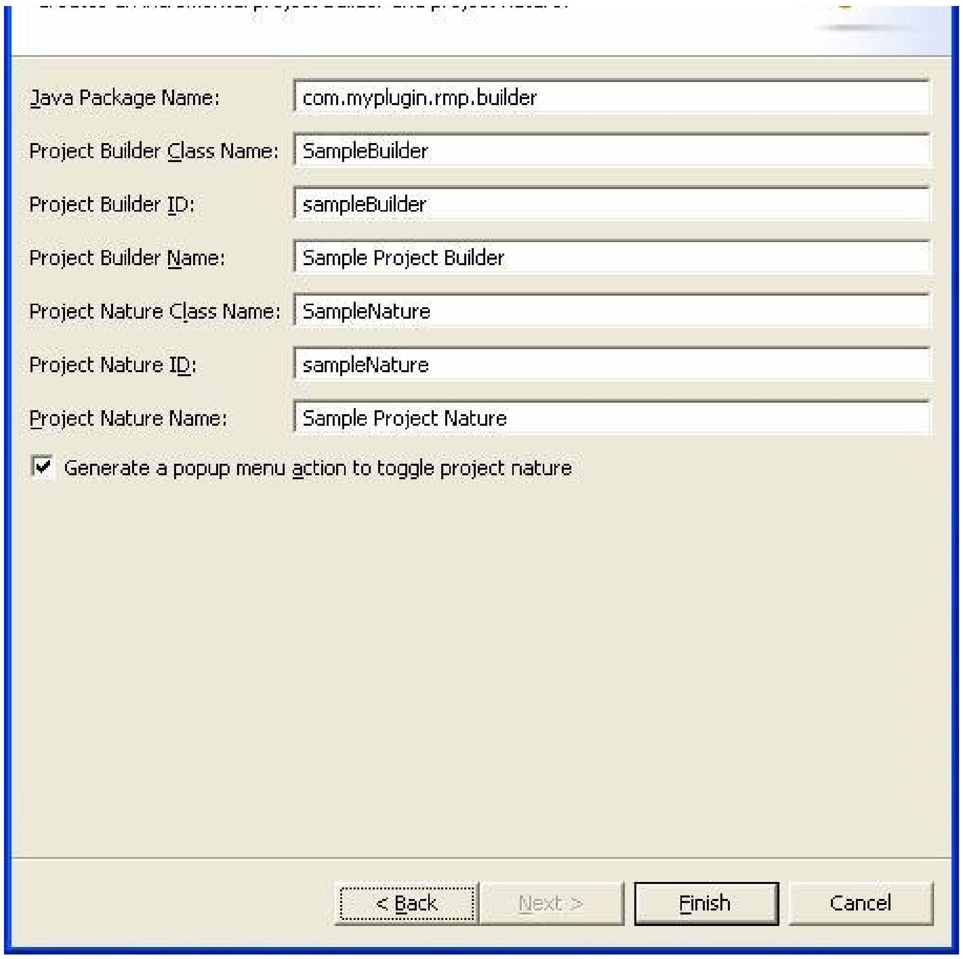
This template creates an incremental project builder and a project nature. The sample builder checks XML files in the project and adds a problem marker to not well formed files. The builder runs for projects of the sample nature.

Select **Add/Remove Sample Nature** action from a project context menu, to toggle the nature on a project.

Click next and use default values as shown below

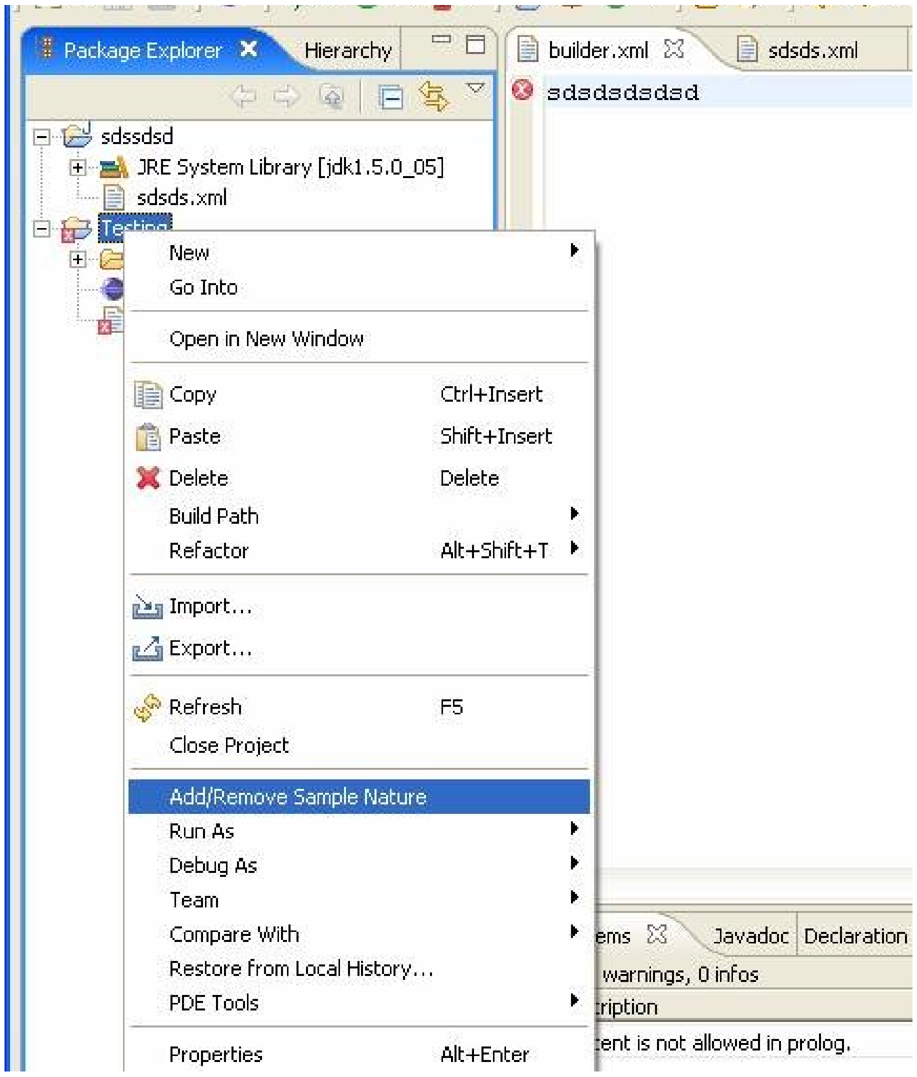
CHAPTERS

OTHER SUBJECTS



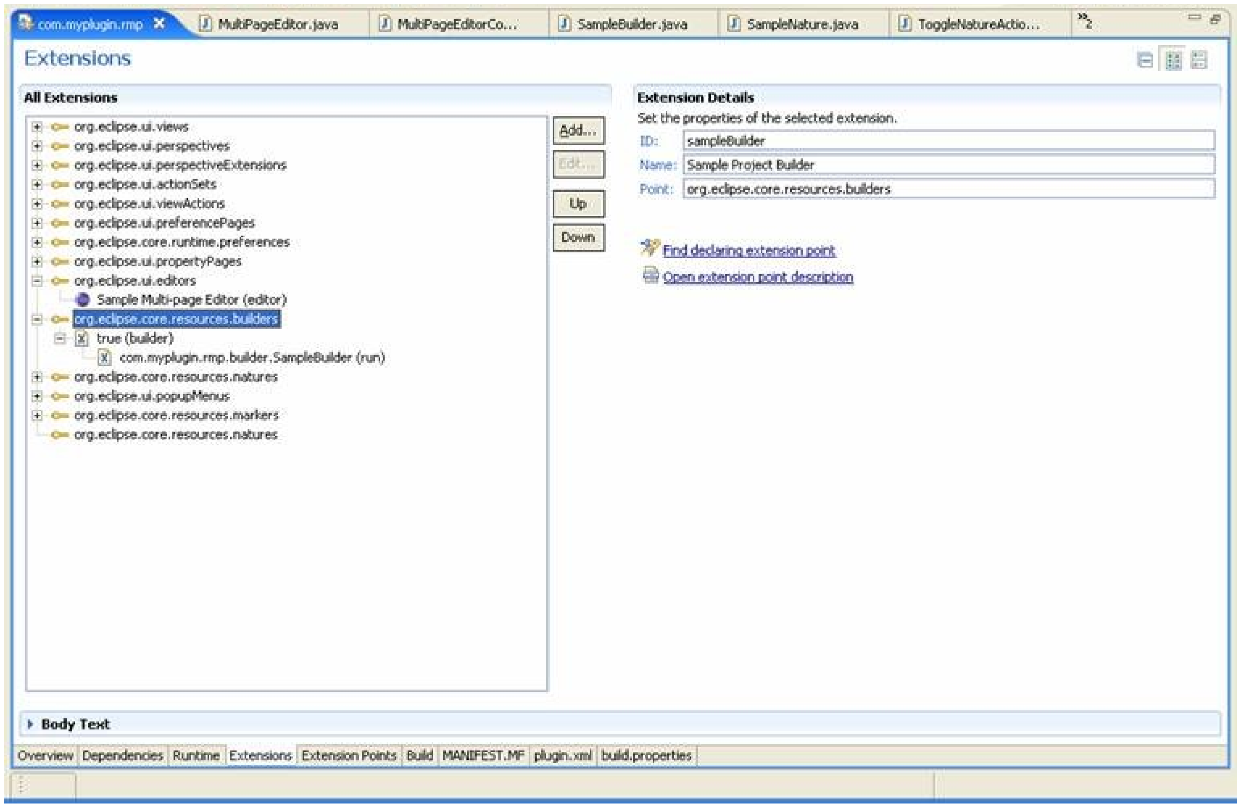
Java Package Name: com.myplugin.rmp.builder
Project Builder Class Name: SampleBuilder
Project Builder ID: sampleBuilder
Project Builder Name: Sample Project Builder
Project Nature Class Name: SampleNature
Project Nature ID: sampleNature
Project Nature Name: Sample Project Nature
[✓] Generate a popup menu action to toggle project nature

Click finish and save the plug-in manifest file. Before we move onto discuss the generated code let's first see the builder in action. Launch eclipse runtime application. Create a new xml file in existing project (create new project if workspace is new). Edit newly created xml file such that it is not well formed. Right click on project and click Add/Remove sample Nature. Notice that the marker will appear inside the editor (as shown in figure below).



Eclipse Builders

Let's review the plug-in manifest file to see what it takes to create a new project builder. New builder is created by extending the org.eclipse.core.resources.builders extension point. ID and name are used to give builder a data value and name respectively. Point is nothing but the extension point which we have extended.



Extensions

All Extensions

```
org.eclipse.ui.views
org.eclipse.ui.perspectives
org.eclipse.ui.perspectiveExtensions
org.eclipse.ui.actionSets
org.eclipse.ui.viewActions
org.eclipse.ui.preferencePages
org.eclipse.core.runtime.preferences
org.eclipse.ui.propertyPages
org.eclipse.ui.editors
    Sample Multi-page Editor (editor)
org.eclipse.core.resources.builders
    true (builder)
        com.myplugin.rmp.builder.SampleBuilder (run)
org.eclipse.core.resources.natures
org.eclipse.ui.popupMenus
org.eclipse.core.resources.markers
org.eclipse.core.resources.natures
```

Extension Details

Set the properties of the selected extension.

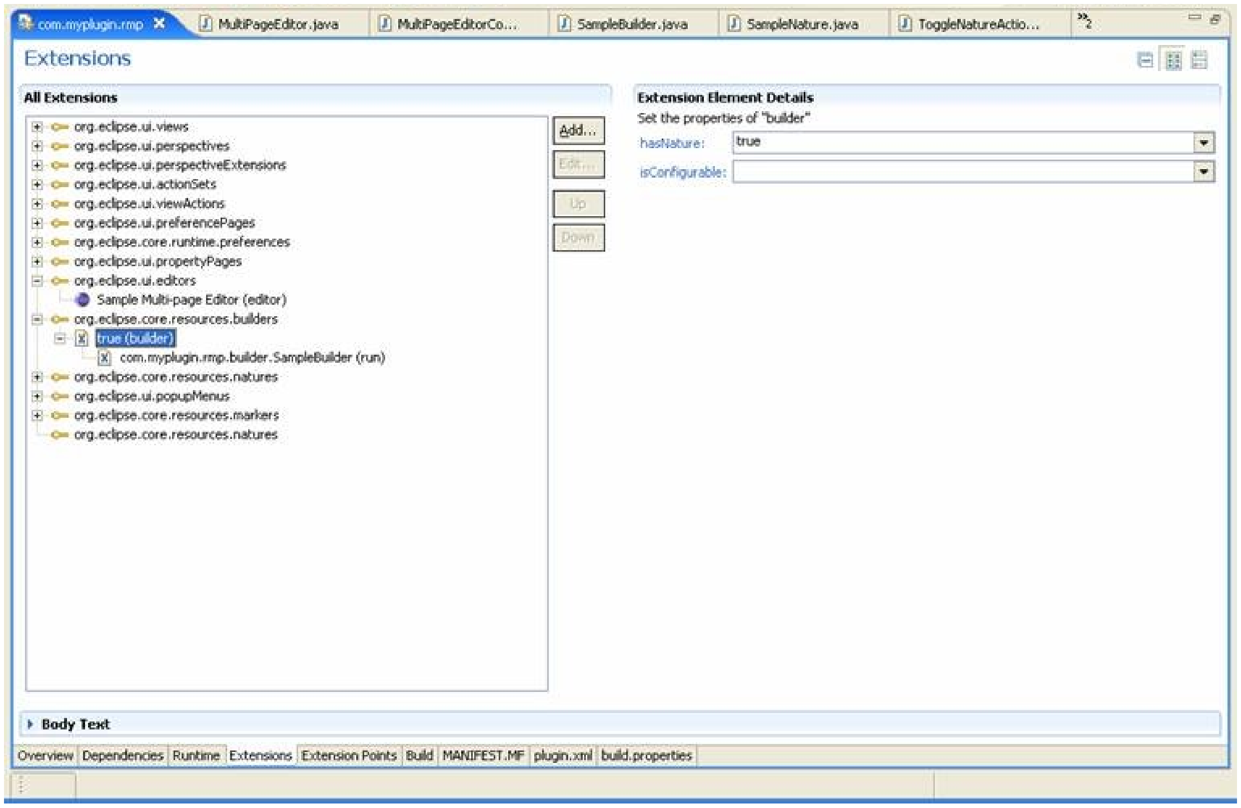
ID: sampleBuilder
Name: Sample Project Builder
Point: org.eclipse.core.resources.builders

Next, is the builder element. The hasNature attribute flags whether this builder is associated with any nature.

C H A P T E R S

O T H E R

S U B J E C T S



```
org.eclipse.ui.views
org.eclipse.ui.perspectives
org.eclipse.ui.perspectiveExtensions
org.eclipse.ui.actionSets
org.eclipse.ui.viewActions
org.eclipse.ui.preferencePages
org.eclipse.core.runtime.preferences
org.eclipse.ui.propertyPages
org.eclipse.ui.editors
    Sample Multi-page Editor (editor)
org.eclipse.core.resources.builders
    true (builder)
        com.myplugin.rmp.builder.SampleBuilder (run)
org.eclipse.core.resources.natures
org.eclipse.ui.popupMenus
org.eclipse.core.resources.markers
org.eclipse.core.resources.natures
```

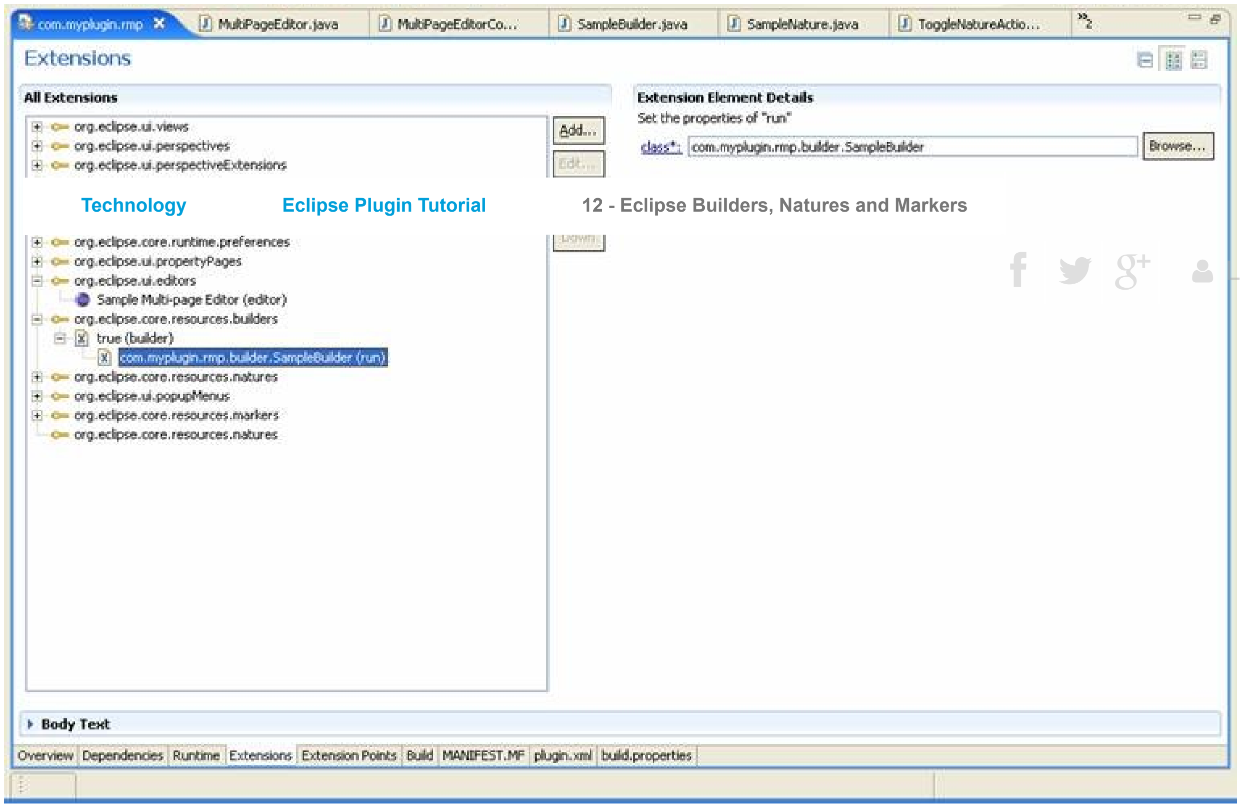
Extension Element Details

Set the properties of "builder"

hasNature: true

isConfigurable:

Next, is the run element (class attribute). Here we specify the actual java class which provides the functionality of the builder.



org.eclipse.ui.views
org.eclipse.ui.perspectives
org.eclipse.ui.perspectiveExtensions
org.eclipse.core.runtime.preferences
org.eclipse.ui.propertyPages
org.eclipse.ui.editors
Sample Multi-page Editor (editor)
org.eclipse.core.resources.builders
true (builder)
com.myplugin.rmp.builder.SampleBuilder (run)
org.eclipse.core.resources.natures
org.eclipse.ui.popupMenus
org.eclipse.core.resources.markers
org.eclipse.core.resources.natures

class*: com.myplugin.rmp.builder.SampleBuilder

IncrementalProjectBuilder

The java class (com.myplugin.rmp.builder.SampleBuilder) discussed above needs more attention as this is the class which is doing the actual work. Any builder class must be a subclass of IncrementalProjectBuilder. All builders must subclass this class according to the following guidelines:

- Subclasses should implement the build method of this class
- Subclasses may implement remaining methods
- Subclasses should supply a public and no arg constructor

Please refer to **Eclipse API Specification** to find details of all the methods provided by this Class.

The build method requires special attention here. This method of the builder is called whenever build is required (for ex: user saves java file and automatic build preference is checked)

build(int kind, Map args, IProgressMonitor monitor)

kind: What kind of build is required?? Kind parameter could be a FULL_BUILD, INCREMENTAL_BUILD, and AUTO_BUILD.

Let's review generated code related to builders

In this section we will review generated class (com.myplugin.rmp.builder.SampleBuilder) and will discuss builder/marker related code segments.

We will not go into details of IResourceDelta, IResourceDeltaVisitor or IResourceVisitor in this chapter. For more details on these topics refer to resource change tracking chapter.

```
protected IProject[] build(int kind, Map args, IProgressMonitor ?
                if (kind == FULL_BUILD) {
                        fullBuild(monitor);
                } else {
                        IResourceDelta delta = getDelta(getProjec
                        if (delta == null) {
                                fullBuild(monitor);
                        } else {
                                incrementalBuild(delta, monitor
                        }
                }
                return null;
}
```

Eclipse will call build method of the builder whenever any type of build is required. In above code segment we are using kind parameter to understand the kind of build request. We have implemented two methods for each kind of build full build/ incremental build discussed next.

```
protected void fullBuild(final IProgressMonitor monitor) throws ?
                try {
                        getProject().accept(new SampleResourceVis
                } catch (CoreException e) {
                }
}

protected void incrementalBuild(IResourceDelta delta, IProgressMo
                                throws CoreException {
                // the visitor does the work.
                delta.accept(new SampleDeltaVisitor());
}
```

In both of above methods we are delegating calls to Visitors. Visitor will visit all the resources (in case of delta only the modified resources since last build). For each resource we are calling checkXML() method. CheckXML method essentially delegates call to XMLErrorHandler defined in builder class which will check if XML document is well formed or not. In case it is not well formed it will create a marker (as shown below). I am not going into details of SAX parsing or SAX event handlers (it is out of scope for this tutorial). We

C H A P T E R S

```
private void addMarker(SAXParseException e, int severity) {
                    SampleBuilder.this.addMarker(file, e.getMe
                                            .getLineNumber(), sever
}
```

1 2 3 next

O T H E R S U B J E C T S

TOC

11 - Eclipse Plugin Editors Tutorial

13 - Eclipse Plugin Help Tutorial

Like us on Facebook

Wideskills