



EMF
Parsley

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

EclipseCon Europe 2015
Ludwigsburg - Nov 4, 2015

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)

Intro a EMF Parsley

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

A dialog box titled "Book Lorenzo's Book" with the following fields: publicationDate (text), copies (text, value 0), borrowers (text with a menu icon), title (text, value "Lorenzo's Book"), pages (text, value "100"), category (dropdown, value "Mystery"), and author (dropdown, value "Writer Lorenzo Bettini"). It has OK and Cancel buttons at the bottom.

A tree view showing a library structure: Library My Library > Writer Lorenzo Bettini > Borrower: Foo > Book: Lorenzo's Book > Book: Another Book > Video Cassette My Video > Video Cassette My Damaged Video. Below it is a form for "Book: Another Book" with fields: publicationDate, copies (0), borrowers (with menu icon), title (Another Book), pages (100), category (Mystery), and author (dropdown with "Writer Fake Writer" selected).

A tree view showing email folders for three users: lorenzo@foobar (Inbox, Sent, Trash, myfolder, myfolder2), vincenzo@foobar (Inbox, Sent, Trash, myfolder, myfolder2), and francesco@foobar (myfolder2).

subject	from
Test subject 1	foo@foobar
Test subject 2	foo@foobar

An email message form titled "Test subject 1" with fields: from (foo@foobar), to (dest@foobar), and message (This is a test message. Number 1. Cheers!).

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
    }
  }
}
```

Package Explorer

- org.eclipse.emf.parsley.examples.mail.messageview
 - src
 - org.eclipse.emf.parsley.examples.mail.messageview
 - MessageviewActivator.java
 - MessageviewExecutableExtensionFactory.java
 - MessageviewGuiceModule.java
 - module.parsley
 - org.eclipse.emf.parsley.examples.mail.messageview.views
 - MessageView.java
 - emfparsley-gen
 - org.eclipse.emf.parsley.examples.mail.messageview
 - EmfParsleyGuiceModuleGen.java
 - EmfParsleyGuiceModuleGen
 - EmfParsleyGuiceModuleGen(AbstractUIPlugin)
 - bindFeatureCaptionProvider(): Class<? extends FeatureCaptionProvider>
 - bindFeaturesProvider(): Class<? extends FeaturesProvider>
 - bindFormControlFactory(): Class<? extends FormControlFactory>
 - bindLabelProvider(): Class<? extends ILabelProvider>
 - org.eclipse.emf.parsley.examples.mail.messageview.binding
 - FormControlFactoryGen.java
 - org.eclipse.emf.parsley.examples.mail.messageview.ui.provider
 - FeatureCaptionProviderGen.java
 - FeatureCaptionProviderGen
 - text_Mail_recipients(EStructuralFeature): String
 - FeaturesProviderGen.java
 - LabelProviderGen.java
 - LabelProviderGen
 - LabelProviderGen(AdapterFactoryLabelProvider)
 - image(Mail): Object
 - text(Mail): String
 - JRE System Library [J2SE-1.5]
 - Plug-in Dependencies
 - icons
 - META-INF
 - build.properties
 - plugin.xml
 - plugin.xml_emfparsley_gen

Generated Code

```
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'  
        }  
    }  
}
```

Demo

A simple Gmail webapp client with EMF Parsley and RAP

The screenshot shows a web application window titled "Mail App". On the left is a sidebar with a list of folders: "INBOX", "Work", and "Home". The "Work" folder is selected and highlighted with a red box. Three red arrows point from the "Work" folder to the first three rows of a table. The table has four columns: "Date", "From", "Subject", and "Snippet". The first row is highlighted with a red box and contains the following data: "23/09/2015", "John", "Hi", and "Just wanna say hello". A red arrow points from the "Hi" subject to a detail view below. The detail view has a light green background and contains three input fields: "Date" (with "23/09/2015" entered), "From", and "Subject".

Date	From	Subject	Snippet
23/09/2015	John	Hi	Just wanna say hello
25/09/2015	Paul	Meeting staff	Monday 9.30 meeting staff
29/09/2015	Helen	App V 1.0.4 deployed	Version v. 1.0.4

Date:
From:
Subject:

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

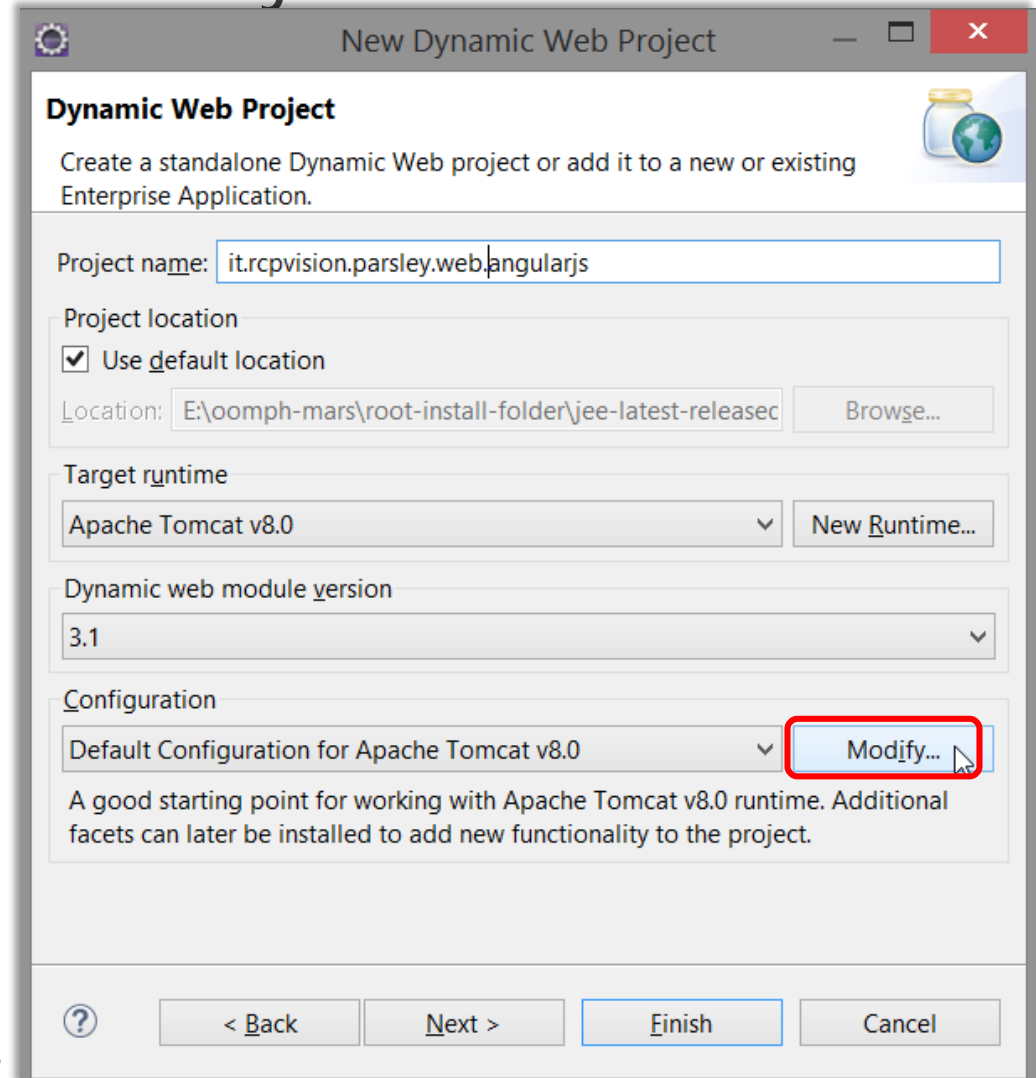
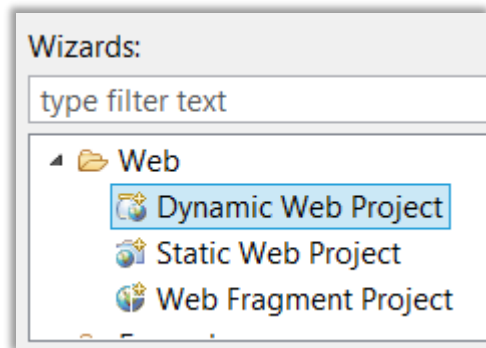
- Separated Parsley Core from (SWT) UI
- Surrounded Parsley Core with a JEE layer
- RCP Plugins dependencies are used as plain JARs in a pure Java environment
- Thanks to usage of Google Guice Injection Parsley Core keeps being fully operative

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

Project Facets

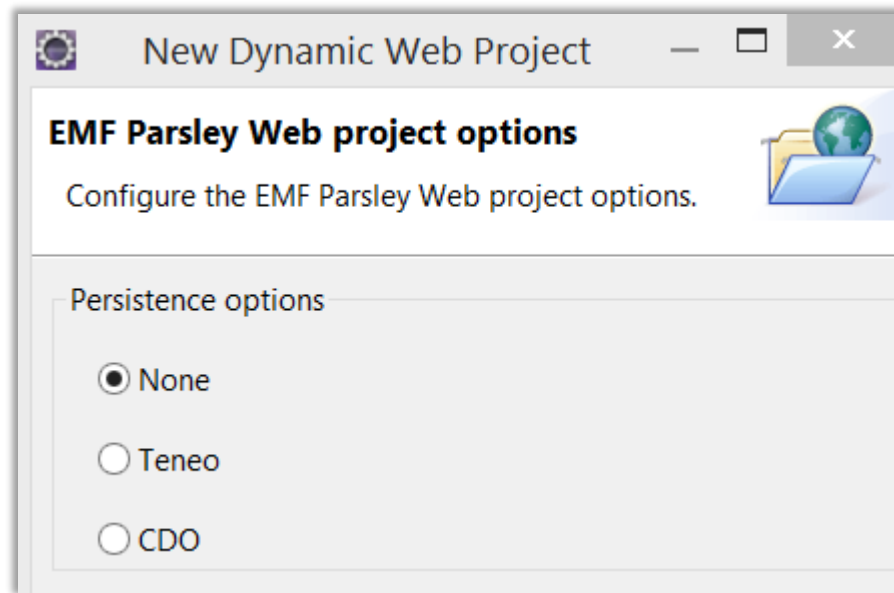
Select the facets that should be enabled for this project.

Configuration: <custom>

Project Facet	Version
<input type="checkbox"/> Axis2 Web Services	
<input type="checkbox"/> CXF 2.x Web Services	1.0
<input checked="" type="checkbox"/> Dynamic Web Module	3.1
<input checked="" type="checkbox"/> EMF Parsley Web	
<input checked="" type="checkbox"/> EMF Parsley Web Server	0.5
<input checked="" type="checkbox"/> UI AngularJS client	0.5
<input checked="" type="checkbox"/> Java	1.7
<input checked="" type="checkbox"/> JavaScript	1.0
<input type="checkbox"/> JavaServer Faces	2.2
<input type="checkbox"/> JAX-RS (REST Web Services)	1.1
<input type="checkbox"/> JAXB	2.2
<input type="checkbox"/> JPA	2.1
<input type="checkbox"/> WebDoclet (XDoclet)	1.2.3

Parsley JEE Wizard

Persistence choice



The generated JEE project

The screenshot displays the Eclipse IDE's Project Explorer for a project named `it.rcpvision.parsley.web.angularjs`. The project structure includes:

- `JAX-WS Web Services`
- `Deployment Descriptor: it.rcpvision.parsley.web.angularjs`
- `Java Resources`
 - `src`
 - `it.rcpvision.parsley.web.angularjs`
 - `ParsleyContextListener.java`
 - `ParsleyGuiceServletContextListener.java`
 - `ParsleyWebGuiceModule.java`
 - `module.parsley` (highlighted with a red box and labeled "The DSL")
 - `emfparsley-gen`
 - `Libraries`
 - `JavaScript Resources`
 - `Deployed Resources`
 - `.settings`
 - `build`
 - `target`
 - `WebContent`
 - `META-INF`
 - `WEB-INF`
 - `lib`
 - `web.xml`
 - `add.png`
 - `details.html`
 - `table.html` (highlighted with a red box and labeled "The AngularJS UI")
 - `.classpath`
 - `.project`
 - `pom.xml`

On the right side, a list of JEE dependencies is shown, with two specific dependencies highlighted by red boxes and labeled "Parsley dependencies":

- `org.eclipse.emf.parsley.common_0.5.0.v20150910-1540.jar`
- `org.eclipse.emf.parsley.runtime.common_0.5.0.v20150910-1540.jar`

Other dependencies listed include various Eclipse core and EMF components.

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

The screenshot shows an IDE window titled "JEE & ... - Debug - it.rcpvision.parsley.web.angularjs/src/it/rcpvision/parsley/web/angularjs/ParsleyWebGuice". The IDE interface includes a menu bar (File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help) and a toolbar with various icons. The "Debug" view shows a "Tomcat v8.0 Server at localhost [Debugging, Synchronized]" and a sub-entry for "it.rcpvision.parsley.web.angularjs [Synchronized]".

Overlaid on the IDE is a browser window showing the application running at "localhost:8081/it.rcpvision.parsley.web.angularjs/table.html?s=UsersList". The browser's address bar and bookmarks are visible. The application page displays a "UsersList" table with a search input field. The table has three columns: "First name", "Last name", and "Group". The data rows are:

First name	Last name	Group
John	Smith	Group DEV
Bob	Martin	Group TEST

Below the table is a blue circular button with a plus sign. A modal form titled "John" is open, showing the details for the selected user:

John

Name:

Surname:

Group:

Keeping DSL & Parts customizations

```
Project Explorer
├── it.rcpvision.parsley.web.angularjs
│   ├── JAX-WS Web Services
│   ├── Deployment Descriptor: it.rcpvision.parsley.web.angularjs
│   └── Java Resources
│       └── src
│           ├── it.rcpvision.parsley.v
│           └── it.rcpvision.parsley.v
│               ├── GroupsView.java
│               └── UsersView.java
└── module.parsley

module.parsley
UsersView.java
1 package it.rcpvision.parsley.web.angularjs.parts;
2
3 import org.eclipse.emf.parsley.web.servlets.WebViewPart;
4
5 public class UsersView extends WebViewPart {
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

parts {
  viewpart UsersList {
    viewname "User View Name"
    viewclass UsersView
  }
  viewpart GroupsList {
    viewname "Group View Name"
    viewclass GroupsView
  }
}

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

eClass {
  UsersView -> ProductsPackage.Literals.USER
  GroupsView -> ProductsPackage.Literals.GROUP
}
}
```

EMF URIs

Keeping SWT concepts in DSL

A thin and headless SWT layer allows to use SWT APIs in the DSL
UI customization via DSL

The diagram illustrates the transformation of a text input field into a date picker widget through a DSL. It consists of three main parts: a source UI, a DSL code snippet, and a target UI.

Source UI (Left): A web browser window showing a form for a user named John. The form includes fields for Name (John), Surname (Smith), Group (Group DEV), and birthDate (an empty text input field). A "Save" button is at the bottom.

DSL Code (Middle): A code snippet for `dialogControlFactory` defining a `control` for the `birthDate` field. The code is as follows:

```
dialogControlFactory {
  control {
    User : birthDate -> {
      createDateTime(SWT.DROP_DOWN);
    }
    target observeSelection
  }
}
```

The `createDateTime(SWT.DROP_DOWN);` line is highlighted with a green box. A green arrow points from this line to the date picker widget in the target UI. Another green arrow points from the `birthDate` field in the source UI to the date picker widget in the target UI, with the word "Text" written above the arrow and "Date" written below it.

Target UI (Right): The same web browser window, but the `birthDate` field is now a date picker widget. It displays "mm/dd/yyyy" and a calendar for November 2015. The calendar shows the following dates:

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

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

birthDate

birthDate mm/dd/yyyy

Save

November 2015

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

Birth Date

Birth Date 11/ 3/2015

November 2015

Sun	Mon	Tue	Wed	Thu	Fri	Sat
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

Today: 11/3/2015

Parsley on GWT

timeSlot	room	title	speakers
09:00 - 12:00	Theater Stage	What every Eclipse developer should know about Eclipse 4 (e4)	Jonas Helming [EclipseSource Munich], Philip Langer [EclipseSource Services]
09:00 - 12:00	Bürgersaal 2		
09:00 - 12:00	Silchersaal		
09:00 - 12:00	Schubartsaal		
09:00 - 12:00	Seminarraum 5		
09:00 - 12:00	Seminarräume 1-2		
09:00 - 12:00	Seminarräume 3-4		
09:00 - 12:00	FMZ Präsentationsraum		
12:00 - 12:30			
12:30 - 13:30			

127.0.0.1:8888/ParsleyGwt.html

File Open

Library

- Library
 - Book Modern Art
 - Book Business Models
 - Book Computer science origins
- Author Bruce Harper
 - 357, Portland Road - [Ottawa]
 - 75, Franz Str. - [Berlin]
- Author Joe Sanders
- Author Mark Allen

Book Modern Art

TITLE: Modern Art

authors: Author Bruce Harper

pages: 0

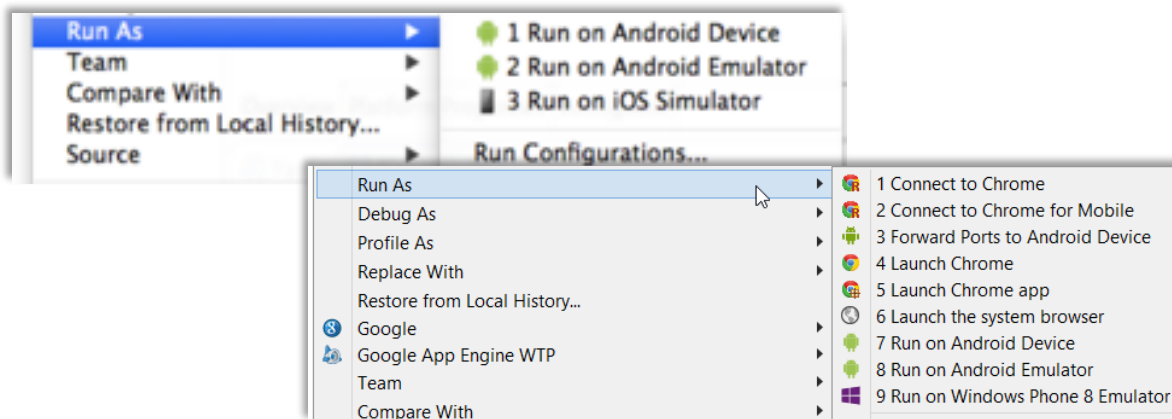
reviewer: Author Joe Sanders

- Author Bruce Harper
- Author Joe Sanders
- Author Mark Allen



Parsley on Eclipse Thym

- www.eclipse.org/thym
- based on Apache Cordova
- allows cross-mobile dev
 - ✓ Android
 - ✓ iOS
 - ✓ Windows Phone

A screenshot of a mobile application interface. At the top, there is a search bar and a status bar showing the time as 19:25 and 72% battery. Below the search bar is a table with three columns: 'timeSlot', 'room', and 'title'. The table contains several rows of event information.

timeSlot	room	title
09:00 - 12:00	Theater Stage	What every Eclipse developer should know about Eclipse 4 (e4)
09:00 - 12:00	Bürgersaal 2	Docker Beginners Tutorial
09:00 - 12:00	Silchersaal	Oomph: Eclipse the Way You Want It
09:00 - 12:00	Schubartsaal	OSGi enRoute, Building OSGi Apps, Release 1.0!
09:00 - 12:00	Seminarraum 5	Shoot-A-Pi with Eclipse Kura
09:00 -	Seminarräume 1-2	eXXXtreme Tutorial - Xtext,

Parsley on Eclipse Andmore

- projects.eclipse.org/projects/tools.andmore
- fork of ADT

The screenshot shows the Google Play Store interface. At the top, there's a search bar with the text "Cerca" and a magnifying glass icon. Below the search bar, there are navigation options: "App", "Categorie", "Home page", "Le migliori app", and "Nuove uscite". On the left side, there's a sidebar menu with options: "Le mie app", "Acquista", "Giochi", "Famiglia", "Da non perdere", "Attività su Play", "La mia lista desideri", "Codice promozionale", "Acquista carta regalo", and "Guida per i genitori". The main content area features a large promotional banner for "EclipseCon Europe 2015". The banner includes the text "eclipsecon Europe 2015", "Ludwigsburg, Germany 3 - 5 November", and the OSGi logo. To the right of the banner, there's a section for "EclipseCon Europe 2015" with the text "Vincenzo Caselli [RCP Vision] Notizie e riviste", "PEGI 3", and "L'app è compatibile con tutti i tuoi dispositivi." Below the banner, there are two smaller screenshots of the app's interface, showing a list of events with details like time, location, and speaker.

The screenshot shows the EclipseCon Europe 2015 app interface. At the top, there's a status bar with the time "02:26" and various system icons. Below the status bar, there's a header "Dedicated Exhibit Time". The main content area is a list of events. Each event entry includes a time slot, a location, and a title. The events listed are:

- 12:15 - 13:45 **Lunch**
- 13:45 - 14:20 **Theater**
The Future of Xtext
Sebastian Zarnekow [itemis], Stefan Oehme [itemis]
- 13:45 - 14:20 **Theater Stage**
High productivity development with Eclipse and Java 8
Noopur Gupta [IBM India]
- 13:45 - 14:20 **Bürgersaal 2**
Doclipse or how I've put Docker in your favorite IDE
Mario Loriedo [Zenika]
- 13:45 - 14:20 **Silchersaal**
From EMF to UIs: how to use EMF Parsley to get desktop, web and mobile UIs from the model
Vincenzo Caselli [RCP Vision], Lorenzo Bettini, Francesco Guidieri
- 13:45 - 14:20 **Schubartsaal**
Modules all the way down: OSGi and the Java

EMF Parsley resources

Homepage

- www.eclipse.org/emf-parsley

Documentation

- <https://www.eclipse.org/emf-parsley/documentation.html>

Forum

- <https://www.eclipse.org/forums/index.php/f/263/>

Bugzilla

- <https://bugs.eclipse.org/bugs/buglist.cgi?product=EMF.Parsley>



eclipsecon Europe

Ludwigsburg, Germany, 3 - 5 November 2015

Evaluate the sessions at www.eclipsecon.org

+1

0

-1