From EMF to UIs: how to use EMF Parsley to get desktop, web and mobile UIs from the model

Vincenzo Caselli
Lorenzo Bettini
About us

Lorenzo Bettini
Dip. Informatica, Univ. Torino, Italy
bettini@dsi.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
Talk Goals

• A practical use case of Parsley usage: how to create and customize a Gmail webapp client with RAP and Parsley in minutes
  ✓ RAP (Remote Application Platform) allows porting RCP applications on the web

• New Parsley feature for other non-desktop environments (web and mobile)
The goal was to provide an easy way to get a User Interface from an EMF model.

EMF Parsley provides:

• simple UI components that can be used in existing applications
• easy-to-use customization mechanisms via Dependency Injection
• DSL to customize UI aspects
Parsley Components

Basic components can be used out of the box:

- Tree
- Table
- Form
- Tree Form
- Table Form
- Dialog
Reference implementations

- Views can also be considered as reference implementation of Parsley components usage.
  - Resource based
  - Selection based

- Project Wizards are provided with initial templates
How it works

• Parsley UI is built on top of EMF.Edit reflective framework,
  • so the default behavior is delegated to EMF Edit

• Customizations are based on Dependency Injection:
  • no need to customize parts and composites
  • customize single behaviors
  • inject them in the framework
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
import org.eclipse.emf.parsley.examples.mail.Account
import org.eclipse.emf.parsley.examples.mail.Folder
import org.eclipse.emf.parsley.examples.mail.accountsview.views.AccountsView

/** org.eclipse.emf.parsley.examples.mail.accountsview Emf Parsley Dsl Module file */
module org.eclipse.emf.parsley.examples.mail.accountsview {

    parts {
        viewpart org.eclipse.emf.parsley.examples.mail.accountsview.views.AccountsView {
            viewname "Mail Accounts View"
            viewclass AccountsView
            viewcategory org.eclipse.emf.parsley.examples.mail.accountsview
        }
    }

    labelProvider {
        image {
            Account -> "account.gif"
            Folder -> {
                switch (name) {
                    case "Inbox" : "inbox.gif"
                    case "Sent" : "sent.gif"
                    case "Trash" : "trash.gif"
                    default: "folder.gif"
                }
            }
        }
    }

    text {
        Account -> email
        Folder -> name
    }

    viewerContentProvider {
        children {
            Folder -> subfolders // don't show emails
        }
    }
}
```java
module org.eclipse.emf.parsley.examples.mail.messageview {

    parts {
        viewpart org.eclipse.emf.parsley.examples.mail.messageview.views.MessageView
        viewname "Mail Message View"
        viewclass MessageView
        viewcategory org.eclipse.emf.parsley.examples.mail.messageview
    }

    labelProvider {
        text {
            Mail -> subject
        }
        Image {
            Mail -> "email.png"
        }
    }

    formControlFactory {
        control {
            Mail : message ->
            val t = createText("",
                SWT.MULTI, SWT.BORDER,
                SWT.WRAP, SWT.V_SCROLL)
            t.editable = false
            t
            target observeText(SWT.Modify)
        }
        featuresProvider {
            features {
                // the subject is already in the title
                Mail -> from, recipients, message
            }
        }
        featureCaptionProvider {
            text {
                Mail : recipients -> 'to'
            }
        }
    }
}
```
A simple Gmail webapp client with EMF Parsley and RAP
Beyond RCP Concepts

• Ok, Parsley can be used in Web Applications with RAP (it is already in production)

• We wanted to go beyond the OSGi/RCP/RAP concepts

• Would it be possible to run Parsley on a pure Java Enterprise Environment (JEE)?
Chopping Parsley

• Separated Parsley Core from (SWT) UI

• Surrounded Parsley Core with a JEE layer

• RCP Plugins dependencies are used as plain JARs in a pure Java environment

• Thanks to usage of Google Guice Injection Parsley Core keeps being fully operative
Dropping Parsley

• The JEE layer (servlets) access the Parsley Core and explores EMF Resources according with Parsley customizations

• JEE layer exposes Parsley via JSON APIs

• Also widgets customizations are available thanks to a lightweight headless porting of the SWT layer
Parsley JEE Wizard

Use classic Dynamic Web Project wizard
... but with the new Parsley Facet
Parsley JEE Wizard

Two facets available:

- JSON API Parsley Server
- UI AngularJS
Persistence choice

EMF Parsley Web project options
Configure the EMF Parsley Web project options.

Persistence options
- None
- Teneo
- CDO
The generated JEE project

- The DSL
- Parsley dependencies
- JEE dependencies
- The AngularJS UI
Parsley flavours

• JSON Parsley APIs allows building UIs with any technology

• Some initial implementations are available:
  ✓ AngularJS and GWT for the web
  ✓ Eclipse Andmore for mobile Android
  ✓ Eclipse Thym for hybrid mobile
Parsley on AngularJS
Keeping DSL & Parts customizations

EMF URIs

```java
package it.rcpvision.parsley.web.angularjs.parts;

import org.eclipse.emf.parsley.web.servlets.WebViewPart;

public class UsersView extends WebViewPart {

    parts {
        viewport UsersList {
            viewname "User View Name"
            viewclass UsersView
        }

        viewport GroupsList {
            viewname "Group View Name"
            viewclass GroupsView
        }
    }

    configurator {
        resourceURI UsersView -> {
            URI.createURI("hibernate://dsn=MyDb&query1=from User u WHERE u.active=true")
        }

        GroupsView -> {
            URI.createURI("hibernate://dsn=MyDb&query1=from Group")
        }
    }

    eClass {
        UsersView -> ProductsPackage.Literals.USER
        GroupsView -> ProductsPackage.Literals.GROUP
    }
}
```
Keeping SWT concepts in DSL

A thin and headless SWT layer allows to use SWT APIs in the DSL UI customization via DSL.
Keeping SWT concepts in DSL (cont’d: as in desktop/RCP, BTW)

Note that the DSL customization works in the same way for web and desktop implementation.
<table>
<thead>
<tr>
<th>timeSlot</th>
<th>room</th>
<th>title</th>
<th>speakers</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00 - 12:00</td>
<td>Theater Stage</td>
<td>What every Eclipse developer should know about Eclipse 4 <em>(e4)</em></td>
<td>Jonas Helming [EclipseSource Munich], Philip Langer [EclipseSource Services]</td>
</tr>
<tr>
<td>09:00 - 12:00</td>
<td>Bürgersaal 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>09:00 - 12:00</td>
<td>Sichersaal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>09:00 - 12:00</td>
<td>Schubartsaal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>09:00 - 12:00</td>
<td>Seminarraum 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>09:00 - 12:00</td>
<td>Seminarräume 1-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>09:00 - 12:00</td>
<td>Seminarräume 3-4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>09:00 - 12:00</td>
<td>FMZ Präsentationsraum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:00 - 12:30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:30 - 13:30</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Parsley on Eclipse Thym

- www.eclipse.org/thym
- based on Apache Cordova
- allows cross-mobile dev
  - ✔ Android
  - ✔ iOS
  - ✔ Windows Phone
Parsley on Eclipse Andmore

- projects.eclipse.org/projects/tools.andmore
- fork of ADT
EMF Parsley resources

Homepage
- www.eclipse.org/emf-parsley

Documentation

Forum

Bugzilla
- https://bugs.eclipse.org/bugs/buglist.cgi?product=EMF.Parsley
Evaluate the sessions at www.eclipsecon.org