CONSUMING an EMF MODEL in UR ECLIPSE UI
PART 1

There are lots of good article available on EMF Model Generation / Notification and Serialization.
One of my favourites : Modelling Section in http://www.vogella.com/eclipse.html

But what I realize isn't available is a clear documentation on how to consume these awesome EMF Models into your own Eclipse UI. I see a lot of fear in the developers eyes during my trainings to use the .edit and .editor plugin as at its 1st look the code does look scary.

Trying to explain the .editor plugin in training, a lot of people get lost and not sure which piece of code is mandatory or useful and which is optional. Therefore I did decide to document a small article on how to consume or use the generated EMF model in my VIEW using the ItemProviders generated by the .edit plugin.

Assuming that you already know how to create your model and edit plugin from your existing ecore, lets get to our business of displaying it on the UI. Model that I am using to showcase this usecase is as shown below in fig.1

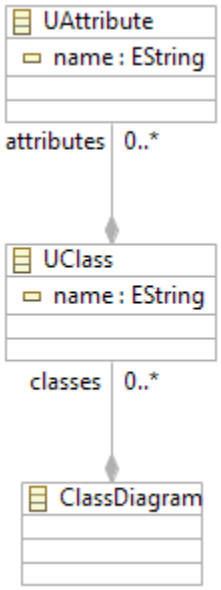

Fig 1 : Model used in this Tutorial

Author : Annamalai
February 7, 2013

ANCIT CONSULTING

Steps to follow

1. Create a View and place a TreeViewer on it.
   Your createPartControl method would look something like below.

```
@Override
public void createPartControl(Composite parent) {
    TreeViewer tViewer = new TreeViewer(parent);
    tViewer.setContentProvider(provider);
    tViewer.setLabelProvider(labelProvider);
    tViewer.setInput(input);
}
```

Fig 2 : Code Snippet after Step 1

2. Your viewer needs to be provided with Content and Label Provider. But in our case we do not have Content and LabelProvider but we have Item Providers generated and available in EMF .edit plugin.

   These ItemProviders are accessible thru the ItemProviderFactory present in the .edit plugin. Therefore the ItemProviderFactory needs to be adapted to behave like a Content and Label Provider. EMF provides these Adapter classes for Content and Label Provider to do the needful.

   These classes are namely, AdapterFactoryContentProvider and AdapterFactoryLabelProvider.

```
@Override
public void createPartControl(Composite parent) {
    ClassDiagram cDiagram = createModel();

    TreeViewer tViewer = new TreeViewer(parent);
    tViewer.setContentProvider(new AdapterFactoryContentProvider());
    tViewer.setLabelProvider(new AdapterFactoryLabelProvider());
    tViewer.setInput(cDiagram);
}
```

Fig 3 : Code Snippet after Step 2

3. We need to wrap our ItemProviderAdapterFactory into these classes.

```
tViewer.setContentProvider(new AdapterFactoryContentProvider(
        new CdiagramItemProviderAdapterFactory()));
tViewer.setLabelProvider(new AdapterFactoryLabelProvider(
        new CdiagramItemProviderAdapterFactory()));
```

Fig 4: Code Snippet after Step 3

4. Run your Code. And Open your View that you just created in Step 1. You should see a TreeViewer showing your Model Structure.

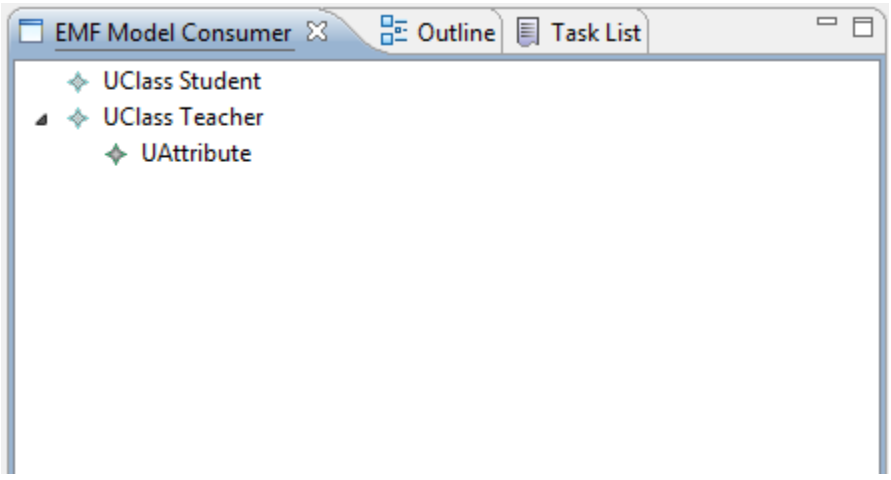

Fig 5: Screenshot after Step 4.