

## Leveraging the Model-View- Presenter Pattern in Rich Client Applications

Patrick Paulin  
Eclipse RCP Trainer and Consultant  
RCP Quickstart

[patrick@rcpquickstart.com](mailto:patrick@rcpquickstart.com)

[www.rcpquickstart.com/training/presentations/mvp-and-rcp-ew2008](http://www.rcpquickstart.com/training/presentations/mvp-and-rcp-ew2008)

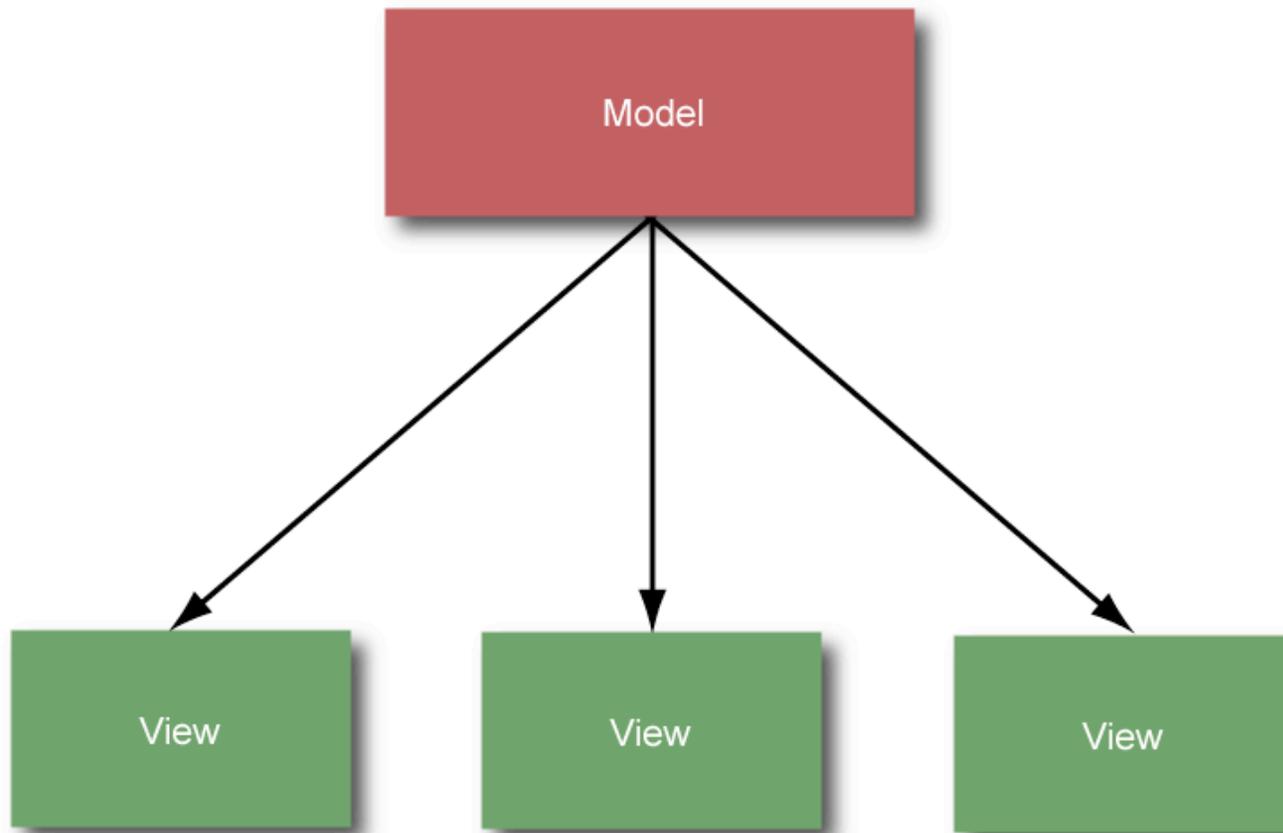
## Agenda

- A brief history lesson
- Presenter pattern in detail
- MVP and RCP
- RAP / RCP dual use demo

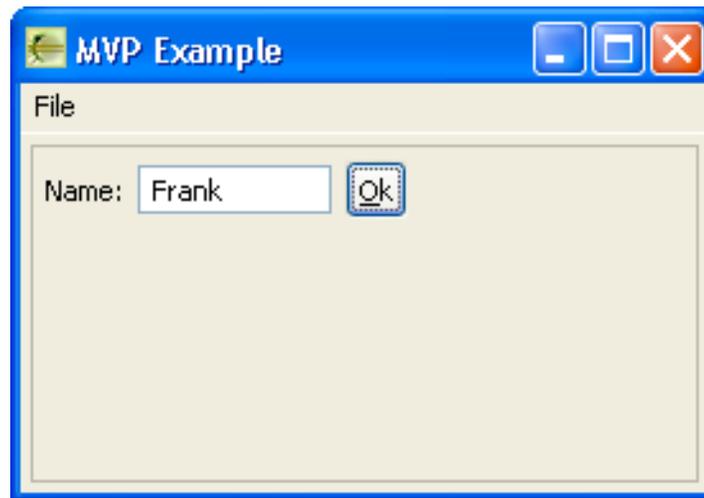
## UI architecture

- Forms and controls
- Model-View-Controller (MVC)
- Model-View-Presenter (MVP)

## Model / View separation

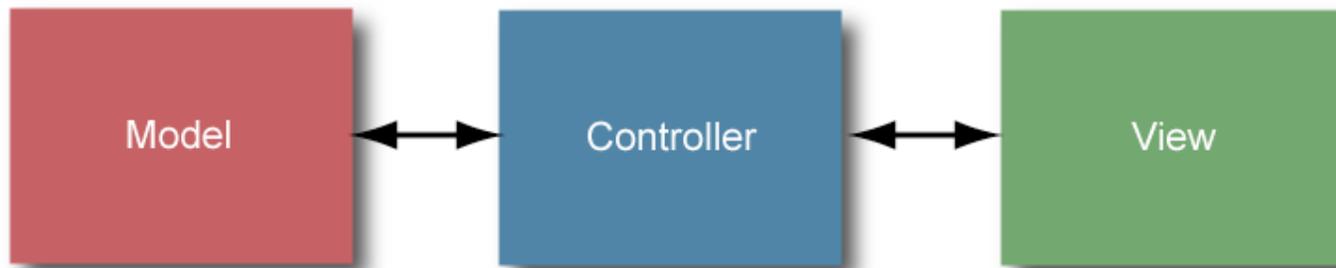


## Forms and controls

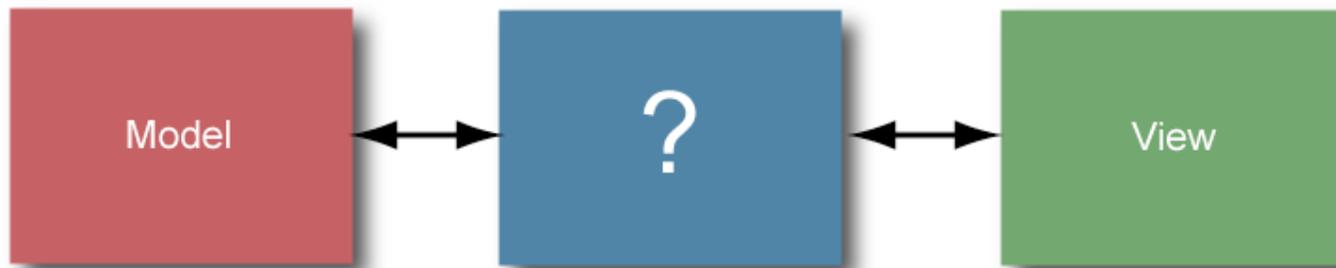


```
Text name = new Text(parent, SWT.BORDER);
name.addModifyListener(new ModifyListener()
    public void modifyText(ModifyEvent e) {
        model.setName(nameText.getText());
    }
});
```

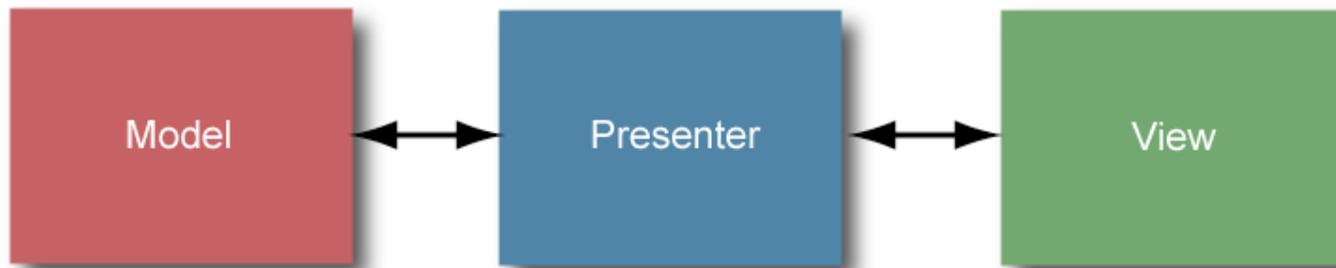
## Model-View-Controller (MVC)



## What is a controller, anyway?



## Model-View-Presenter (MVP)



## Agenda

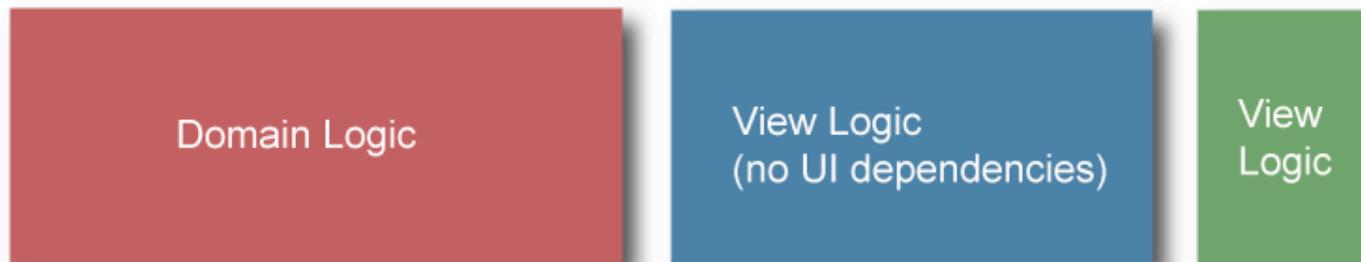
- A brief history lesson
- Presenter pattern in detail
- MVP and RCP
- RAP / RCP dual use demo

## What is MVP?

Before MVP



After MVP



## Why MVP?

- Manage complexity
- Flexibility
- Testability

## Managing complexity

### View Logic - non UI

Driven by use case

Depends on  
`org.eclipse.core.runtime`

Hand coded

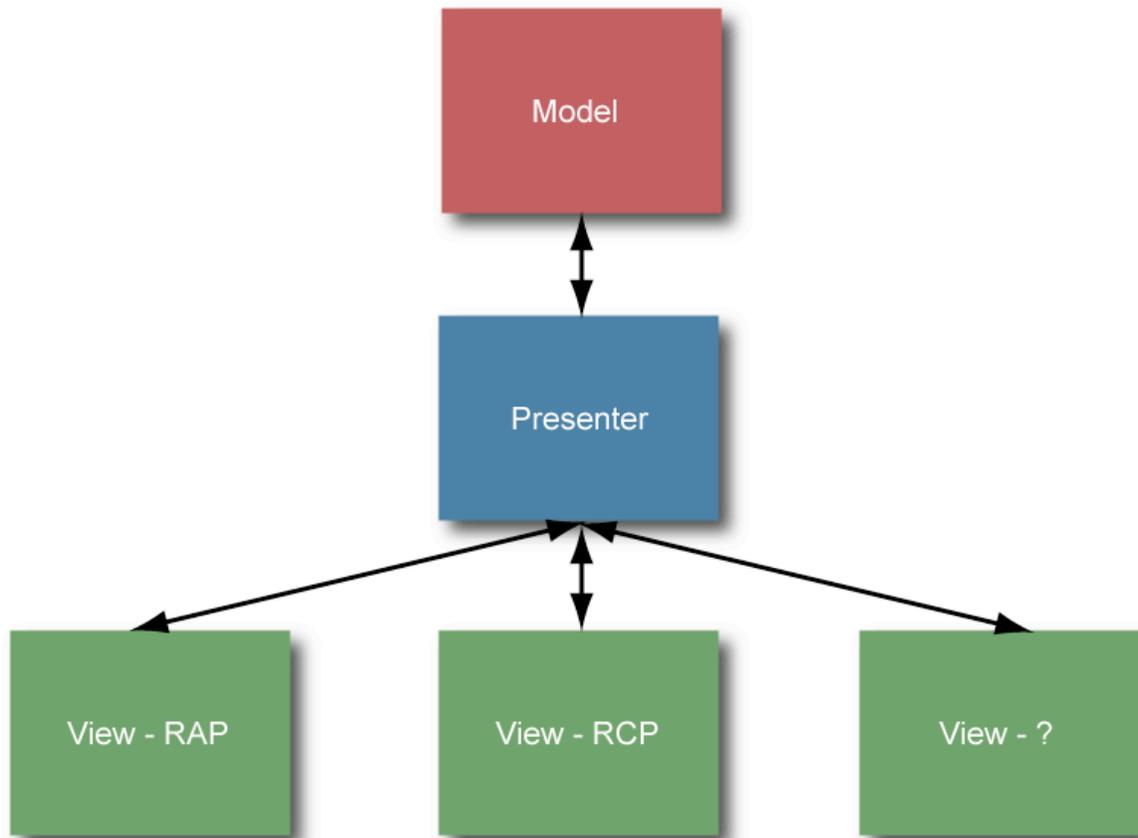
### View Logic - UI

Driven by technology choice

Depends on  
`org.eclipse.ui` (SWT, JFace)

Generated code

## Flexibility



## Testability is hard

- UI testing is hard
- Tooling isn't there yet
- Unit tests must be easy and quick to run

*“Any object that is difficult to test should have minimal behavior.”*

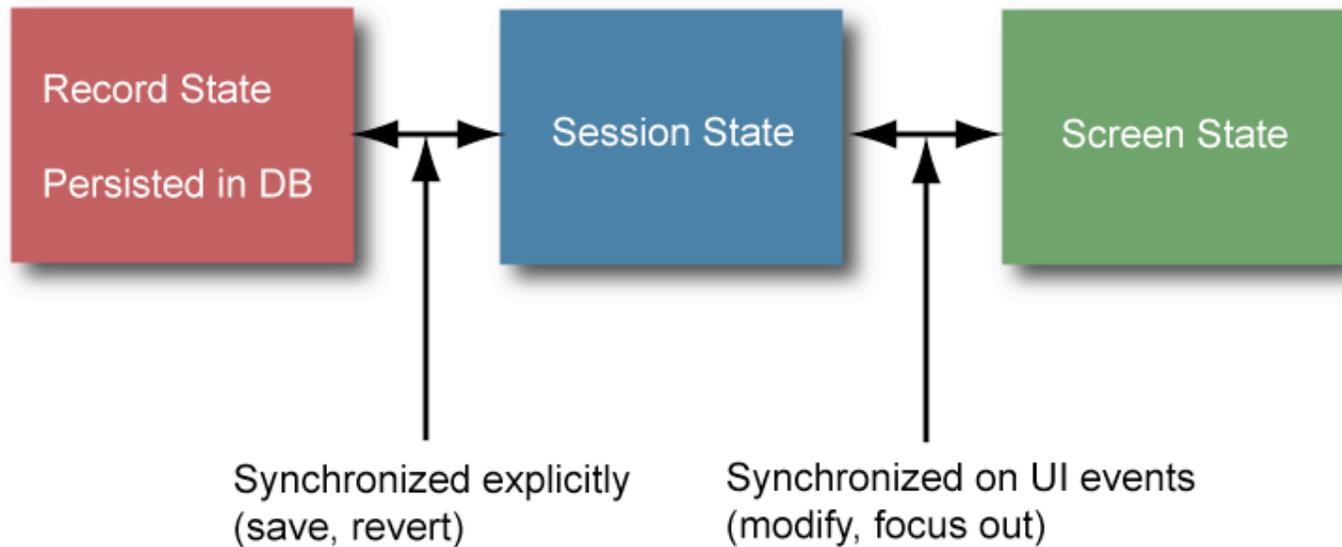
- Martin Fowler

## Acceptance testing

```
public class MyPresenter {  
  
    setProperty(String property)  
    getValidationMessages()  
}
```

```
public class MyPresenterTest {  
  
    @Test  
    testUseCase() {  
        MyPresenter p = new  
            MyPresenter();  
  
        p.setProperty("value");  
        assertEquals(0,  
            p.getValidationMessages().  
                size())  
    }  
}
```

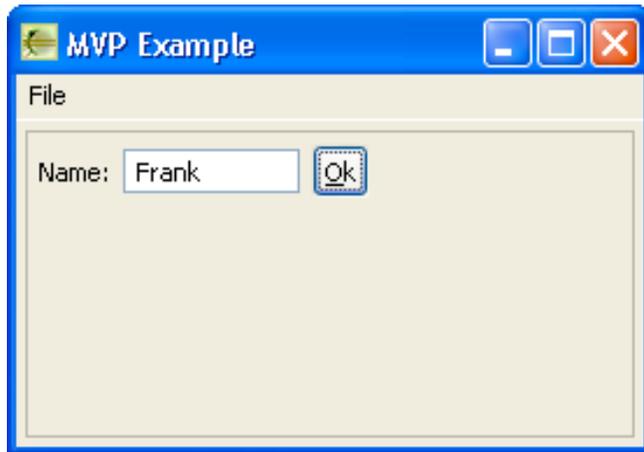
## MVP and data



# RCP Quickstart

---

## MVP and logic

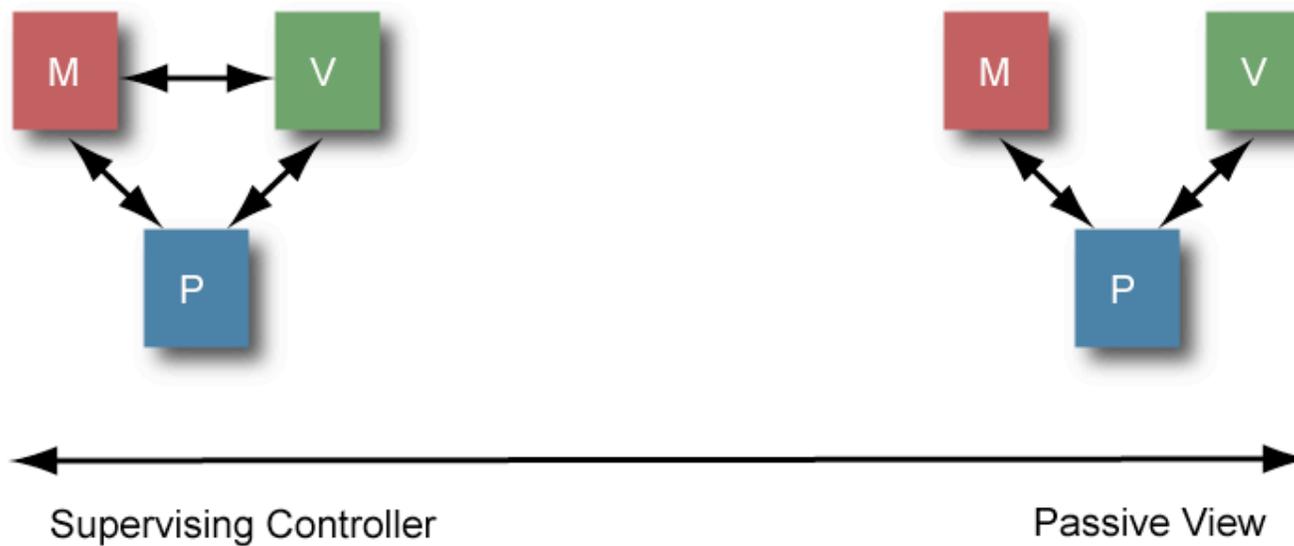


```
public class ExamplePresenter {  
    public void setName(String name)  
    public boolean getOkEnabled()  
}
```

# RCP Quickstart

---

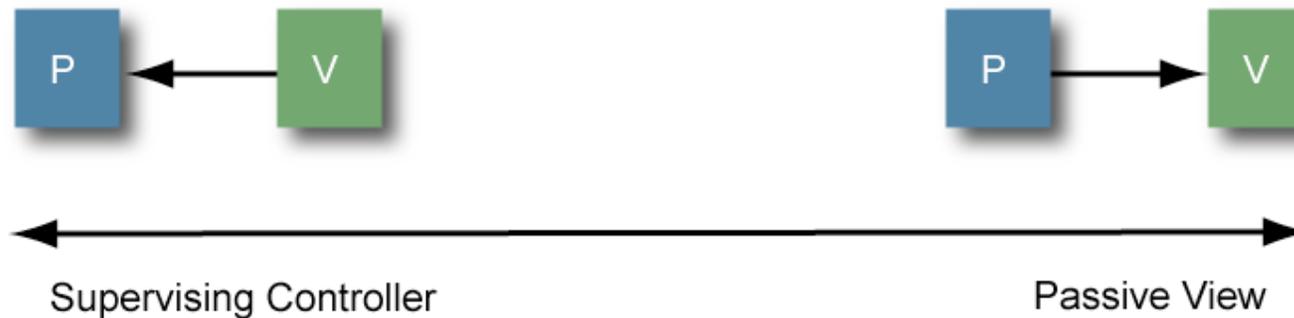
## MVP continuum



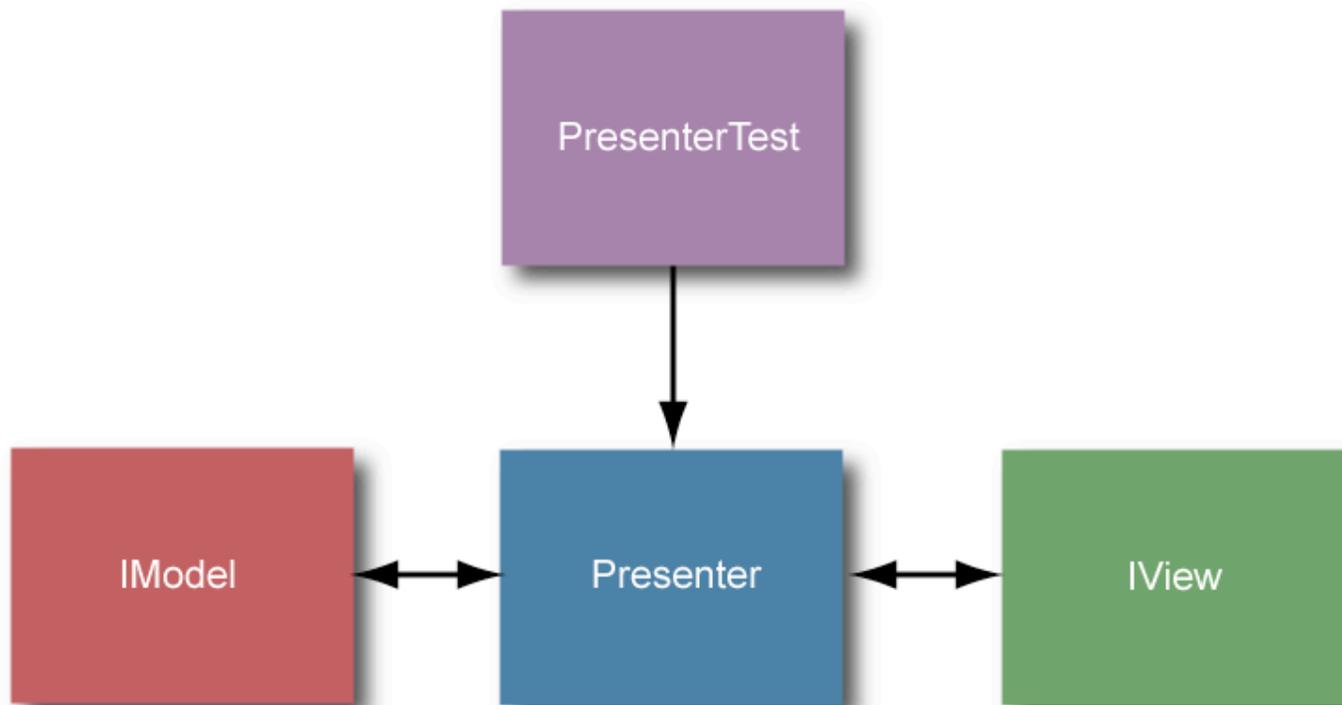
# RCP Quickstart

---

## Push vs pull



## Passive view testing



## Sample test case

### Presenter

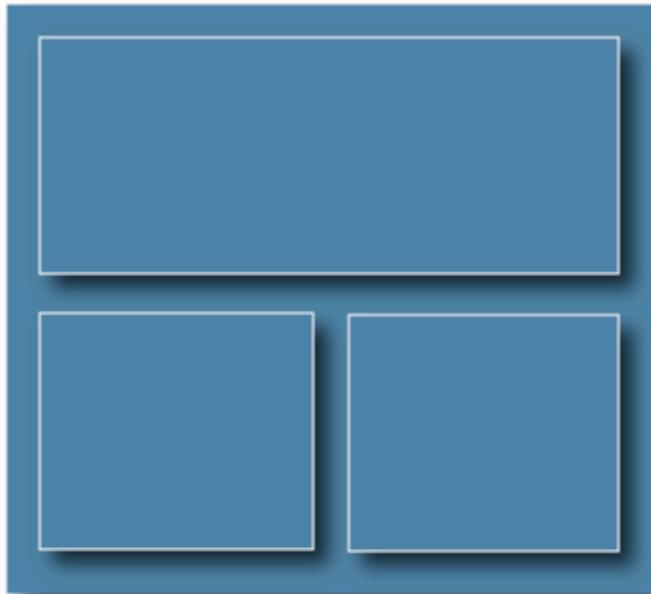
```
public class ExamplePresenter {  
  
    private IView view;  
  
    public ExamplePresenter(IModel model,  
        IView view) {  
        this.view = view;  
  
    public void setName(String name)  
        this.view.setOkEnabled(name !=  
            null);  
  
}
```

### Presenter Test Case

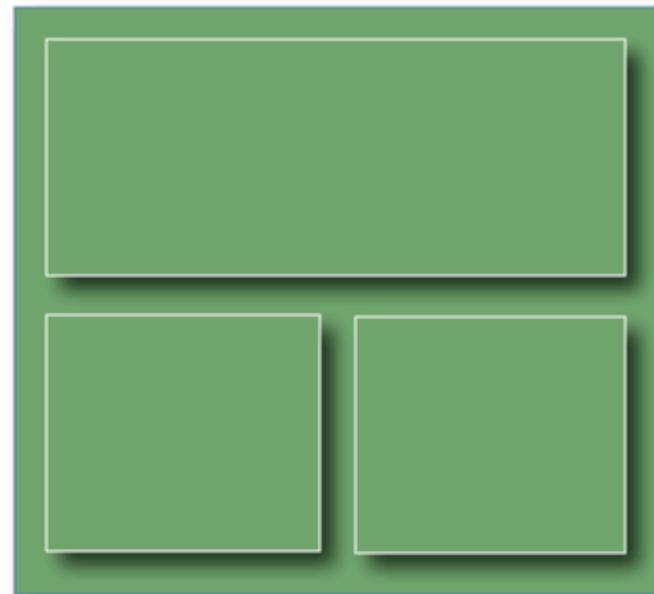
```
public class ExamplePresenterTest {  
  
    protected void testSetName() {  
        // create mock model and view  
  
        ExamplePresenter presenter = new  
            ExamplePresenter(mockModel,  
                mockView);  
  
        presenter.setName("Frank");  
  
        assertTrue(mockView.isOkEnabled());  
  
}
```

## Composite presenters

Presenters



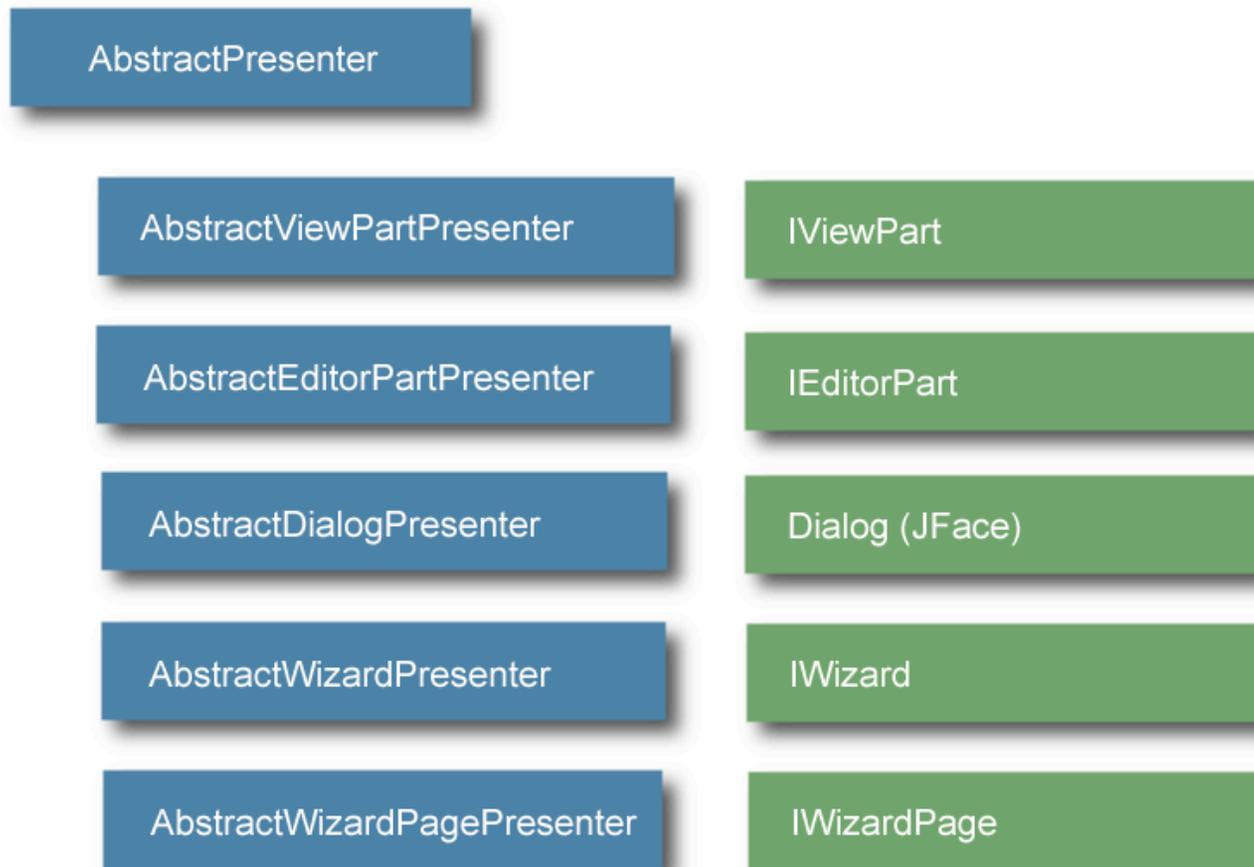
Views



## Agenda

- A brief history lesson
- Presenter pattern in detail
- MVP and RCP
- RAP / RCP dual use demo

## Presenter hierarchy



## AbstractPresenter

AbstractPresenter

AbstractViewPartPresenter

AbstractEditorPartPresenter

AbstractDialogPresenter

AbstractWizardPresenter

AbstractWizardPagePresenter

- Composite pattern
- Property change management (if data binding is used)

## AbstractViewPartPresenter

AbstractPresenter

AbstractViewPartPresenter

AbstractEditorPartPresenter

AbstractDialogPresenter

AbstractWizardPresenter

AbstractWizardPagePresenter

- Create in IViewPart.init()
- Route saveState()
- Mementos
- Filtering
- Sorting

## AbstractEditPartPresenter

AbstractPresenter

AbstractViewPartPresenter

AbstractEditorPartPresenter

AbstractDialogPresenter

AbstractWizardPresenter

AbstractWizardPagePresenter

- Create in IEditorPart.init() or extract from editor input
- Dirty state management
- Save and revert
- Form header messages
- Can have nested page and section presenters

## AbstractDialogPresenter

AbstractPresenter

AbstractViewPartPresenter

AbstractEditorPartPresenter

AbstractDialogPresenter

AbstractWizardPresenter

AbstractWizardPagePresenter

- Create in constructor
- Handle button enablement
- Handle button presses

## AbstractWizardPresenter

AbstractPresenter

AbstractViewPartPresenter

AbstractEditorPartPresenter

AbstractDialogPresenter

AbstractWizardPresenter

AbstractWizardPagePresenter

- Page management (page presenters or constants)
- Determines if wizard can finish
- Performs finish or cancel logic

## AbstractWizardPagePresenter

AbstractPresenter

AbstractViewPartPresenter

AbstractEditorPartPresenter

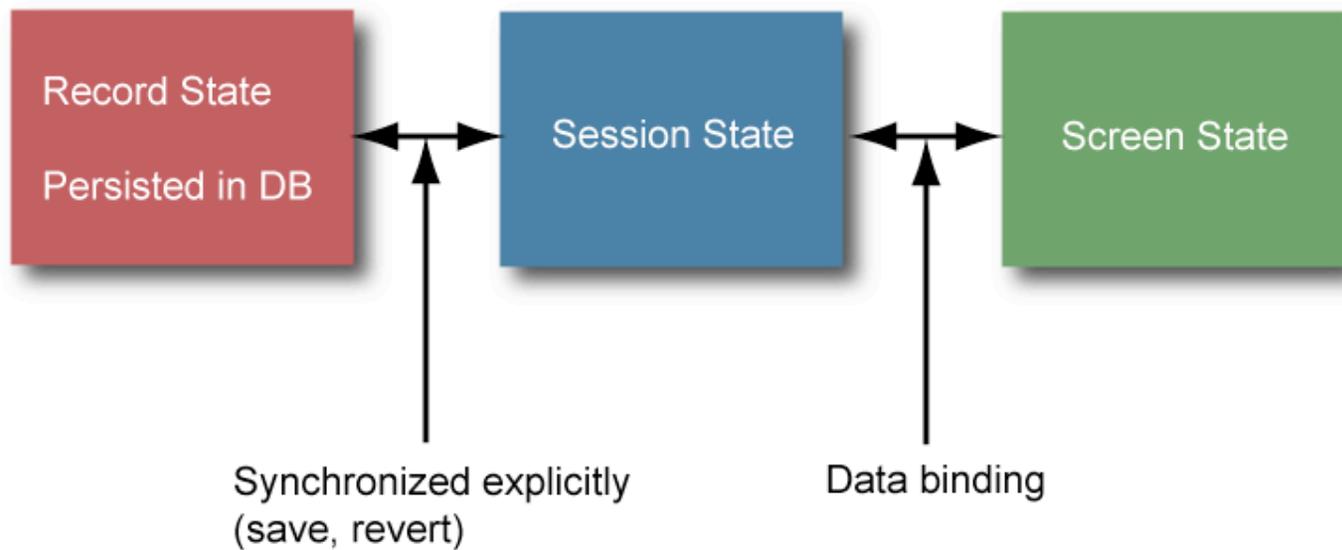
AbstractDialogPresenter

AbstractWizardPresenter

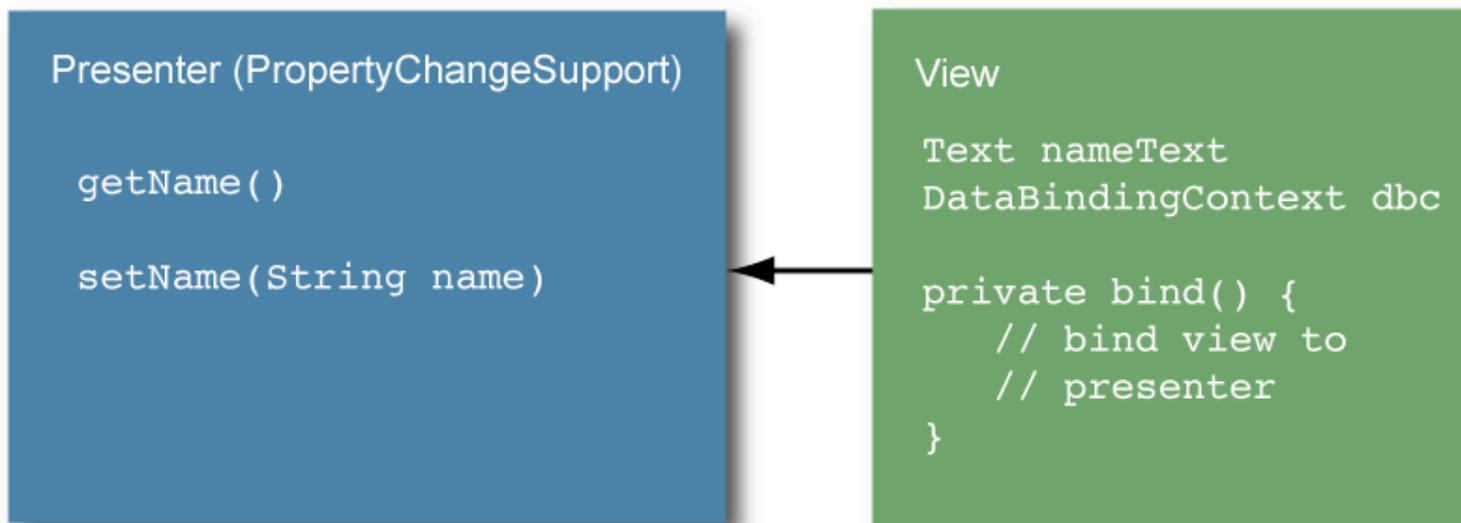
AbstractWizardPagePresenter

- Page management
- Handles isPageComplete()

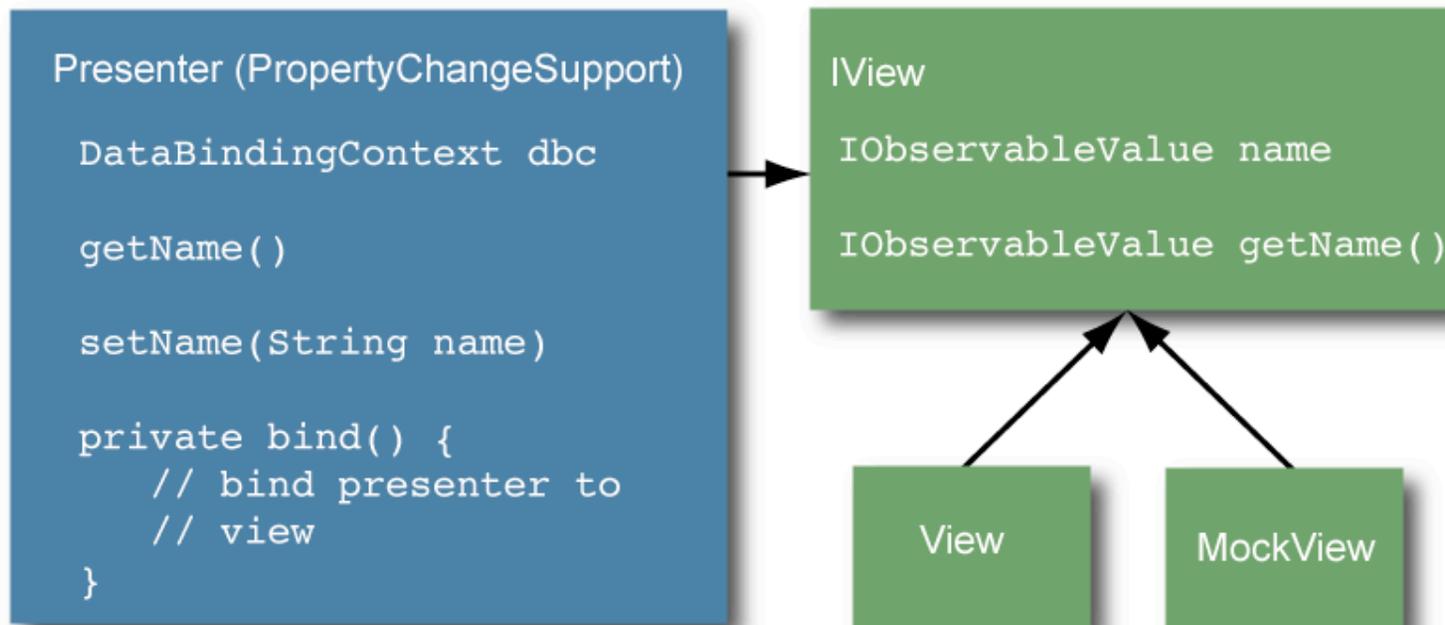
## Data binding



## Data binding - view in control



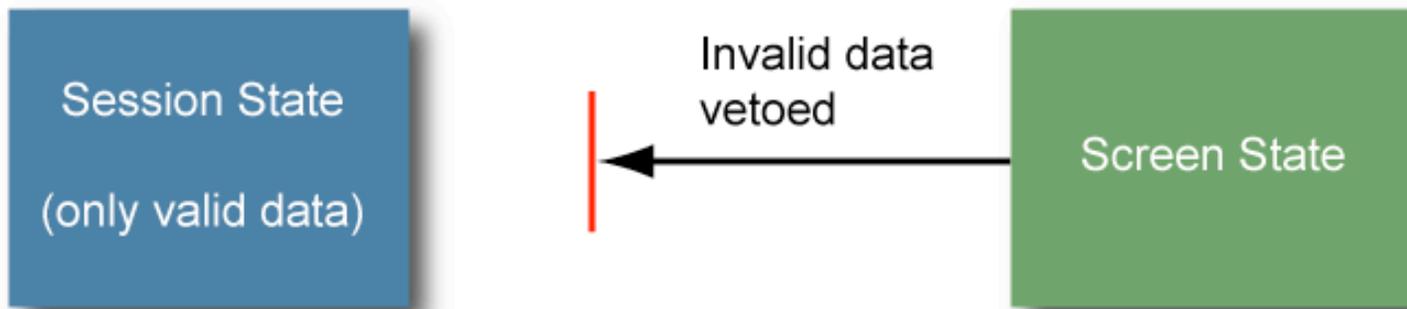
## Data binding - presenter in control



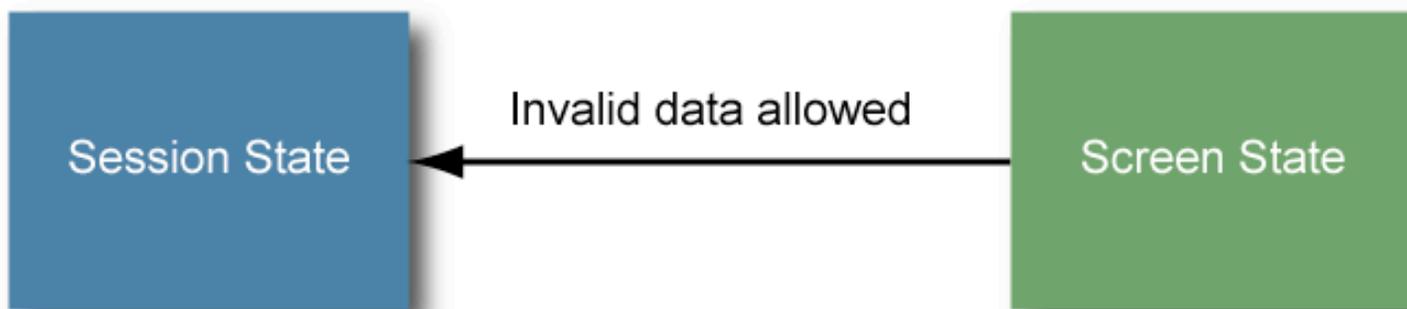
# RCP Quickstart

---

## Data binding validation



## Presenter validation



## Data binding - beyond data

- Control enablement
- Selections
- View still needs to call presenter to take action (e.g. `okPressed`)
- Presenter can set a listener into view if full control is desired

## Agenda

- A brief history lesson
- Presenter pattern in detail
- MVP and RCP
- **RAP / RCP dual use demo**

## What is RAP?

- Rich Ajax Platform
- Parallel UI toolkit including SWT, JFace and RCP Workbench
- UI classes run on server and emit Javascript to browser
- Now supports data binding!
- <http://www.eclipse.org/rap/>

# RCP Quickstart

## RAP Example

The screenshot shows a web browser window displaying the RAP Showcase application. The browser's address bar shows the URL `http://rap.innoo pract.com/rap`. The application has a blue header bar with the text "RAP Showcase". Below the header, there are navigation links: "User Help", "Login/Logoff", "Registration", "Add talk to my schedule", and "Help Content".

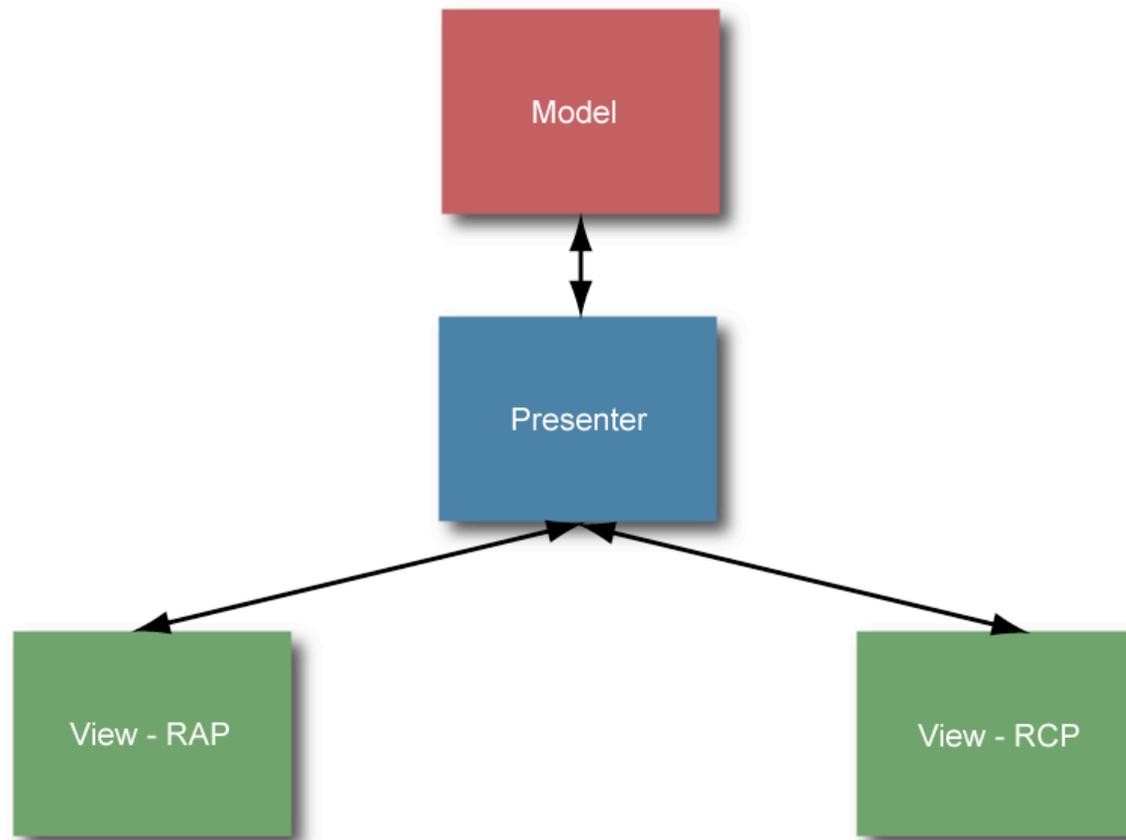
The main content area is divided into several sections:

- Explorer:** A sidebar menu with items: "Me: (please login)", "Contacts", "Outgoing contact requests", and "Incoming contact requests".
- Tracks:** A sidebar menu with items: "Database Development Track", "Director's Choice Track", "(BOF) Committer and Pro...", "(L) Login and Go: Flipping", "(L) Migrating a Visual Stuc...", "(L) RAP - Eclipse style dev...", "(P) Eclipse Community Prc...", and "(S) Due Diligence - Why d...".
- Map:** A central map showing a location in West Valley, Mission College. The map includes navigation controls (arrows, zoom in/out) and a "Map" label.
- Talk Details:** A section with tabs for "Map", "Talk Details", and "Intro".
- Search and Schedule:** Buttons for "Search People", "Talk Comments", and "My Schedule".

At the bottom of the page, there is a table with the following data:

Date/Place	Title
Wed 14:30-15:20 Great America Ballroom JK	RAP - Eclipse style devel

## Presenter dual use



## MVP Summary

- Replacement for MVC
- Flexibility and testability
- MVP and RCP work well together
- MVP prepares you for the future

## More Information

Check out my website:

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

Detailed notes for presentation at:

[www.rcpquickstart.com/training/presentations/mvp-and-rcp-ew2008](http://www.rcpquickstart.com/training/presentations/mvp-and-rcp-ew2008)

Email me:

[patrick@rcpquickstart.com](mailto:patrick@rcpquickstart.com)