

EMF Parsley: a toolbox for UI parts built on EMF

Lorenzo Bettini

Dipartimento di Informatica, Università di Torino, Italy
itemis Schweiz Consultant

joint work with **Francesco Guidieri** and **Vincenzo Caselli**
RCP-Vision

EclipseDay Florence 2013



Boilerplate code

```
adapterFactory = new ComposedAdapterFactory
    (ComposedAdapterFactory.Descriptor.Registry.INSTANCE);
adapterFactory.addAdapterFactory(new ResourceItemProviderAdapterFactory());
adapterFactory.addAdapterFactory(new MyModelItemProviderAdapterFactory());
adapterFactory.addAdapterFactory(new ReflectiveItemProviderAdapterFactory());

BasicCommandStack commandStack = new BasicCommandStack();
commandStack.addCommandStackListener
    (new CommandStackListener() {...});

editingDomain = new AdapterFactoryEditingDomain(adapterFactory, commandStack, ...);

Tree tree = new Tree(composite, SWT.MULTI);
TreeViewer viewer = new TreeViewer(tree);

viewer.setContentProvider(new AdapterFactoryContentProvider(adapterFactory));
viewer.setLabelProvider(new AdapterFactoryLabelProvider(adapterFactory));
viewer.setInput(editingDomain.getResourceSet());

new AdapterFactoryTreeEditor(viewer.getTree(), adapterFactory);
```

Customizing Generated Code

Follow the @generated pattern:

```
public class BookItemProvider extends ... {  
    /** @generated NOT */  
    @Override  
    public Object getImage(Object object) {  
        return overlayImage(object,  
            getResourceLocator().getImage("mypath/custom_book.png"));  
    }  
  
    /** @generated NOT */  
    @Override  
    public String getText(Object object) {  
        return "Book: " + " " + ((Book)object).getTitle();  
    }  
}
```

Emf Parsley

- A lightweight framework that allows easy and quick development of applications based on EMF.
- It can be configured to use all kinds of EMF persistence implementations (e.g., XMI, Teneo, CDO).
- It aims at providing a set of reusable components like trees, tables and detail forms that manage the model with the introspective EMF capabilities.
- Using these components one can easily build more complex widgets, editors and applications.
- The framework provides basic UI implementations which are customizable with injection mechanisms.
- A **DSL** to make customizations much easier.

Small Reusable and Customizable Blocks

- Split and delegate responsibilities into small classes.
- Easy to customizing a single aspect without customizing the components themselves.
- The custom version will be **injected** in the framework (Google Guice).
- All components relying on that aspect will be assured to use our specific version.

Specify Custom Bindings

A custom reflective way of specifying bindings (inspired by Xtext)

```
public class MyEmfComponentsModule extends EmfComponentsGuiceModule {  
  
    public Class<? extends ResourceLoader> bindResourceLoader() {  
        return MyResourceLoader.class;  
    }  
  
    public Class<? extends EmfViewerMouseAdapter> bindEmfViewerMouseAdapter() {  
        return MyEmfViewerMouseAdapter.class;  
    }  
  
    public Class<? extends FeatureLabelProvider> bindFeatureLabelProvider() {  
        return MyFeatureLabelProvider.class;  
    }  
  
}
```

What we provide

- Viewers and Composites:
 - TreeViewer, TableViewer, FormComposite
- Editors
- Views
 - Saveable view (based on Resource)
 - Based on selection

Create a TreeViewer

```
public class MyView extends ViewPart {  
  
    @Inject ViewerInitializer initializer;  
  
    @Override  
    public void createPartControl(Composite parent) {  
        ...  
        viewer = new TreeViewer(parent, ...);  
        // initialize with an EMF Resource  
        initializer.initialize(viewer, resource);  
        // or alternatively, if you have an EObject  
        initializer.initialize(viewer, eObject);  
    }  
}
```


Form Composites

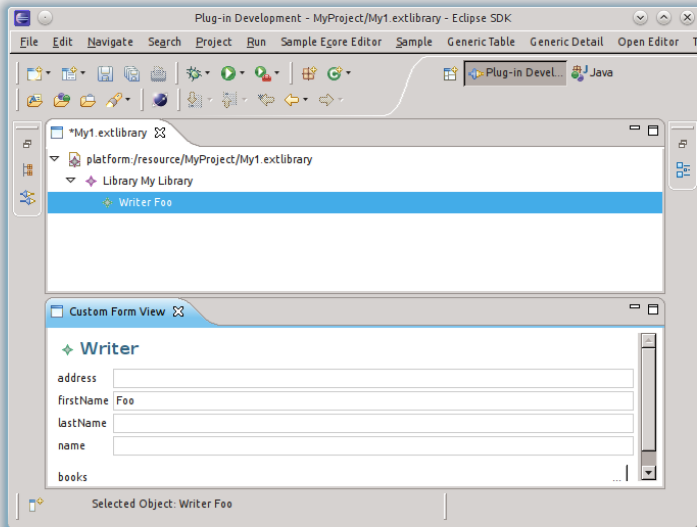
```
public class MyView extends ViewPart
    implements ISelectionChangedListener {

    @Inject FormFactory factory;

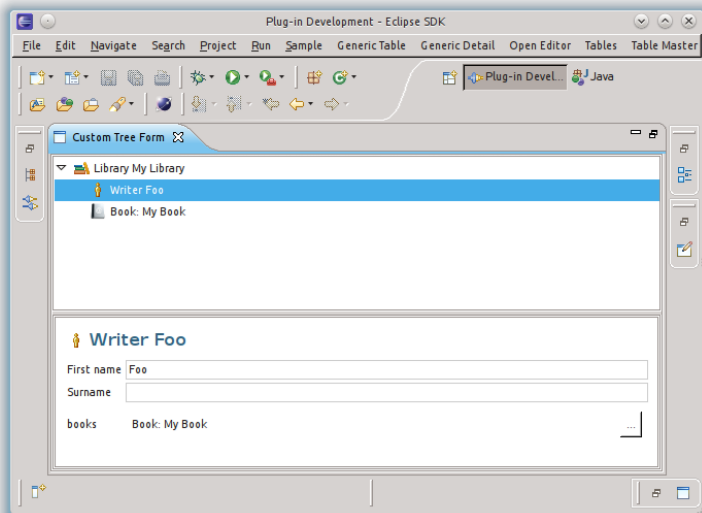
    public void selectionChanged(SelectionChangedEvent event) {
        EObject selectedObject = getFirstSelectedEObject
            (event.getSelection());

        // relevant lines
        FormDetailComposite detailForm = factory.
            createFormDetailComposite(parent, SWT.BORDER);
        detailForm.init(selectedObject);
    }
}
```

Using Components



Using Components



Declarative Customization

```
public class CustomLabelProvider extends ViewerLabelProvider {  
    public String text(Book book) {  
        return "Book: " + book.getTitle();  
    }  
  
    public String image(Book book) {  
        return "book.png";  
    }  
  
    public String text(Borrower b) {  
        return "Borrower: " + b.getFirstName();  
    }  
}
```

Methods selected at run-time with polymorphic dispatch according to the run-time type of the argument.

Custom Bindings

```
public class MyCustomModule extends EmfComponentsGenericModule {  
    public Class<? extends CompositeLabelProvider>  
        bindCompositeLabelProvider() {  
        return CustomLabelProvider.class;  
    }  
    ...  
}
```

A DSL implemented in Xtext/Xbase

```
Java - my.emf.component.project/src/my.emf.component.project/module.emfcomponents - Eclipse SDK
File Edit Navigate Search Project Run Sample Menu Window Help
EFS TEST
Quick Access
Package Explorer
my.emf.component.project
  src
    my.emf.component.project
      Activator.java
      EmfComponentsGuiceModule.java
      ExecutableExtensionFactory.java
      module.emfcomponents
    emfcomponents-gen
      my.emf.component.project
        EmfComponentsGuiceModuleGen.java
      my.emf.component.project.ui.provider
        EStructuralFeaturesProviderGen.java
        FeatureLabelProviderGen.java
        LabelProviderGen.java
  JRE System Library [J2SE-1.5]
  Plug-in Dependencies
  META-INF
  MANIFEST.MF
  build.properties
module.emfcomponent
import java.util.*
import it.rcpvision.emf.components.examples.library.*

/* my.emf.component.project Emf Components Dsl Module file */
module my.emf.component.project {
  featureProvider {
    features {
      Library -> name, address
      Person -> firstName, lastName
    }
  }

  labelProvider {
    labels {
      Library lib ->
      {
        "Library: " +
        if (lib.name != null)
          lib.name.toFirstUpper
        else
          "Not specified"
      }
      Writer writer -> writer.firstName + " " + writer.lastName
    }
  }

  imageProvider {
    Library -> 'lib.gif'
    Person -> 'Person.jpeg'
  }
}

featureLabelProvider {
```

Tooling fully integrated with Java

The screenshot shows the Eclipse IDE with a project named "my.emf.component.project". The Package Explorer on the left shows the project structure, including the "module.emfcomponents" package. The main editor displays the DSL code for the "module.emfcomponent" package:

```

Library -> name, address
Person -> firstName, lastName
}

LabelProvider {
  labels {
    Library lib ->
    {
      "Library: " +
      if (lib.name != null)
        lib.name.toFirstUpper
      else
        'Not specified'
    }
  }
  Writer writer -> writer.
  + " " + writer.lastName
}
  
```

A tooltip is visible over the `address` attribute, showing the following information:

- `String`
- `it.rcpvision.emf.components.examples.library.Addressable.getAddress()`
- Returns the value of the 'Address' attribute.
- Returns: the value of the 'Address' attribute.
- See Also: [setAddress\(String\)](#), [it.rcpvision.emf.components.examples.library.EXTLibraryPa](#)
- @model
- @generated

The bottom status bar shows "Writable", "Insert", and "23 : 37".

More Static Checks

```
featureLabelProvider {  
  Labels {  
    Person : firstName -> "Name"  
    Person : lastName -> "Surname"  
  }  
}
```

Problems 3 errors, 2 warnings, 0 others

Description	Resource	Path
Couldn't resolve reference to JvmIdentifiableElement 'firstName'.	module.emfco	/my.er

Works with RAP

RAP Single View - Mozilla Firefox

File Edit View History Bookmarks Tools Help

RAP Single View

127.0.0.1:47934/view?startup=it.rcpvision.emf.components.examples.rap.u

Google

Most Visited Getting Started Latest Headlines my del.icio.us Planet Linux Mint Community Forums Blog

RAP with a View

File

View

- Model: My Model
 - Element: First Element (1)
 - Element: Second Element (2)
 - Element: Third Element (3)
 - Item First Item
 - Item Second Item
 - Item Third Item
 - Item Fourth Item

Item Third Item

name Third Item

*Saveable View

- Model: My Model
 - Element: First Element (1)
 - Element: Second Element (3)
 - Element: Third Element (3)
 - Item First Item
 - Item Second Item
 - Item Third Item
 - Item Fourth Item

Element: Second Element (2)

name Second Element

age 3

Items Item First Item

Links

Eclipse project proposal:

<http://eclipse.org/proposals/modeling.emfparsley/>

https://bugs.eclipse.org/bugs/show_bug.cgi?id=400609

Current implementation:

[http:](http://code.google.com/a/eclipseorg/p/emf-components/)

[//code.google.com/a/eclipseorg/p/emf-components/](http://code.google.com/a/eclipseorg/p/emf-components/)

Discussion thread:

<http://www.eclipse.org/forums/index.php/t/457873/>

Links

Eclipse project proposal:

<http://eclipse.org/proposals/modeling.emfparsley/>

https://bugs.eclipse.org/bugs/show_bug.cgi?id=400609

Current implementation:

<http://code.google.com/a/eclipseorg/p/emf-components/>

[//code.google.com/a/eclipseorg/p/emf-components/](http://code.google.com/a/eclipseorg/p/emf-components/)

Discussion thread:

<http://www.eclipse.org/forums/index.php/t/457873/>

Thanks!