Five good reasons to spice up your projects with EMF Parsley
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
Five 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)
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()
    .Descriptor.Registry.INSTANCE);
adapterFactory.addAdapterFactory(new ResourceItemProviderAdapterFactory());
adapterFactory.addAdapterFactory(new MyModelItemProviderAdapterFactory());
adapterFactory.addAdapterFactory(new ReflectiveItemProviderAdapterFactory());
BasicCommandStack commandStack = new BasicCommandStack();
commandStack.addCommandStackListener(new CommandStackListener()
            {
                ...
            });
editingDomain = newAdapterFactoryEditingDomain(
    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);
```
@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
• Access to all Java types

• According to project dependencies/classpath

• DSL elements can extends your Java classes

• DSL fully debuggable!
Imports management

- Automatic import
- Quick fixes

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

- Couldn't resolve reference to JvmConstructor 'SimpleDateFormat'.
- 2 quick fixes available:
  - Import 'SimpleDateFormat' (java.text)
  - Create Java class 'SimpleDateFormat'
JDT Integration

```java
labelProvider {
  text {
    Book b -> {
      """ + b.title + """
    }
    Writer w -> {
      w.name
    }
  }
}
menuBuilder {
  val factory = EXTLibrary
  emfMenus {
    Writer w -> [
      onChange("New",
      library
      val book
      library
      books = Book
    ]
  }
}
```

**String Book.getTitle()**

Returns the value of the 'Title' attribute.

Returns:

the value of the 'Title' attribute.

See Also:

- setTitle(String)
- org.eclipse.emf.examples.extlibrary.EXTLibraryPackage.getBook_Title()

```java
@model
@generated
```

---

`org.eclipse.emf.common.util.URI`

A representation of a Uniform Resource Identifier (URI), as specified by **RFC**

- URI - java.net
- URI - org.eclipse.emf.common.util
- URIDereferencer - javax.xml.crypto

---

@EmfParsely

[EMF Parsley](http://www.eclipse.org/emf/parsley)
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

```java
browser | browser.layoutData = GridDataFactory.fillDefaults().grab()
browser.^text = body
```
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

```java
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

```
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

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

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

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

- "A story"
  - Written by: Lorenzo Bettini
  - Title: A story
  - Category: Mystery
Menu customizations

```scala
menuBuilder {
  val factory = EXTLibraryFactory.eINSTANCE

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

Undo Do Command Ctrl+Z
Redo Ctrl+Y
Cut
Copy
Paste
Delete
Validate
Control...

New book

Default Content Features Label Menu 1 Menu 2
Menu customizations

```scala
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

@EmfParsely  www.eclipse.org/emf-parsley
Fully EMF compliance

- EMF Resource Compliant
- Editing Domain
- Notification system, Undo/Redo, DnD, etc..
- EMF standard persistence (CDO, Teneo, XMI)
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 in Neon
- It’s used in deployed applications with both RCP and RAP implementations
- You can switch between persistence implementations without much effort
## Code Quality

### Metrics

<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>Java</td>
<td>47 directories, 18,310 lines</td>
</tr>
<tr>
<td>Files</td>
<td>175 directories, 4,523 files</td>
</tr>
<tr>
<td>Functions</td>
<td>1,030 classes, 4,523 statements, 49 accessors</td>
</tr>
<tr>
<td>Documentation</td>
<td>44.8%</td>
</tr>
<tr>
<td>Public API</td>
<td>578 lines</td>
</tr>
<tr>
<td>Public Undoc. API</td>
<td>319 lines</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>Lines</td>
<td>0</td>
</tr>
<tr>
<td>Blocks</td>
<td>0</td>
</tr>
<tr>
<td>Files</td>
<td>0</td>
</tr>
<tr>
<td>Complexity</td>
<td>1.8/function</td>
</tr>
<tr>
<td>9.9/class</td>
<td></td>
</tr>
<tr>
<td>10.7/file</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1,866</td>
</tr>
<tr>
<td>Technical Debt</td>
<td>0</td>
</tr>
<tr>
<td>Issues</td>
<td>0</td>
</tr>
<tr>
<td>Directory Tangle Index</td>
<td>0.0%</td>
</tr>
<tr>
<td>Cycles</td>
<td>0</td>
</tr>
<tr>
<td>Dependencies To Cut</td>
<td>0 between directories, 0 between files</td>
</tr>
<tr>
<td>Unit Tests Coverage</td>
<td>93.3%</td>
</tr>
<tr>
<td>Line Coverage</td>
<td>95.1%</td>
</tr>
<tr>
<td>Condition Coverage</td>
<td>85.4%</td>
</tr>
<tr>
<td>SQALE Rating</td>
<td>A</td>
</tr>
<tr>
<td>Technical Debt Ratio</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

### Technical Debt Pyramid

- Reusability: 0
- Portability: 0
- Maintainability: 0
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