



**eclipsecon**  
France 2017

Lesson learned from using EMF to  
build  
Desktop & Web Applications

Toulouse 22 June 2017

# 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

# Introduction

RCP Vision is an Italian company based on Eclipse Technologies since 2009.

Our specialization is Eclipse RCP and EMF based applications

- ▶ Evangelization on EMF based technologies
- ▶ Consultant on Graphical and Textual DSL
- ▶ Integration and problem solving with CDO repository
- ▶ EMF Parsley for speed up UI development

In the past years we worked a lot on RAP and Parsley integration.

We do a lot of Architectural work with our customers.

# Our Challenge

We faced recently a big challenge by migrating our biggest application from a Desktop (CDO based) client-server to the WEB.

- Several views
- PDF and Image management
- Document acquisition engine based on EMF
- Several type of acquisition and document format

... and we've been surprised by how we had done it easily in a couple of week.

# So far so good..

So we spent some time asking ourself why we got it so easily..

The answer is probably a mix of good patterns and technologies that we've learned after some years of experience with Eclipse:

- ▶ EMF runtime and edit
- ▶ Reflective UI using EMF Parsley
- ▶ Persistence based on Resource APIs
- ▶ Use of Inversion of Control (Guice and OSGI services)

A red arrow pointing to the right, located at the top left of the slide.

# Our goal

Build up an EMF-based architecture that is independent from Persistence layer for Web and Desktop applications.

RAP framework resolves the problem of UI single sourcing.

Persistence implementation is quite easy with EMF.

We have a lot of frameworks... but what about a legacy DB?

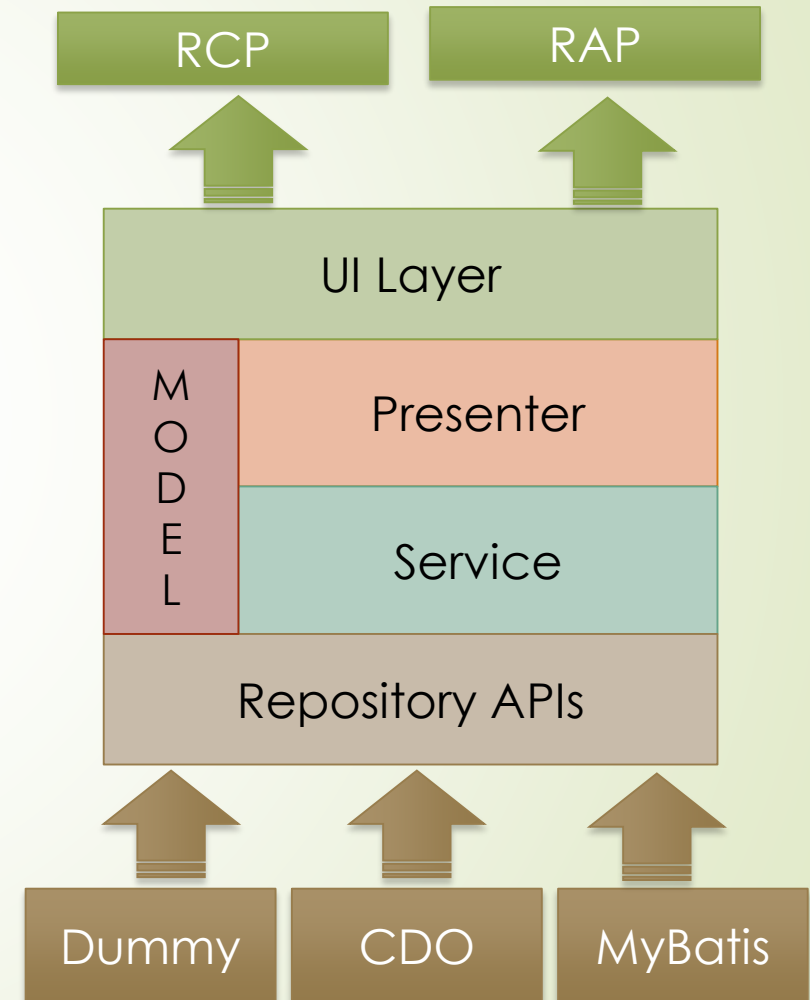
[www.eclipse.org/rap](http://www.eclipse.org/rap)

# About this talk

- ▶ Some theoretical slides on basic patterns and how we (revisited and) applied them to our architecture.
- ▶ Some practical slides with code snippets to highlight most important stuff.
- ▶ A practical example, that is a proof of concept and demonstrates that we can develop one single application with different UI and Persistence implementations.

# Overall Architecture

- ▶ UI layer is 100% compliance RCP / RAP
- ▶ Presenter layer...
- ▶ Service layer uses the Repository API
- ▶ There are 3 different type of repository
- ▶ Persistence is injected as OSGI service

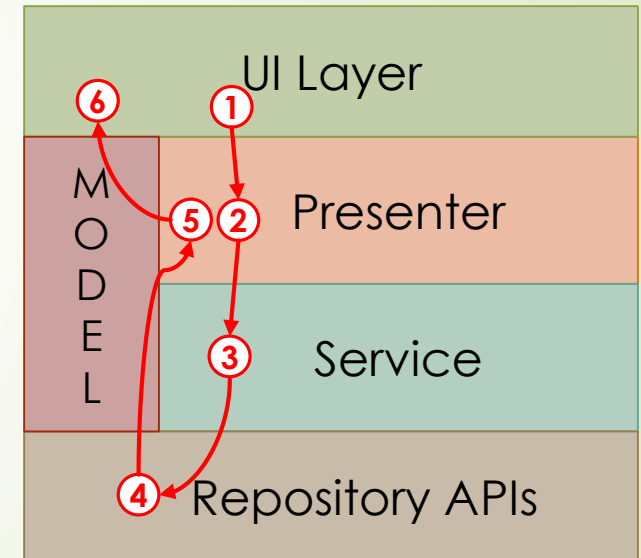




# Architecture simulation

## Querying for objects

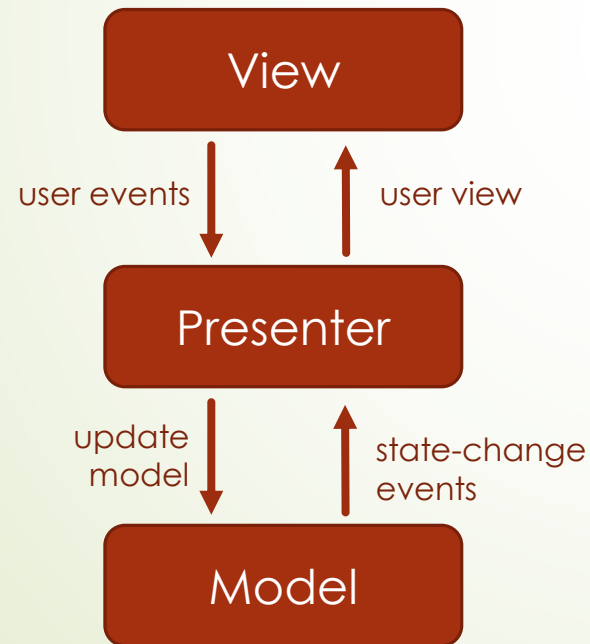
1. The UI layer ask the presenter for objects to be shown
2. The Presenter call the Service layer to get the objects
3. The Service layer implements the business logic and knows which repositories are needed for the job.
4. The repository implementations return all objects that satisfy the query.
5. The presenter prepare the objects for visualization.
6. The UI layer performs the databinding between the objects and the UI controls



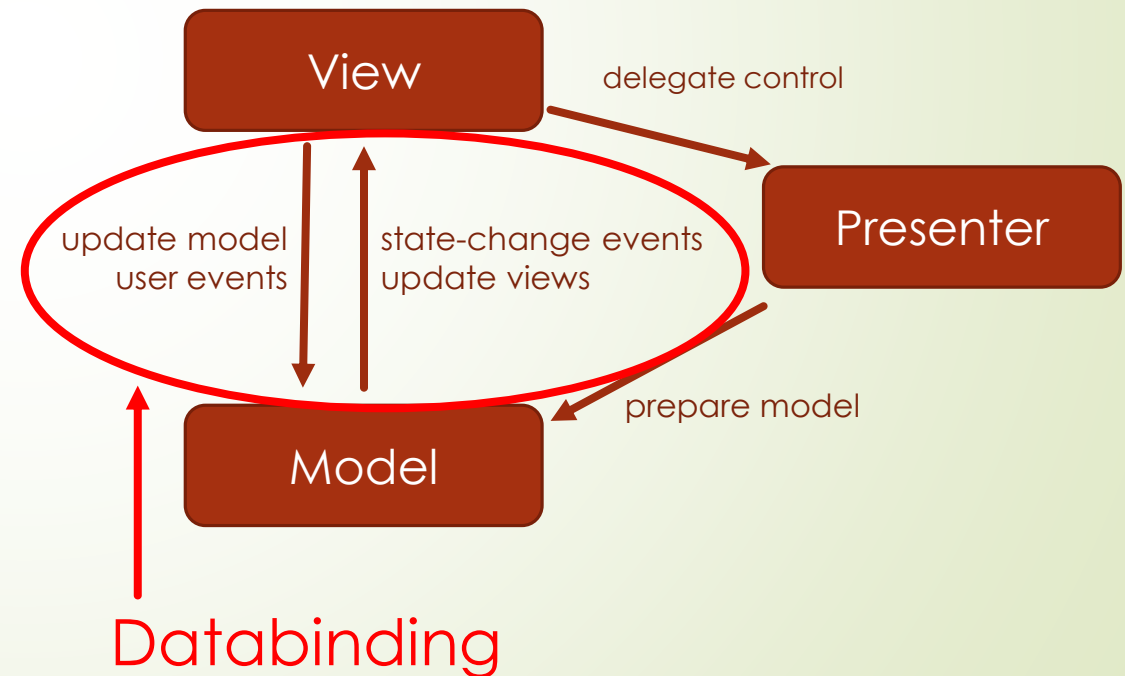
# Pattern: Model View Presenter (revisited)

The UI is strongly coupled with the model and the control logic is completely delegated to a Presenter.

### Original MVP



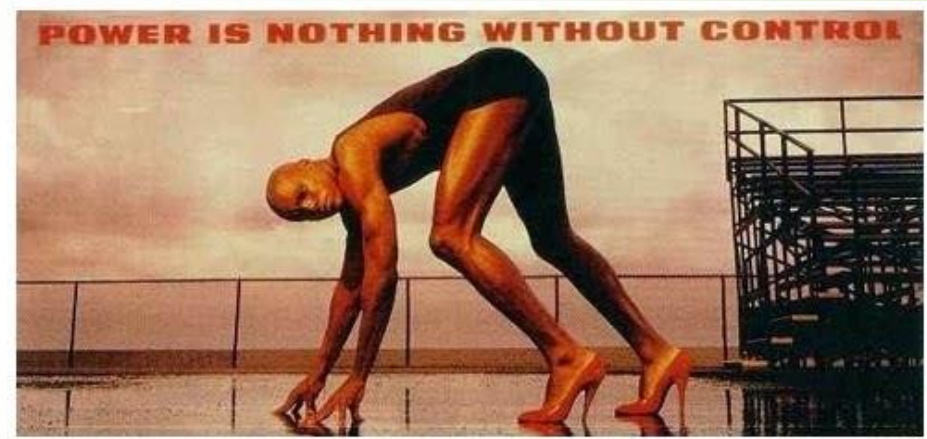
### Revisited MVP



# Full EMF Compliance

EMF is a great framework but you have to use it properly:

- Resource APIs
- Editing Domain
- Adapter factories



...and you'll get (almost) for free

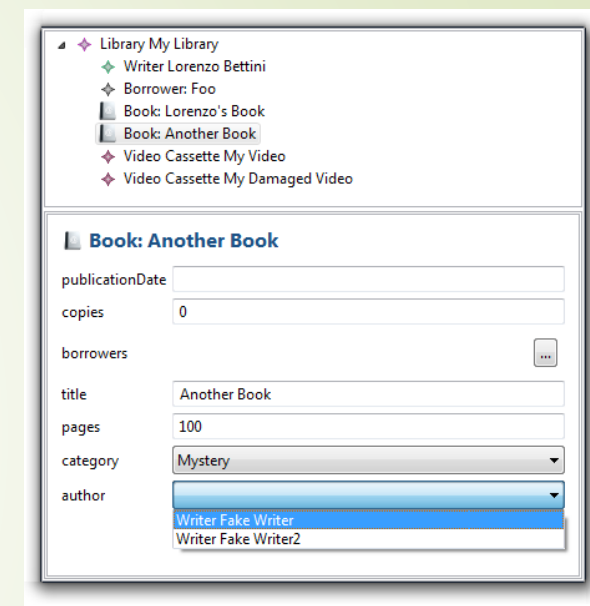
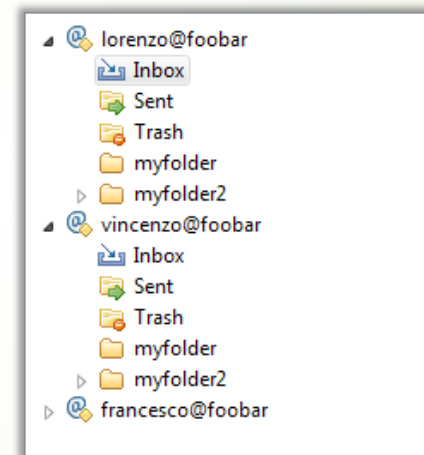
- UI - Model Databinding
- Notificaton system and editing scoping
- more advanced (Validation, DnD support, etc...)

# EMF Parsley

[www.eclipse.org/emf-parsely](http://www.eclipse.org/emf-parsely)

Basic components can be used out of the box:

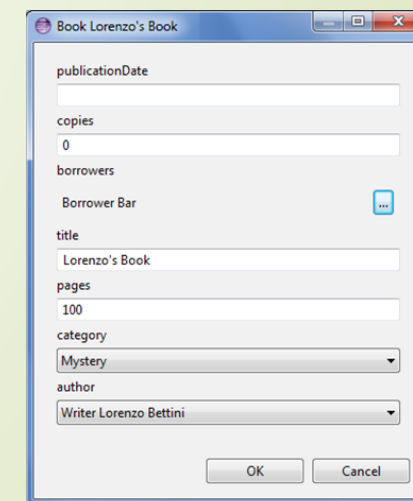
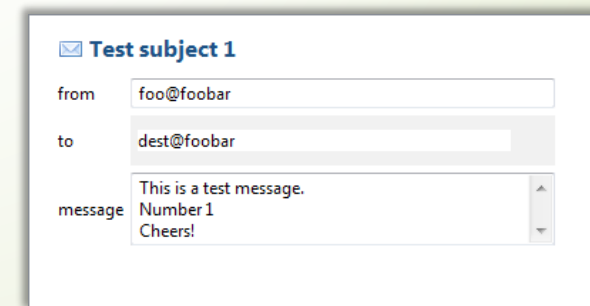
- Tree
- Table
- Form
- Tree Form
- Table Form
- Dialog



unread	date
true	18/11/2015
false	25/09/2015
false	16/09/2015

Mark as Read/Unread

Copy



# EMF Parsley in practice

How we use EMF Parsley to build the Table Form component.

```
public abstract class TableFormView<V extends IViewerPresenter> extends ViewPart {

    @Inject
    private TableFormFactory tableFormFactory; ◀ Parsley Injection

    @Inject
    V presenter;

    @Override
    public void createPartControl(Composite parent) {
        presenter.init();

        tableFormComposite = tableFormFactory ◀ Parsley Table Form creation
            .createTableFormMasterDetailComposite(parent, SWT.BORDER, getEClass());
        tableFormComposite.update(presenter.getViewerInput()); ◀ Viewer input

        Button saveButton = new Button(buttonContainer, SWT.NONE);
        saveButton.setText("Save");
        saveButton.addSelectionListener(
            uiUtil.createSelectionAdapter(presenter::saveButtonPressed));
    }
}
```

# MVP in practice

The View (UI Layer) uses the presenter to delegate all control logic.

```
public abstract class TableFormView<V extends IViewerPresenter> extends ViewPart {

    @Inject
    private TableFormFactory tableFormFactory;

    @Inject
    V presenter; ◀ Presenter Injection

    @Override
    public void createPartControl(Composite parent) {
        presenter.init(); ◀ Explicit Initialization

        tableFormComposite = tableFormFactory
            .createTableFormMasterDetailComposite(parent, SWT.BORDER, getEClass());
        tableFormComposite.update(presenter.getViewerInput()); ◀ Viewer input

        Button saveButton = new Button(buttonContainer, SWT.NONE);
        saveButton.setText("Save");
        saveButton.addSelectionListener(
            uiUtil.createSelectionAdapter(presenter::saveButtonPressed));
    }
    ▶ Lambda action
```

# Inversion of Control

Guice is used to inject implementations in UI, Presenter and Service layers....

It can be used with generics to gain very powerful mechanism for Inversion of Control.

```
public abstract class AbstractViewerPresenter<T> extends EObject,  
    <S> extends IViewerService<T>> implements IViewerPresenter{
```

```
@Inject  
IEditingStrategy editingStrategy;
```

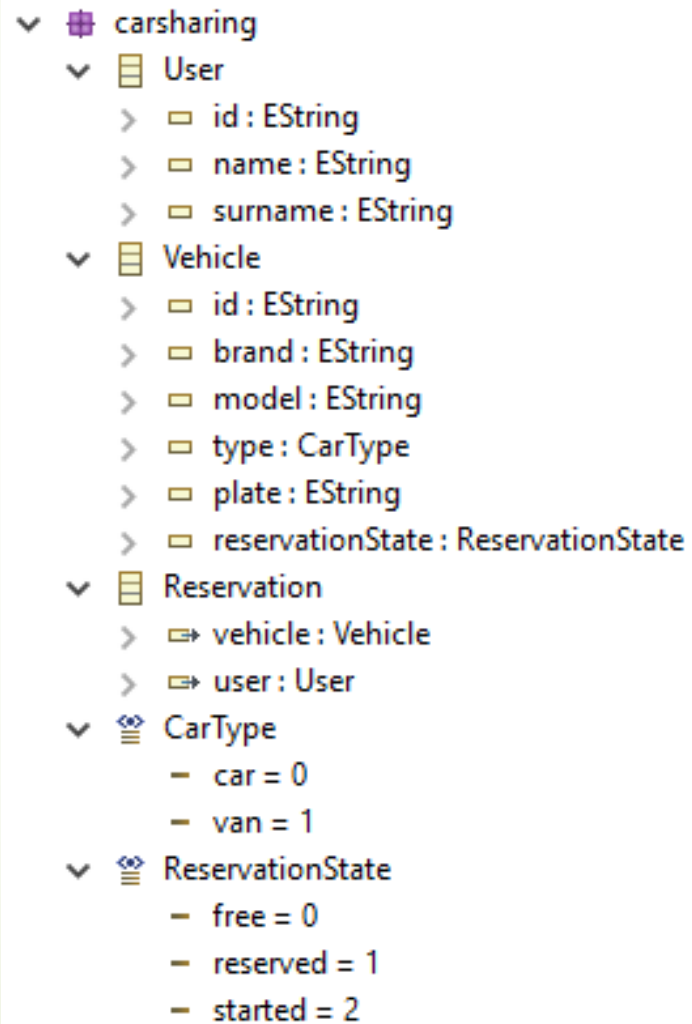
```
@Inject  
protected <S> service;
```

```
@Inject  
private EmfUtil emfUtil;
```

Define the object

Define the service

# Example model



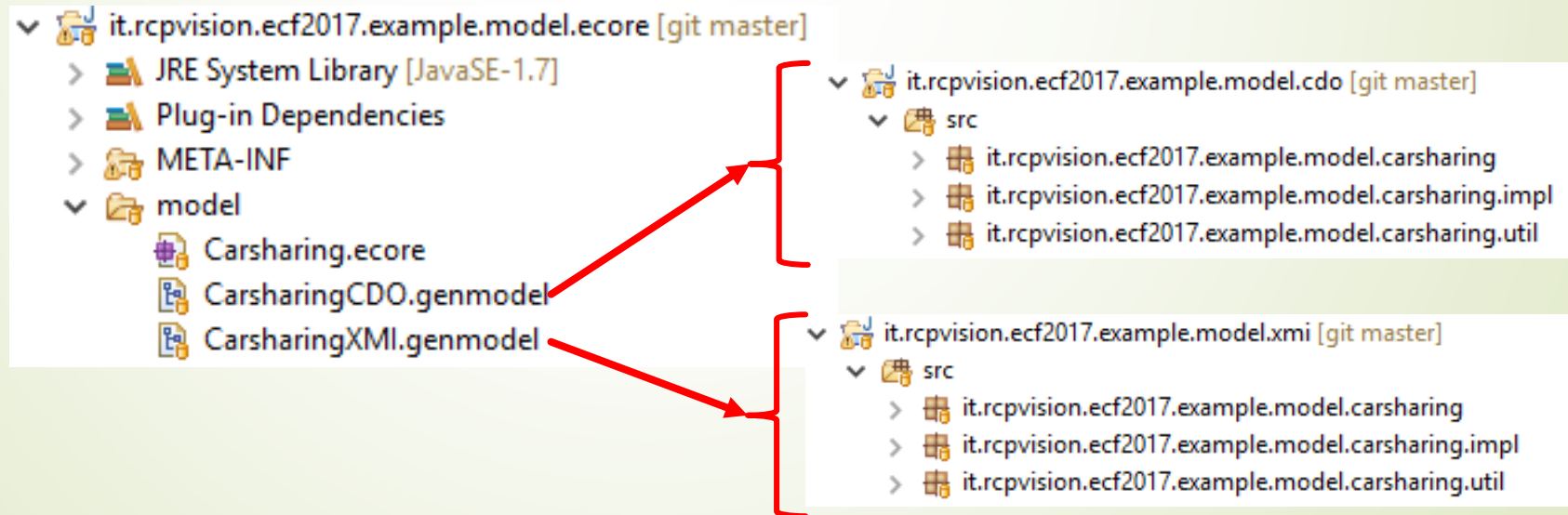


# EMF and CDO generated code















CDO Object is an EObject, but CDOObjectImpl is not an EMFObjectImpl!

EMF and CDO have two different generation how can we manage in a single application?

- Generate EMF and CDO cdo into 2 different plugin
- Using import package instead of required bundle in dependencies (see next slide)



# The Example projects

- >  it.rcpvision.common.cdo.server.feature } CDO Server
- >  it.rcpvision.common.cdo.server.product } CDO Server
- >  it.rcpvision.ecf2017.example.app.rap → RAP Application
- >  it.rcpvision.ecf2017.example.app.rcp → RCP Application
- >  it.rcpvision.ecf2017.example.model.cdo } Model
- >  it.rcpvision.ecf2017.example.model.ecore } Model
- >  it.rcpvision.ecf2017.example.model.xmi } Model
- >  it.rcpvision.ecf2017.example.repository.api } Repository
- >  it.rcpvision.ecf2017.example.repository.cdo } Repository
- >  it.rcpvision.ecf2017.example.repository.dummy } Repository
- >  it.rcpvision.ecf2017.example.repository.mybatis } Repository
- >  it.rcpvision.ecf2017.example.service → Service
- >  it.rcpvision.ecf2017.example.ui → UI
- >  org.eclipse.emf.parsley.examples.rap.targetplatform → RAP target

# RAP / RCP UI plug in

```
Bundle-Name: it.rcpvision.ecf2017.example.ui  
Bundle-Vendor: My Company  
Bundle-Version: 1.0.0.qualifier  
Bundle-SymbolicName: it.rcpvision.ecf2017.example.ui  
Bundle-Activator: it.rcpvision.ecf2017.example.ui.UiActivator  
Bundle-ActivationPolicy: lazy  
Require-Bundle: org.eclipse.core.runtime,  
org.eclipse.ui;resolution=optional,  
org.eclipse.emf.parsley.views;resolution=optional,  
org.eclipse.xtext.xbase.lib,  
org.eclipse.rap.ui;resolution=optional,  
org.eclipse.emf.parsley.rap.views;resolution=optional,  
it.rcpvision.ecf2017.example.service,  
org.eclipse.core.databinding.observable,  
org.eclipse.emf.ecore,  
it.rcpvision.ecf2017.example.repository.api  
Import-Package: it.rcpvision.ecf2017.example.model.carsharing,  
it.rcpvision.ecf2017.example.model.carsharing.impl,  
it.rcpvision.ecf2017.example.model.carsharing.util,  
org.apache.commons.logging,  
org.apache.log4j,  
org.eclipse.emf.cdo;resolution=optional,  
org.eclipse.emf.parsley.cdo;resolution=optional,  
org.eclipse.emf.parsley.cdo.util;resolution=optional
```

In the example UI code is contained in one single project.

The dependencies are defined to support both RAP and RCP target platforms.

And both CDO and EMF have to be supported

# Repository Pattern

*A Repository mediates between the domain and data mapping layers using a collection-like interface for accessing domain objects.*

Objects can be added to and removed from the Repository, as they can from a simple collection of objects, and the mapping code encapsulated by the Repository will carry out the appropriate operations behind the scenes.

<https://martinfowler.com/eaCatalog/repository.html>

An Emf Resource is very similar to a Repository! Don't use DAO, use Repository!

# Resource based Persistence APIs

We use the Repository Pattern to separate the domain model from the persistence layer. The Repository layer uses the Domain model as input.

```
public interface IRepository<T extends EObject> {  
  
    void insert(T obj) throws RepositoryException;  
  
    void update(T obj) throws RepositoryException;  
  
    void delete(T obj) throws RepositoryException;  
  
    T getByKey(Object key);  
  
    List queryAll();  
  
}
```

# Repository implementations

- **DUMMY** implementation fulfills the task to provide the same APIs of other repositories, based on mock-up objects
- **CDO** is one of the implementation of a Model Repository that provides some interesting notification mechanism between different instances of a model. [www.eclipse.org/cdo](http://www.eclipse.org/cdo)
- **MyBatis** is a simple Data Mapper that speeds up the development with a set of facilities and can be very useful with Legacy DB. [www.mybatis.org](http://www.mybatis.org)



# The example sources

Download source code from GitHub

<https://github.com/fraguid/emf-architecture-example.git>

There will be 2 application projects (RCP and RAP) that contain the launch configurations:

## RAP Launches

Rap Application (Dummy).launch  
Rap Application (MyBatis).launch  
Rap Application (CDO).launch

## RCP Launches

Rcp Application (Dummy).launch  
Rcp Application (MyBatis).launch  
Rcp Application (CDO).launch

# The Example IDE setup

We recommend to install Oxygen Modeling EPP, that contains all modeling requirements(EMF, CDO, etc)

For RAP you need to install from Oxygen update site:

RAP Tools (in Web category)

For MyBatis Generator use the update site:

*<https://dl.bintray.com/mybatis/mybatis-generator>*



# The Example target platform setup

You can use the same IDE for RAP and RCP applications, simply switching between the 2 different target platforms:

- ▶ RCP target platform – You can use the running platform of Oxygen with Modeling EPP package.
- ▶ RAP target platform – You can use the Parsley RAP target platform, available as an example, by selecting menu:

*File -> New example -> EMF Parsley ->  
EMF Parsley RAP Target Platform Example*

# The example: Demo time

RCP Application

File

User View Vehicle View

id	brand	model	type	plate	reservationS...
1	Toyota	Yaris	car	AB000CD	free

**Toyota Yaris (AB000CD)**

Brand:

Model:

Type:

Plate:

Insert Save Make a reservation

127.0.0.1:10000/rap

127.0.0.1:10000/rap

User View Vehicle View

id	brand	model	type	plate	reservationState
1	Toyota	Yaris	van	AB000CD	reserved

**Toyota Yaris (AB000CD)**

Brand:

Model:

Type:

Plate:

Insert Save Make a reservation



**eclipse**day

MILANO 2017



**Explore Eclipse**

September 22, 2017

[www.eclipsedaymilano.com](http://www.eclipsedaymilano.com)

A red arrow pointing to the right, located at the top left of the slide.

# Questions?

Thank you for your attention

Remember to evaluate this session

Several thin, curved lines in shades of grey and brown, extending from the left side of the slide towards the center.