

Eclipse Data Binding - Updating RCP Mail 2.0 Handout



Dr. Frank Gerhardt (Gerhardt Informatics), Dr. Boris Bokowski (IBM)

[Eclipse Application Developer Day Karlsruhe, 07.07.2009 \[1\]](#)



All rights reserved. Distributed under [Creative Commons \[2\]](#) Attribution-Noncommercial-Share Alike 3.0 United States License

Abstract

The RCP Mail example shows its age. Many new and useful features have been added to the Eclipse Platform since the example was written, and it is time to take the example to new levels.

In a hands-on way, we will bring the example up to date by adopting some of the more interesting new APIs developed since the original RCP Mail example was written.

In particular, we will show the following:

- How to use the new Commands API to contribute to menus, toolbars, and context menus, and how to create key bindings.
- How to use the Common Navigator to provide a view that shows multiple sets of unrelated content.
- How to use the data binding framework to make the UI code easier to write, and easier to test.

Explanations of the concepts will be given by members of the Eclipse Platform UI team, while the concrete examples will be explained by experienced Eclipse professionals.

We will also show the different tools that are provided by Eclipse PDE and JDT to help you developing RCP applications.

<http://www.eclipsecon.org/2009/sessions?id=641> [3]

Note: this talk only covers data binding

List of Slides

- [Abstract](#)
- [List of Slides](#)
- [EclipseCon 2009 Tutorial](#)
- [The Result](#)
- [Data Binding Tour](#)
- [Vision](#)
- [Model View Controller](#)
- [From Triangle to Straight Line](#)
- [Concepts](#)
- [IObservable](#)
- [Binding](#)
- [DataBindingContext](#)
- [The RCP Mail 2.0 Model](#)
- [Property Change Support](#)
- [RCP Mail Classes](#)
- [Databinding Plug-ins](#)
- [Where Data Binding runs](#)
- [Factories for Observables](#)
- [Binding Text Fields](#)
- [Converters and Validators](#)
- [Data binding tree content provider](#)
- [Data Binding Features](#)
- [How-to](#)
- [Pros and Cons](#)
- [Next steps](#)
- [Acknowledgements](#)
- [External Links](#)

EclipseCon 2009 Tutorial

RCP Mail 2.0 started out as a group effort by 3 UI committers and 3 API users

For each of the 3 areas covered a committer and a user sort of pair-programmed the solution

eclipse CON™ 2009

Home

Monday Tutorials

Tuesday

Wednesday

Thursday

Short Talks

Evening BOFs

Sponsors

Presenters

Other Conferences

RCP Mail 2.0: Commands, Common Navigator, and Data Binding

Michael Scharf (Wind River), Kai Toedter (Siemens AG), Boris Bokowski (IBM), Francis Upton IV, Frank Gerhardt, Paul Webster (IBM)

Eclipse Platform - UI / RCP · Tutorial - 4 hours · (view in LinkedIn)

Monday, 08:00, 4 hours | Room 202

7 · 8 · 9 · 10 · 11 · 12 · 13 · 14 · 15 · 16 · 17 · 18

[Edit this talk](#)

The RCP Mail example shows its age. Many new and useful features have been added to the Eclipse Platform since the example was written, and it is time to take the example to new levels.

In a hands-on way, we will bring the example up to date by adopting some of the more interesting new APIs developed since the original RCP Mail example was written.

In particular, we will show the following:

- How to use the new Commands API to contribute to menus, toolbars, and context menus, and how to create key bindings.
- How to use the Common Navigator to show tree-shaped data structures that are not workspace resources.
- How to use the data binding framework to make the UI code easier to write, and easier to test.

Explanations of the concepts will be given by members of the Eclipse Platform UI team, while the concrete examples will be explained by experienced Eclipse professionals.

We will also show the different tools that are provided by Eclipse PDE and JDT to help you developing RCP applications.

Michael Scharf is one of the architects of the Wind River Workbench, a CDT based IDE for embedded development. He works for WindRiver since 1994. Earlier in his career, he worked for 9 years in the area of computational molecular biology using object oriented technology for analysis and visualization of complex data. He is active in the eclipse community since 2001.

Kai Tödter is software engineer in the architecture department of Siemens Corporate Technology. He has more than 10 years of professional Java experience. His main interest is software architecture for smart clients and he focuses on Java rich client platforms like Eclipse RCP. Kai is Siemens' primary contact in the Eclipse Foundation.

Boris Bokowski is a Software Developer with IBM Rational in Ottawa, Canada and a full-time Eclipse committer working on the Platform UI team and the new e4 project. He is part of the "API police" for the Eclipse Platform, and a member of the Eclipse Architecture Council. Boris is looking at the UI side of the RCP, and among other areas owns the JFace viewers component. He is also the main architect of the JFace data binding framework. He holds a PhD in computer science from Freie Universität Berlin, Germany.

Francis Upton IV is a committer in the Platform UI and Documentation projects and is currently responsible for the maintenance of the Common Navigator Framework. For his day gig, he runs Oakland Software, one of the smaller bay area software product companies where he has been completely consumed with a passion for data transformation for the last 5 years. Francis has spent most of his career working on system software (the type that no users sees) at Digital, Hewlett Packard, Forte Software, and Vibria and remains surprised that he ended up a UI

Gold sponsors

a division of Red Hat

by ACTUATE

SOPERA

AGILE MODEL TRANSFORMATION

Sun
microsystems

Silver sponsors

INNOVENT
SOLUTIONS

Instantiations

EclipseSource

genuitec

SAP BusinessObjects

Software | Consulting | Coaching

Google

innovations
Software Technology
Bosch Group

Hardware Sponsor

CISCO

Lanyard Sponsor

virtual literature bin

upcoming

[LinkedIn](#)

[facebook](#)

[twitter](#)

[Blogger](#)

[ATOM 1.0 FEED](#)

March 23rd-26th
Santa Clara,
California

Michael Scharf

Kai Toedter

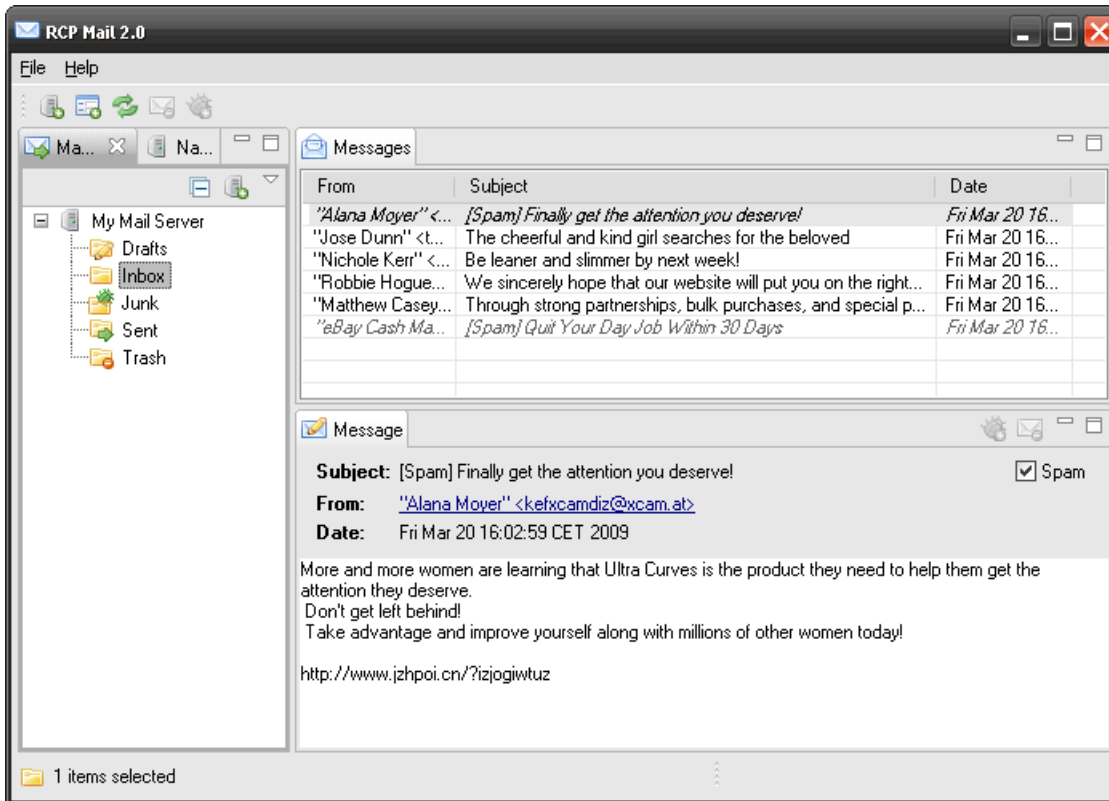
Boris Bokowski

Francis Upton IV

Frank Gerhardt

Paul Webster

The Result

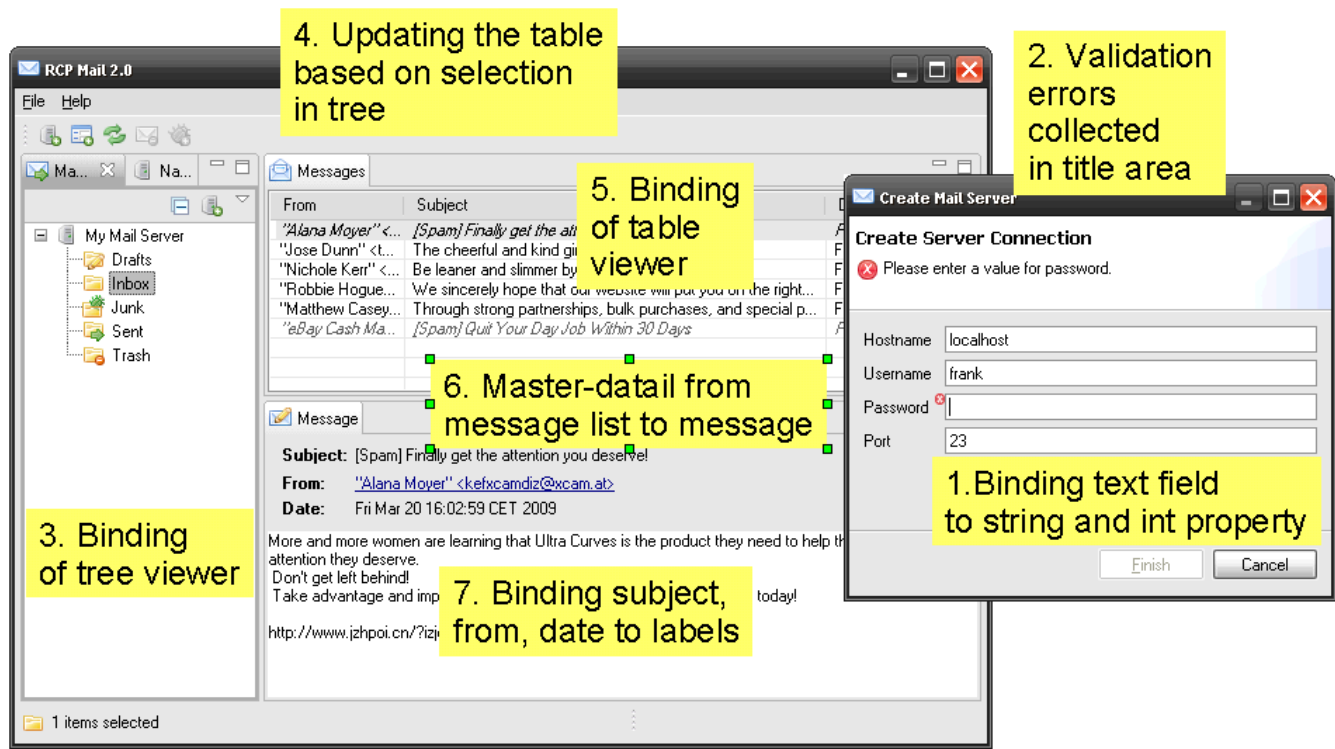


1 minute demo of the features

Data Binding Tour

We introduce data binding in four areas

1. **Wizard** to create a new mail server (**new in 2.0**)
2. **Tree** for servers and folders
3. **Table** of messages (**new in 2.0**)
4. Message display



At the end of the data binding tour will be an exercise where you will add your own binding to a text field in the wizard.

Optional: new column, or third level in tree.

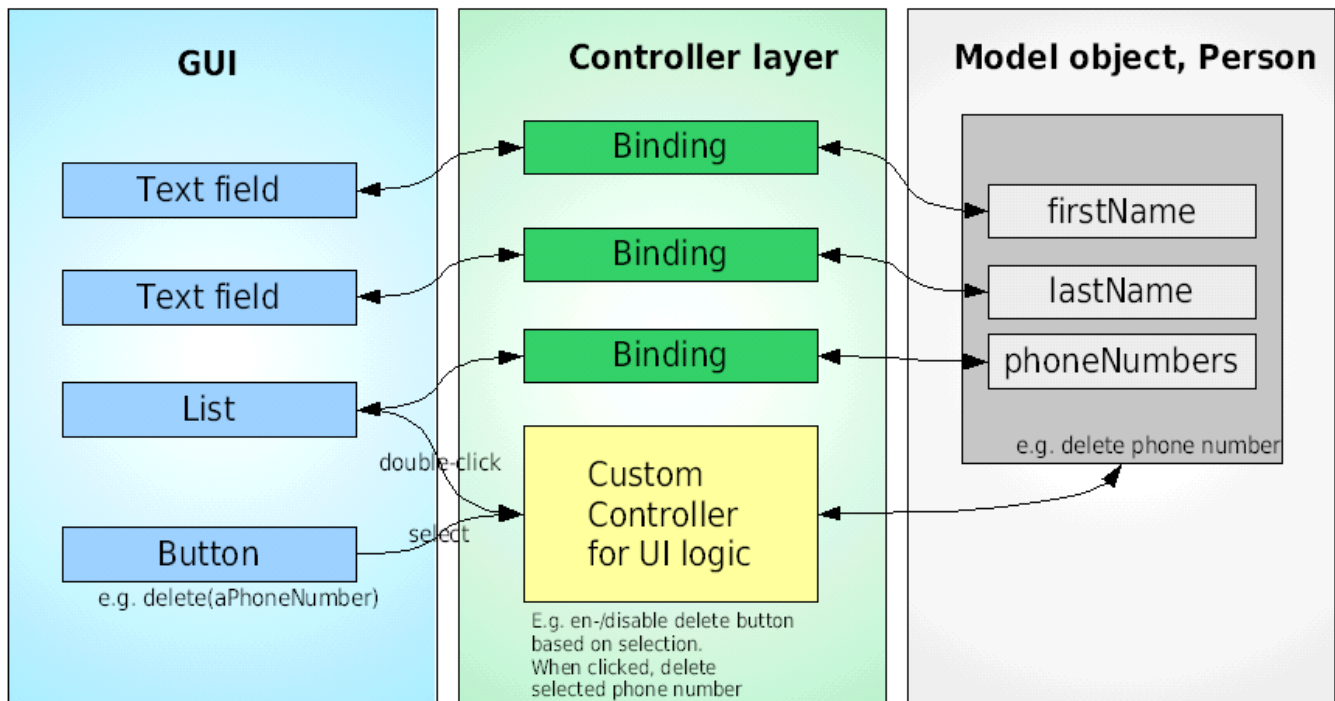
Vision

Why data binding?

Get rid of listeners in UI code!

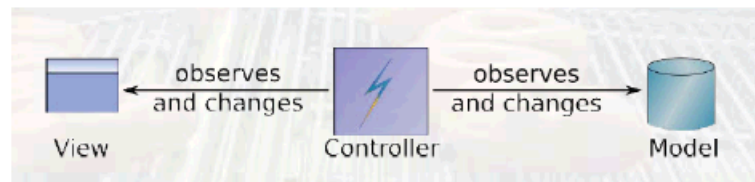
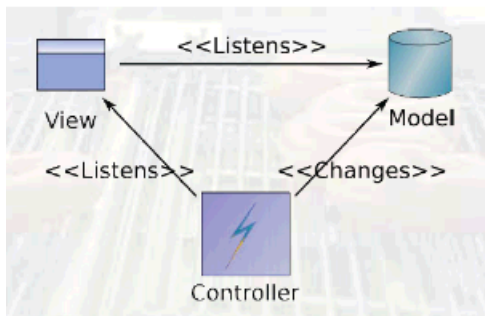
- Hard to write, hard to maintain.
- For every aspect:
 - Copy initial state into widget
 - Hook listener (to widget, to model)
 - Write code to sync state incrementally
 - Validation, conversion typically not separated
 - Threading

Model View Controller



From Triangle to Straight Line

From model-view-controller (MVC) triangle to more **independence**

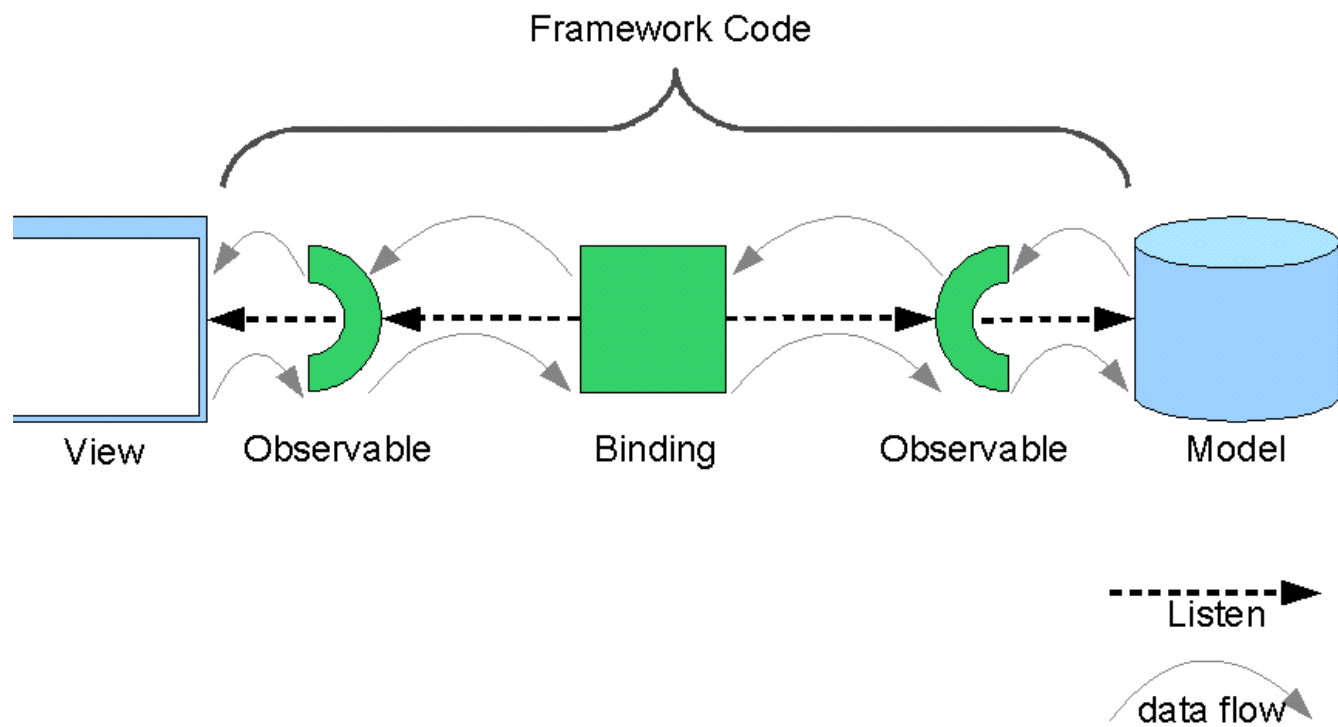


From David Orme, Introducing JFace Data Binding, EclipseCon 2006, made available under EPL 1.0

Concepts

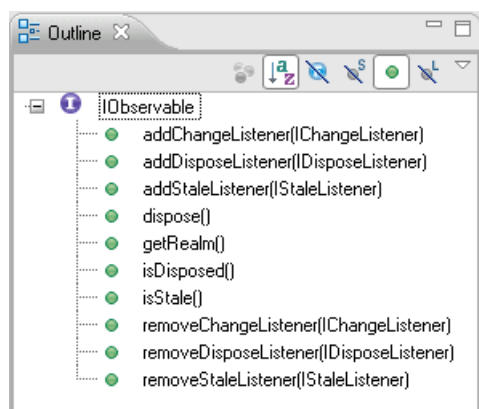
The two main concepts, and *layers* are

- Observables
- Bindings



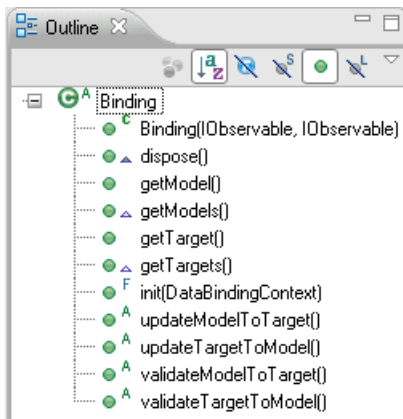
Note: by convention UI always left, model always right, on **diagrams** and in the **API**, see e.g. `DataBindingContext`, `ViewerSupport`

IObservable



This interface makes listening to changes uniform

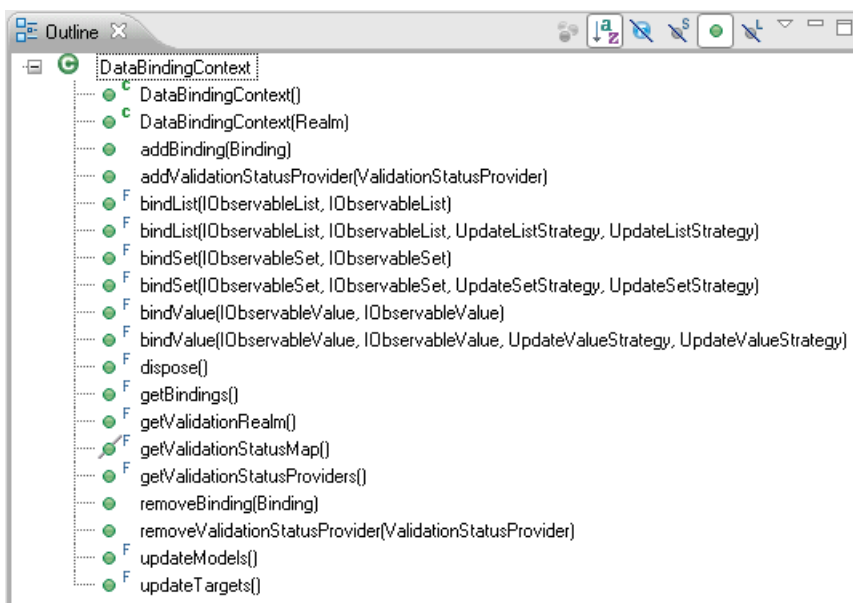
Binding



Represents the binding between two IObservables

Needs to be added to a DataBindingContext

DataBindingContext



Creation and management of Bindings

Aggregates validation statuses

The RCP Mail 2.0 Model

The model is a simple JavaBean model

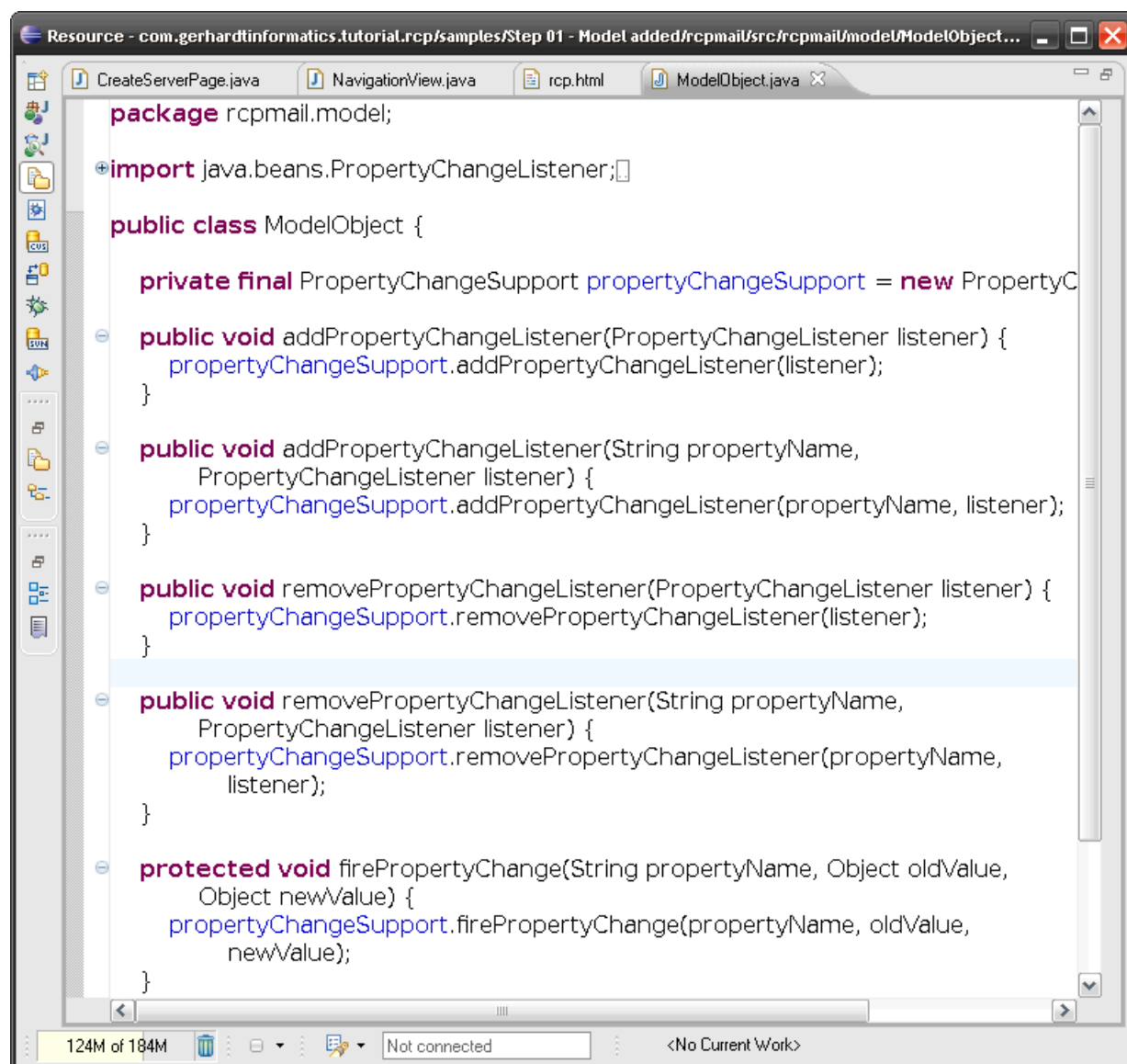
ModelObject is the base class providing PropertyChangeSupport

Changes to the template generated by the SDK

- Model, Server, Folder, Message
- Increased bundle version to 2.0.0
- Including test data
- Icons, splash

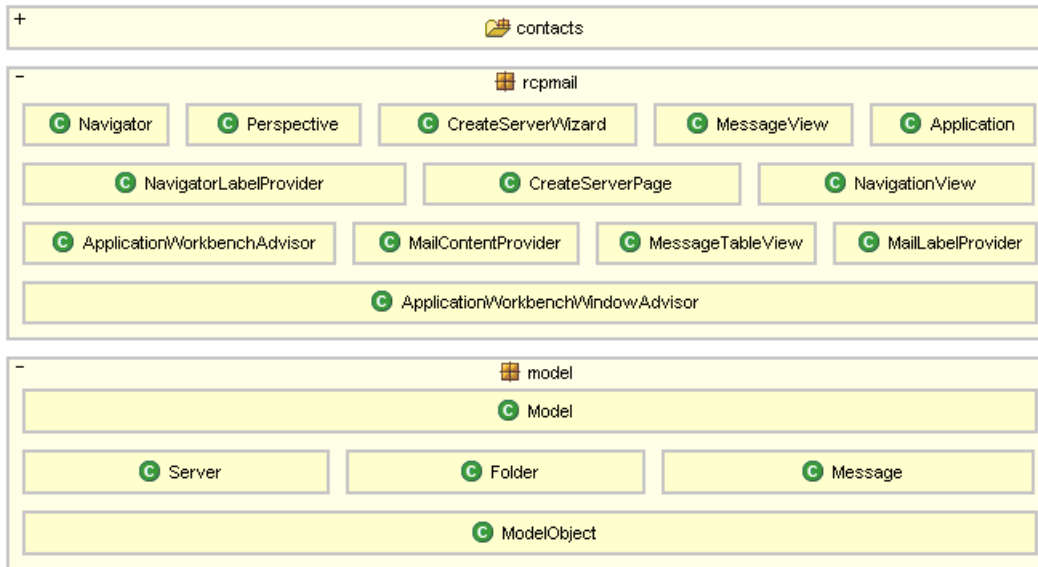
Look at the model in the rcpmail.model package of rcpmail-01

Property Change Support

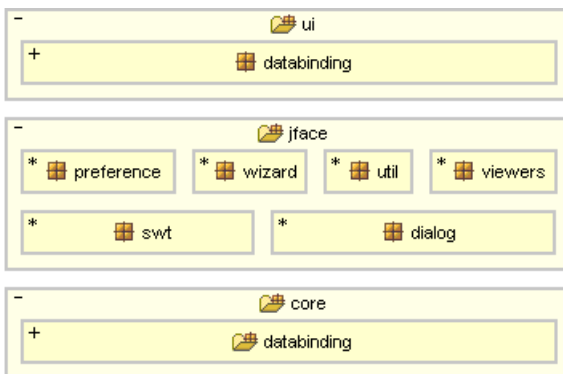


```
Resource - com.gerhardinformatics.tutorial.rcp/samples/Step 01 - Model added/rcpmail/src/rcpmail/model/ModelObject...
CreateServerPage.java  NavigationView.java  rcp.html  ModelObject.java
package rcpmail.model;
import java.beans.PropertyChangeListener;
public class ModelObject {
    private final PropertyChangeSupport propertyChangeSupport = new PropertyC
    public void addPropertyChangeListener(PropertyChangeListener listener) {
        propertyChangeSupport.addPropertyChangeListener(listener);
    }
    public void addPropertyChangeListener(String propertyName,
        PropertyChangeListener listener) {
        propertyChangeSupport.addPropertyChangeListener(propertyName, listener);
    }
    public void removePropertyChangeListener(PropertyChangeListener listener) {
        propertyChangeSupport.removePropertyChangeListener(listener);
    }
    public void removePropertyChangeListener(String propertyName,
        PropertyChangeListener listener) {
        propertyChangeSupport.removePropertyChangeListener(propertyName,
            listener);
    }
    protected void firePropertyChange(String propertyName, Object oldValue,
        Object newValue) {
        propertyChangeSupport.firePropertyChange(propertyName, oldValue,
            newValue);
    }
}
```

RCP Mail Classes

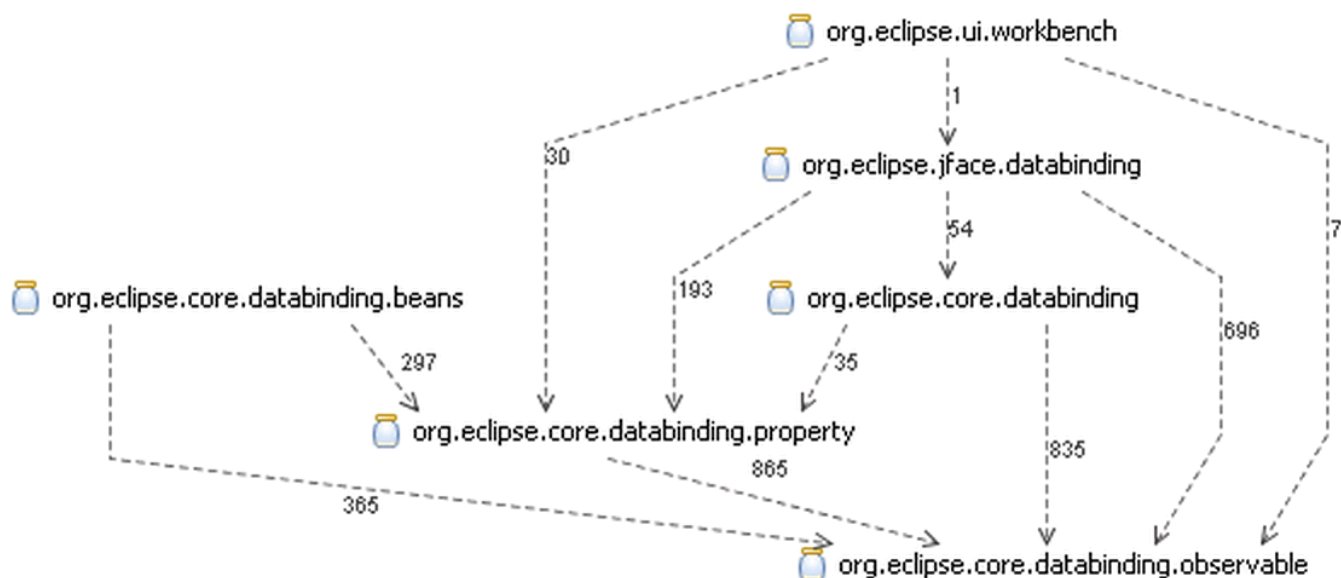


Databinding Plug-ins



Databinding comes in **three layers**

1. WorkbenchObservable in **org.eclipse.ui**
2. **org.eclipse.jface.databinding**
3. Four **core** plug-ins
 - **org.eclipse.core.databinding**
 - **org.eclipse.core.databinding.beans**
 - **org.eclipse.core.databinding.properties**
 - **org.eclipse.core.databinding.observables**



Where Data Binding runs

- SWT/JFace (of course), both RCP and RAP
- Workbench
- EMF
- Google Web Toolkit

Factories for Observables

- Observables
- PojoObservables
- BeansObservables
- SWTObservables
- ViewersObservables
- MasterDetailObservables
- WorkbenchObservables

Binding Text Fields

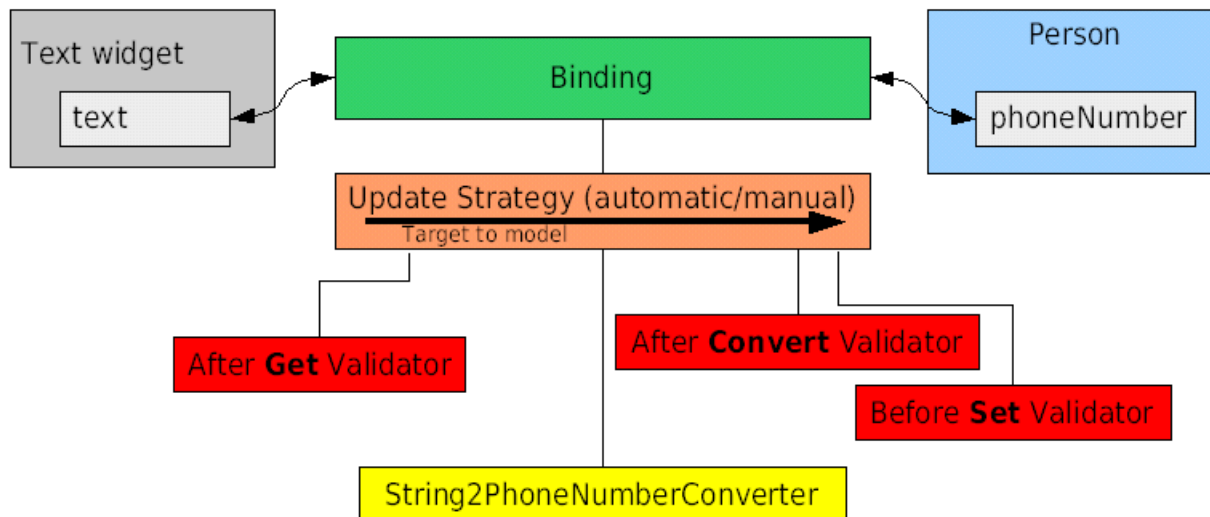
In wizard: binding of text fields to int and string

```

01. | DataBindingContext dbc = new DataBindingContext();
02. | dbc.bindValue(SWTObservables.observeText(hostnameText, SWT.Modify),
03. |               BeansObservables.observeValue(server, "hostname"));
  
```

Converters and Validators

The **Binding** can be configured:
 model to target **update strategy**
 target to model **update strategy**
 each with: **1 Converter** and
3 Validators



```

01. | new UpdateValueStrategy().setBeforeSetValidator(new IValidator() {
02. |     public IStatus validate(Object value) {
03. |         String s = (String) value;
04. |         if (s.contains(" ")) {
05. |             return ValidationStatus.error("no spaces please");
06. |         }
07. |         return ValidationStatus.ok();
08. |     }
  
```

Data binding tree content provider

Goal: For each parent, one `IObservableList` representing children.

To create these lists lazily, we provide a factory returning lists.

```

01. | public IObservable createObservable(Object parent) {
02. |     if (parent instanceof Model) {
03. |         return BeanProperties.list("servers").observe(parent);
04. |     }
05. |     if (parent instanceof Server) {
06. |         return BeanProperties.list("folders").observe(parent);
07. |     }
08. |     return null;
09. | }
  
```

Our (invisible) root element is "Model".

`BeanProperties.list("servers")` is a factory for creating `IObservableLists`.

Second constructor argument is a "TreeStructureAdvisor".

Is consulted when finding an element in the tree that has not been materialized: notice "polish" when starting app.

Second purpose: optimize for elements without children.

Data Binding Features

Binding of

- Values (String, Boolean, Integer...), 1:1
- Trees 1:n, Lists, n:n
- Master-Detail

Can bind to UI state

- Model.locked to Text.enabled
- Color, visibility, many more

Can bind to UI state, selection of list or table

- Zero additional UI logic required

Can bind to validation errors

How-to

Decide which type to use

- Value (1:1), tree (1:n), list (n:n), master detail

For values

- Wrap model into Observable using BeansObservables Factory
- Wrap UI into Observable using SWTObservables or ViewerObservables

For trees, lists

- Use supplied ContentProvider and setInput

Read the example code!!! Monkey see, Monkey do!

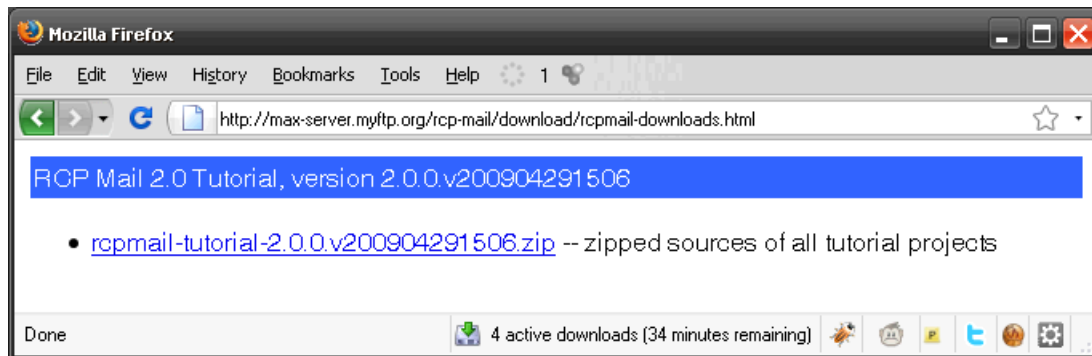
Pros and Cons

- Avoids repetitive, error-prone code
- Works in 80% of the cases
- Set up once, zero effort afterwards
- Eclipse API

- Learning required
- Can be hard to debug
- Need custom solution where it does not work

Next steps

- [Webinar \[4\]](#)
- [JFace Databinding Wiki page \[5\]](#)
- Newsgroup eclipse.platform with [DataBinding]subject
- Bugzilla: Product Platform, Component UI, Summary with [DataBinding]
- Examples and Tests fro nCVS
- Use the source, Luke: **RCP Mail 2.0 source** on [Kai's server. \[6\]](#)



Acknowledgements

This presentation uses the [Slideous \[7\]](#) package by Prof. Stefan Gössner, licensed under the [GNU LGPL License 2.1 \[8\]](#).

External Links

This section lists all hyperlinks included in the presentation. When printing HTML, usually only the blue and underlined hyperlinks are shown and the targets of all hyperlinks are "lost". This handout, when printed (only!), includes a number like a footnote (e.g. [123]) after each hyperlink to refer to the following list of targets.

- [1] <http://wiki.eclipse.org/EclipseApplicationDeveloperDayKarlsruhe>
- [2] <http://creativecommons.org/licenses/by-nc-sa/3.0/us>
- [3] <http://www.eclipsecon.org/2009/sessions?id=641>
- [4] http://admin.adobe.acrobat.com/_a300965365/p77464314
- [5] http://wiki.eclipse.org/index.php/JFace_Data_Binding
- [6] <http://max-server.myftp.org/rcp-mail/download/rcpmail-downloads.html>
- [7] <http://goessner.net/articles/slideous/>
- [8] <http://creativecommons.org/licenses/LGPL/2.1/>