Six good reasons to spice up your projects with EMF Parsley

Francesco Guidieri
Vincenzo Caselli

EclipseCon Europe 2016
Ludwigsburg - October 27, 2016
About us

Lorenzo Bettini
Dip. Informatica, Univ. Firenze, Italy
bettini@disia.unifi.it
@lorenzo_bettini
www.lorenzobettini.it

Vincenzo Caselli
RCP Vision
vincenzo.caselli@rcp-vision.com
@vcaselli
www.rcp-vision.com

Francesco Guidieri
RCP Vision
francesco.guidieri@rcp-vision.com
@fraguid
www.rcp-vision.com
What is EMF Parsley?

Parsley provides an easy way to get a User Interface for EMF models:

• with built-in UI components

• easy customizations

• with a powerful DSL
Six good reasons

• very lightweight and straightforward to use

• out-of-the-box integrations with XMI, Teneo and CDO

• dependency Injection (via Google Guice) instead of Extension Point mechanism

• Eclipse/JDT integrated DSL with a static type system

• code-generation oriented (debuggable!) instead of a reflective approach (no bad runtime surprises)

• out-of-the-box integration with RAP for a unified Web / Desktop development process and know-how

www.eclipse.org/emf-parsley
Parsley Components

Basic components can be used out of the box:

- Tree
- Table
- Form
- Tree Form
- Table Form
- Dialog
... and more!

- All components are provided as Java APIs
- There are some views as reference implementation
- Project Wizards with initial templates
- Many examples to import
- A rich Eclipse tooling
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);
Instead in EMF Parsley the code is very simple

```java
@Inject ViewerFactory viewerFactory;

(...)

treeViewer = new TreeViewer(parent);

viewerFactory.initialize(treeViewer, element);
```
Table code example

```java
@Inject ViewerFactory viewerFactory;

(...)

tableViewer = viewerFactory.createTableViewer(composite, SWT.BORDER | SWT.FULL_SELECTION, object, eClass);
```

<table>
<thead>
<tr>
<th>publicationDate</th>
<th>copies</th>
<th>title</th>
<th>pages</th>
<th>category</th>
<th>author</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>First Book</td>
<td>100</td>
<td>Mystery</td>
<td>Writer A Writer</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Second Book</td>
<td>100</td>
<td>Mystery</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Third Book</td>
<td>100</td>
<td>Mystery</td>
<td></td>
</tr>
</tbody>
</table>
Form code example

```java
@Inject FormFactory formFactory;

(...)

formComposite = formFactory.createFormDetailComposite(parent, SWT.NONE);
formComposite.init(eObject);
```
First demo

• Import First Example

• Project structure
  • Dsl and generated code

• Run
  • Show runtime features
How it works

• Parsley UI is built on top of EMF>Edit reflective framework,
  • the default behavior is delegated to EMF Edit
  • or your .edit project

• Customizations are based on Dependency Injection:
  • no need to inherit parts and composites
  • change single behaviors / aspects
  • inject them in the runtime
  • purely Java
EMF Parsley DSL

• Implemented in Xtext, using Xbase
  • Interoperable with the Java type system
  • IDE tooling (including Debugging)

• Specify customizations in one single file in a compact form
  • Generates the corresponding Java code
  • Generates the Guice bindings

• You can use the DSL and manually written Java code together
Static type checking

All expressions are statically type-checked / inferred
DSL Java interoperability

- Access to all Java types
- According to project dependencies/classpath
- DSL elements can extend your Java classes
- DSL fully debuggable!
Imports management

- Automatic import
- Quick fixes

```java
tableLabelProvider {
  text {
    MailMessage: date -> {
      new SimpleDateFormat("dd/MM/yyyy")
    }
  }
}
```

Could't resolve reference to JvmConstructor 'SimpleDateFormat'.

2 quick fixes available:

- Import 'SimpleDateFormat' (java.text)
- Create Java class 'SimpleDateFormat'

www.eclipse.org/emf-parsley
JDT Integration

```kotlin
labelProvider {
    text {
        Book b -> {'"' + b.title + '"'}
        Writer w -> { w.name }
    }
}

menuBuilder {
    val factory = EXTLibrary

    emfMenus {
        Writer w -> [
            actionChange("New",
                library
                val book
                library
                Books -> Book

            SetTitle(String)
            org.eclipse.emf.examples.extlibrary.EXTLibraryPackage.getBook_Title()
        @model
        @generated
        }
    }
}
```

A representation of a Uniform Resource Identifier (URI), as specified by RFC
All artifacts are generated and kept in synch:
• Java implementations
• plugin.xml

• Fully integrated with Eclipse Building mechanism
  • All files are generated on DSL save
Second Demo: Debug
Customizations

There are many levels of customizations:

• EMF Edit – the default behaviour

• DSL customizations – the easy way

• Google Guice / Java – not only for the brave! 😊
Default behavior is delegated to EMF.Edit
Tree content customization

viewerContentProvider {
    children {
        Library -> {
            writers + books
        }
    }
}

- Library My Library
  - Writer Lorenzo Bettini
    - Book A story
    - Book another story
    - Book
    - Book

- Book A story
  - Publication Date
  - Copies: 100
  - Borrowers
  - Title: A story
  - Pages: 100
  - Category: Mystery
  - Author: Writer Lorenzo Bettini

Default | Content | Features | Label | Menu 1 | Menu 2
Feature customizations

```java
featuresProvider {
    features {
        Library -> name
        Writer -> name, address, books
        Book -> author, title, category
    }
}
```

- **Library My Library**
  - **Writer Lorenzo Bettini**
    - **Book A story**
    - **Book another story**
    - **Book**
    - **Book**

- **Book A story**
  - **Author**: Writer Lorenzo Bettini
  - **Title**: A story
  - **Category**: Mystery
Label and caption customizations

```
featureCaptionProvider {
  text {
    Book : author -> "Written by:"
    Writer : name -> "Name:"
  }
}

labelProvider {
  text {
    Book b -> {
      "" + b.title ?: "untitled" + ""
    }
    Writer w -> {
      w.name
    }
  }
}
```

![Diagram showing label and caption customizations](image)
Menu customizations

menuBuilder {  
  val factory = EXTLibraryFactory.getInstance

  emfMenus {  
    Writer w -> [#
      actionChange("New book", w.eContainer as Library, 
      library |  
      val book = factory.createBook
      library.books += book
      book.title = "A new book"
      book.author = w
    )
  }
}

Library My Library
- Lorenzo Bettini
- "A story"
- "another story"
- "untitled"
- "untitled"
- "A new book"

"A new book"
Written by: Lorenzo Bettini
Title: A new book
Category: Mystery

Menu 1
- Default
- Content
- Features
- Label
- Menu 2

Menu 2
Menu customizations

Book b -> #[
    actionChange("New writer", b.eContainer as Library,
    [
        library |
        val writer = factory.createWriter
        library.writers += writer
        writer.name = "A new writer"
        writer.books += b
    ]
),
    actionAdd("Make a copy",
    (b.eContainer as Library).books,
    factory.createBook,
    [title = b.title
     author = b.author
    ]
    )
]
Demo from scratch

- Create a new Project with EMF Parsley wizard
- Use a “On Selection Tree Form” view
- Customize to visualize an Ecore
- Show interactions with Ecore editor
Fully EMF compliance

- EMF Resource Compliant
- Editing Domain
- Notification system, Undo/Redo, DnD, etc..
- EMF standard persistence (CDO, Teneo, XMI)
Big App friendly

EMF Parsley helps you to organize big applications

• All customizations related to a single view are in the same plugin

• The DSL collects all aspects configurations / customizations in a compact form

• You can use Ctrl-Shift-F3 to search across projects
Easy upgrading

When a new version is released, a simple project clean will upgrade your code

• Purely Java runtime APIs

• No migrations needed (no model underneath)

• If runtime is updated the DSL will do the work for you
EMF Parsley is ready to be used in production!

- Graduation release since Neon
- It’s used in deployed applications with both RCP and RAP implementations
- You can switch between persistence implementations without much effort
Code Quality

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines Of Code</td>
<td>11,439</td>
</tr>
<tr>
<td>Files</td>
<td>175</td>
</tr>
<tr>
<td>Functions</td>
<td>1,030</td>
</tr>
<tr>
<td>Technical Debt</td>
<td>0</td>
</tr>
<tr>
<td>Directory Tangle Index</td>
<td>0.00%</td>
</tr>
<tr>
<td>Dependencies To Cut</td>
<td>0</td>
</tr>
<tr>
<td>Unit Tests Coverage</td>
<td>93.3%</td>
</tr>
<tr>
<td>SOA Rating</td>
<td>A</td>
</tr>
<tr>
<td>Technical Debt Ratio</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Java</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Directories</td>
<td>47</td>
</tr>
<tr>
<td>Lines</td>
<td>18,310</td>
</tr>
<tr>
<td>Classes</td>
<td>189</td>
</tr>
<tr>
<td>Comments</td>
<td>11.0%</td>
</tr>
<tr>
<td>Comment Lines</td>
<td>1,414</td>
</tr>
<tr>
<td>Duplications</td>
<td>0.0%</td>
</tr>
<tr>
<td>Complexity</td>
<td>1.8/fun</td>
</tr>
<tr>
<td>9.9/class</td>
<td>10.7/FL</td>
</tr>
<tr>
<td>Total</td>
<td>1,866</td>
</tr>
</tbody>
</table>

@EmfParsley

www.eclipse.org/emf-parsley
Prototypes

AngularJS

GWT

Hybrid (Thym)

Android (Andmore)
EMF Parsley resources

Homepage
  • www.eclipse.org/emf-parsley

Documentation
  • https://www.eclipse.org/emf-parsley/documentation.html

Forum

Bugzilla
  • https://bugs.eclipse.org/bugs/buglist.cgi?product=EMF.Parsley