HOWTO

CREATE A SIMPLE

RCPFORM APPLICATION

1. Overview

RCPForms-SimpleSample application is embedded in an eclipse view. Start an eclipse application and add the plugin to the runtime.

To get started follow the installation instructions in Appendix A: SimpleSample Template installation.

2. Create ViewPart

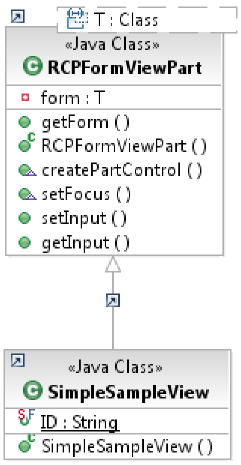


Figure 1 Hierarchy of the `SimpleSampleView`

```
[…]
<extension point="org.eclipse.ui.views">
   <view allowMultiple="true"
      class="net.sf.rcpforms.simplesample.example.SimpleSampleView"
      icon="icons/arrow_right_blue.gif"
      id="net.sf.rcpforms.examples.simplesampleview"
      name="Simple Sample for RCPForms"
      restorable="true">
   </view>
</extension>
[…]
```

Listing 1 Snippet from the plugin.xml

```
    public SimpleSampleView()
    {
        // create
        super(new SimpleSampleStackForm());
        // and set input
        setInput(SimpleSampleStackForm.createModels());
    }
```

Listing 2 Adapt constructor

3. Create Form with Formpart

3.1 Form

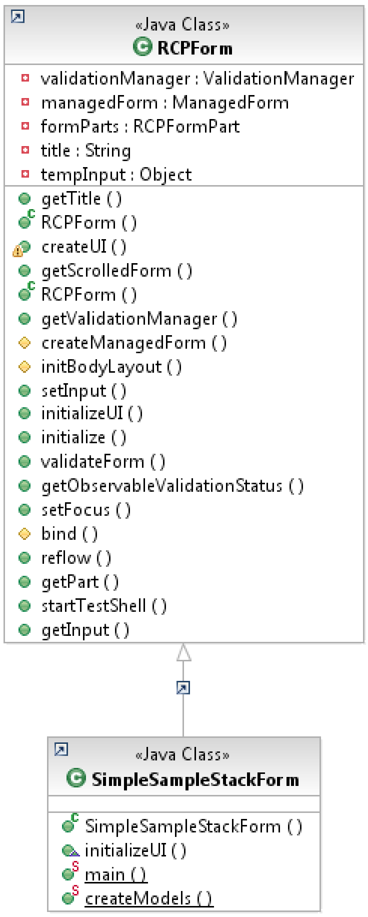


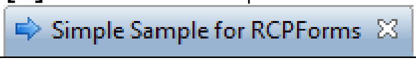
Figure 2 Hierarchy of the `SimpleSampleStackForm`

```
1       public SimpleSampleStackForm()
2       {
3           // create form with the given title and form parts
4           super("SimpleSample for RCPForms", new PersonFormPart());
5       }
```

Listing 3 Adapt constructor

What happened:

[4] Create the rcpform with the given string as title for the eclipse formpart:



3.2 FormPart

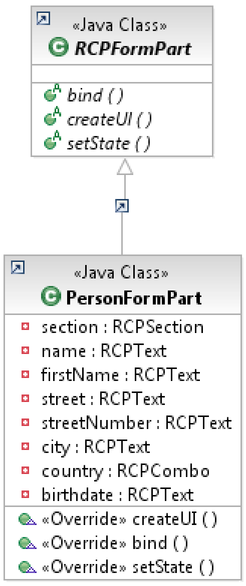


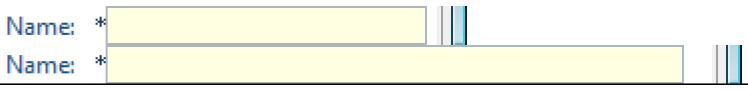
Figure 3 Hierarchy of the `PersonFormPart`

```
1      @Override
2      public void createUI(FormToolkit toolkit, Composite parent)
3      {
4          //create RCPForm elements
5          section = new RCPSection("This title will be invisible",
6  Section.NO_TITLE);
7          name = new RCPText("Name: ");
8          name.setState(EControlState.MANDATORY, true);
9          firstName = new RCPText("Firstname: ");
10         firstName.setState(EControlState.MANDATORY, true);
11         street = new RCPText("Street/Number: ");
12         streetNumber = new RCPText(null);
13         city = new RCPText("City: ");
14         country = new RCPCombo("Country: ");
15         birthdate = new RCPText("Birthdate: ");
16         birthdate.setState(EControlState.MANDATORY, true);
17
18         //add elements to container
19         GridBuilder formPartBuilder = new GridBuilder(toolkit, parent, 2);
20         GridBuilder sectionBuilder = formPartBuilder.addContainer(section, 4);
21         //1st line
22         sectionBuilder.addLineGrabAndFill(name, 3);
23         //2nd line
24         sectionBuilder.addLine(firstName);
25         //3rd line
26         sectionBuilder.add(street);
27         sectionBuilder.add(streetNumber);
28         sectionBuilder.fillLine();
29         //4th line
30         sectionBuilder.addLine(city);
31         //5th line
32         sectionBuilder.addLine(country);
33         //6th line
34         sectionBuilder.addLine(birthdate, 10);
35     }
```

Listing 4 Build `createUI()`-Method

☞ **What happened:**

`[5,6]`	Create main section (**`Section.NO_TITLE`** hides title area)
`[7,16]`	Create and configure rcpforms objects
`[7]`	Declare the `name` field as mandatory.
`[19]`	Create main builder
`[20]`	Create builder for main section

	[21-34] Add all the rcpforms stuff; the swt widgets itself will be created at this time, not earlier. *Special:* **[20]** text will grab 3 columns an fill the existing space. Resizing the form will also resize the text. 

```
1       @Override
2       public void setState(EControlState state, boolean value)
3       {
4           section.setState(state, value);
5       }
```

Listing 5 Add `setState()`-Method

☞	**What happened:** **[4]** The new state will be set to the whole (main) section of this formpart.

Add following import statements:

```
import net.sf.rcpforms.widgetwrapper.builder.GridBuilder;
import org.eclipse.ui.forms.widgets.Section;
```

Start application now, open the view (Alt+Shift+q, q) and you should get the following screen:

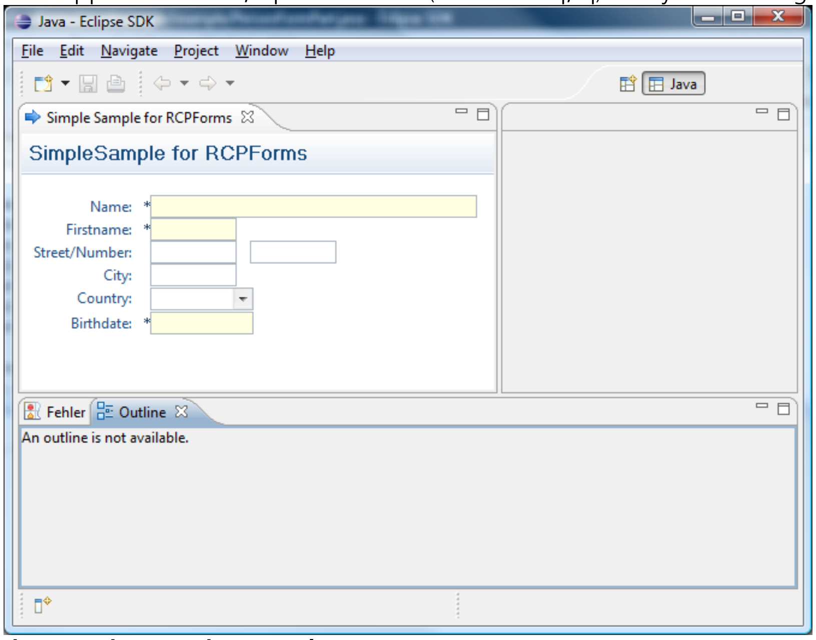

Java - Eclipse SDK
File Edit Navigate Project Window Help
Java
Simple Sample for RCPForms
SimpleSample for RCPForms
Name: *
Firstname: *
Street/Number:
City:
Country:
Birthdate: *
Fehler
Outline
An outline is not available.

Figure 4 First running sample

4. Create DataModel

Recommended to use only objects in datamodel, not primitive data types. Reason: three states: null, value or empty value. E.g. String:

- `null` ⇨ not set
- `foo bar` ⇨ set
- “” ⇨ set, but deleted

Other reason: using primitive types -> initialization with ‘`0`’ ⇨ ‘`0`’ will be shown in text field ⇨ is initially not empty.

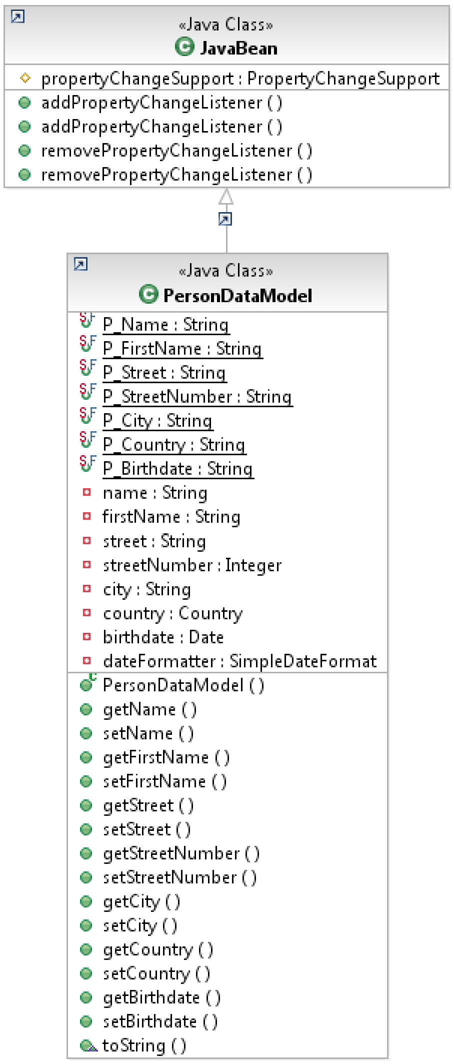


Figure 5 Sample hierarchy of the `PersonDataModel` model

```
public String getName()
    {
        return name;
    }
    public void setName(String name)
    {
        Object oldValue = this.name;
        this.name = name;
        propertyChangeSupport.firePropertyChange(
        PersonDataModel.P_Name, oldValue, name);
    }
    public String getFirstName()
    {
        return firstName;
    }
    public void setFirstName(String firstName)
    {
        Object oldValue = this.firstName;
        this.firstName = firstName;
        propertyChangeSupport.firePropertyChange(
        PersonDataModel.P_FirstName, oldValue, firstName);
    }
    public String getStreet()
    {
        return street;
    }
    public void setStreet(String street)
    {
        Object oldValue = this.street;
        this.street = street;
        propertyChangeSupport.firePropertyChange(
        PersonDataModel.P_Street, oldValue, street);
    }
    public Integer getStreetNumber()
    {
        return streetNumber;
    }
    public void setStreetNumber(Integer streetNumber)
    {
        Object oldValue = this.streetNumber;
        this.streetNumber = streetNumber;
        propertyChangeSupport.firePropertyChange(
        PersonDataModel.P_StreetNumber, oldValue, streetNumber);
    }
    public String getCity()
    {
        return city;
    }
    public void setCity(String city)
    {
        Object oldValue = this.city;
        this.city = city;
        propertyChangeSupport.firePropertyChange(
        PersonDataModel.P_City, oldValue, city);
    }
    public Country getCountry()
    {
        return country;
    }
    public void setCountry(Country country)
    {
        Object oldValue = this.country;
        this.country = country;
        propertyChangeSupport.firePropertyChange(
        PersonDataModel.P_Country, oldValue, country);
    }
    public Date getBirthdate()
    {
        return birthdate;
    }
    public void setBirthdate(Date birthdate)
    {
        Object oldValue = this.birthdate;
        this.birthdate = birthdate;
        propertyChangeSupport.firePropertyChange(
        PersonDataModel.P_Birthdate, oldValue, birthdate);
    }
```

Listing 6 create data model

☞ **What happened:**
`[...]` All setters are supposed to fire change events if a new value is set. The field `propertyChangeSupport` is declared and instantiated in super class.`JavaBean`.

5. Connect View and Model

Binding with Model2Target and Target2Model update strategy.

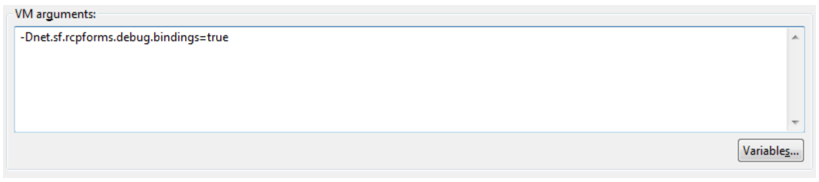
```
1         @Override
2         public void bind(ValidationManager bm, Object modelBean)
3         {
4             bm.bindValue(name, modelBean, PersonDataModel.P_Name);
5             bm.bindValue(firstName, modelBean, PersonDataModel.P_FirstName);
6             bm.bindValue(street, modelBean, PersonDataModel.P_Street);
7             bm.bindValue(streetNumber, modelBean, PersonDataModel.P_StreetNumber);
8             bm.bindValue(city, modelBean, PersonDataModel.P_City);
9             bm.bindValue(country, modelBean, PersonDataModel.P_Country);
10            bm.bindValue(birthdate, modelBean, PersonDataModel.P_Birthdate);
11
12        }
```

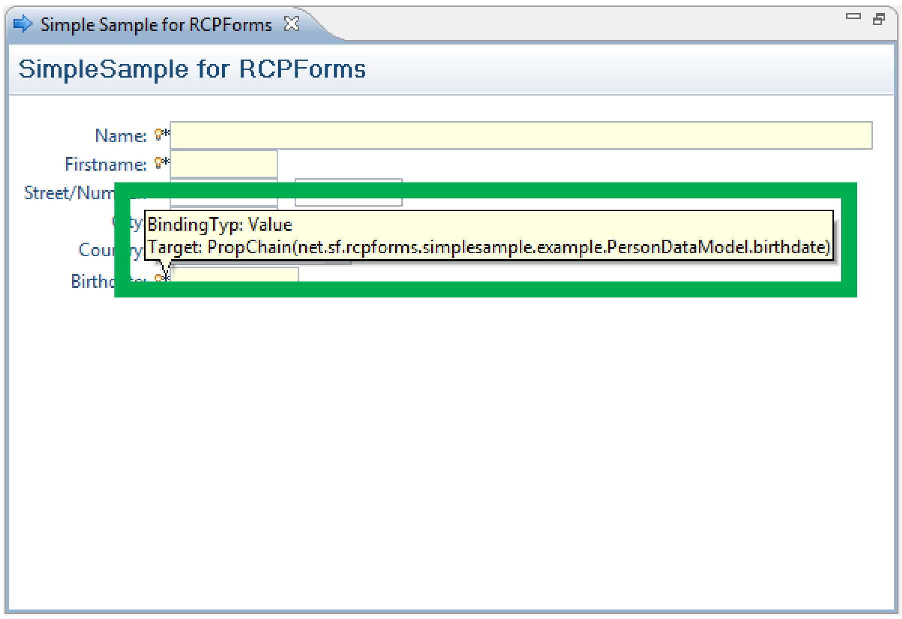
Listing 7 Bind view to model

What happened:
`[4-10]` binds the (rcp)widget to the specified property in the model (bean).

Special:
Nested properties support is integrated in RCPForms. See the sandbox sample how it works.

Hint: to get debugging information concerning the binding, add the parameter `-Dnet.sf.rcpforms.debug.bindings=true` to the environment. The tooltip you get contains information about binding type and binding target (see Figure 6 Debug message for binding).


VM arguments:
-Dnet.sf.rcpforms.debug.bindings=true



BindingTyp: Value
Target: PropChain(net.sf.rcpforms.simplesample.example.PersonDataModel.birthdate)

Figure 6 Debug message for binding

6. Add Validation

Setting the "`mandatory`" flag will not automatically check if values are set or even valid. "`Mandatory`" is only a visual representation for the required-state. To get this functionality the following chapters.
.

6.1 Required Fields

```
1       @Override
2       public void bind(ValidationManager bm, Object modelBean)
3       {
4           [...]
5           //add required validator
6           bm.addValidator(this, new RequiredValidator(PersonDataModel.P_Name,
7           PersonDataModel.P_FirstName, PersonDataModel.P_Birthdate));
8
9       }
```

Listing 8 Add required fields to the validation

What happened:
`[6-7]` Adds a required field validator to the validation manager. The ⓘ-Tags should be displayed for all the given properties. Declaring the rcpwidget as "`mandatory`" will not enable automatically the ⓘ-Tag for that field.
See Figure 8 Initial state of the view to get an idea how it looks like.

Add following import statement to your sourcecode:
```
import net.sf.rcpforms.modeladapter.converter.RequiredValidator;
```

6.2 Date Validation

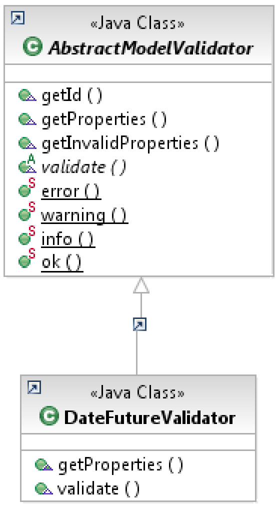


Figure 7 Hierarchy of the `DateFutureValidator`

```
1        [...]
2        /**
3        * Class DateFutureValidator is an example for a custom validator
4        *
5        * @author Remo Loetscher
6        */
7       public static final class DateFutureValidator extends
8   AbstractModelValidator
9       {
10          public Object[] getProperties()
11          {
12              return new String[]{PersonDataModel.P_Birthdate};
13          }
14
15          public IStatus validate(Object value)
16          {
17              PersonDataModel model = (PersonDataModel) value;
18              IStatus result = ok();
19              if (model.getBirthdate() != null && model.getBirthdate().after(new
20  Date()))
21              {
22                  result = error("Birthdate has to be in the past!");
23              }
24              return result;
25          }
26      }
27      [...]
```

Listing 9 Create custom validator

What happened:
`[7-26]` Definition of a custom validator. This "`DateFutureValidator`" is used to check that the birthdate is in past.
See Figure 9 RCPForm with an error to get an idea how it looks like.

Add following import statements:

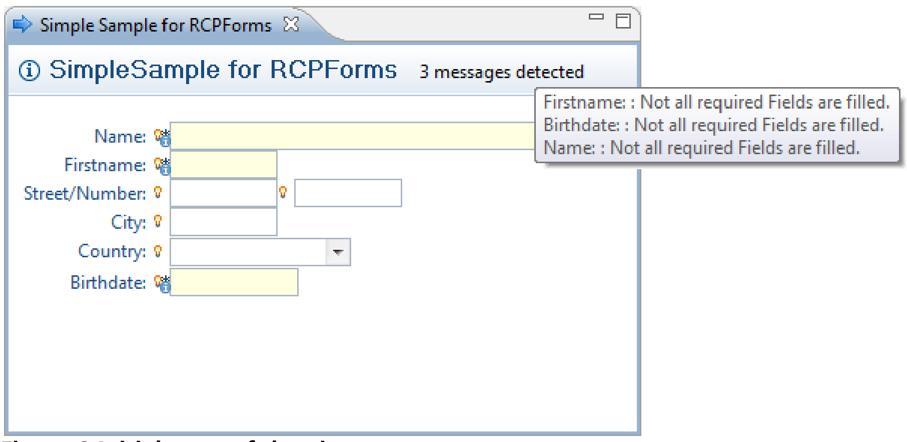
```
import net.sf.rcpforms.modeladapter.converter.AbstractModelValidator;
import org.eclipse.core.runtime.IStatus;
import java.util.Date;
```

```
1       @Override
2       public void bind(ValidationManager bm, Object modelBean)
3       {
4           [...]
5           //add date validator
6           bm.addValidator(this, new DateFutureValidator());
7
8       }
```

Listing 10 Add custom validator

What happened:
`[6]` Adds a custom validator (`DateFutureValidator`) to the validation manager.

7. Application

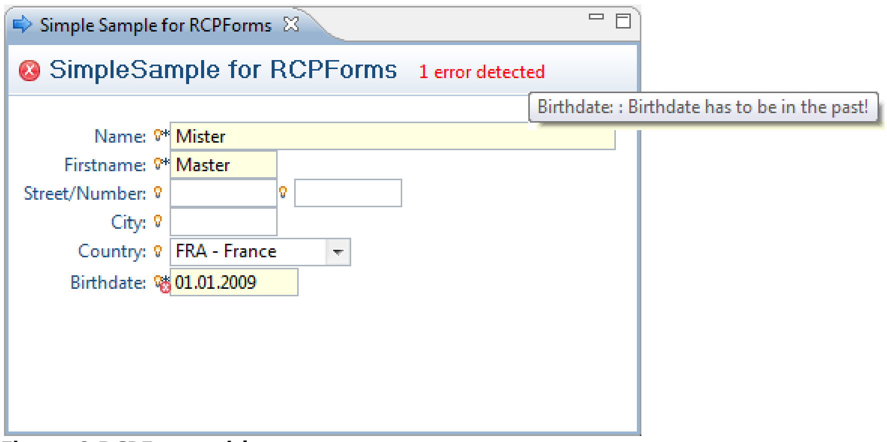

Simple Sample for RCPForms

SimpleSample for RCPForms 3 messages detected

Firstname: : Not all required Fields are filled.
Birthdate: : Not all required Fields are filled.
Name: : Not all required Fields are filled.

Name:	
Firstname:	
Street/Number:	
City:	
Country:	
Birthdate:	

Figure 8 Initial state of the view

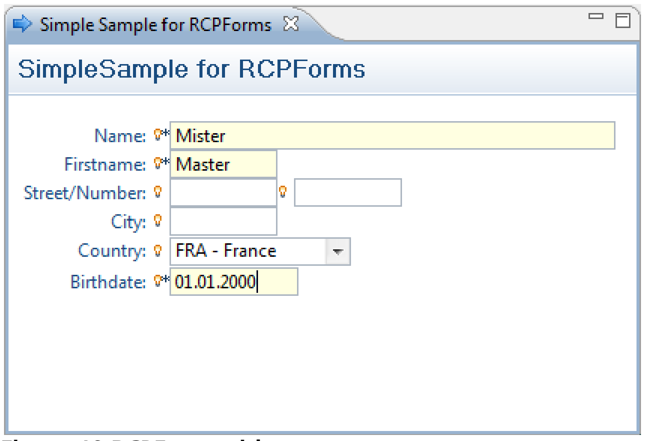

Simple Sample for RCPForms

SimpleSample for RCPForms 1 error detected

Birthdate: : Birthdate has to be in the past!

Name:	Mister
Firstname:	Master
Street/Number:	
City:	
Country:	FRA - France
Birthdate:	01.01.2009

Figure 9 RCPForm with an error


Simple Sample for RCPForms

SimpleSample for RCPForms

Name:	Mister
Firstname:	Master
Street/Number:	
City:	
Country:	FRA - France
Birthdate:	01.01.2000

Figure 10 RCPForm without errors

UI-Element	Description
"3 messages detected"	(Some) required fields were not filled
"1 error detected"	1 error occurred somewhere in the formpart
i-Tag (ⓘ, ⓘ)	(Some) required fields were not filled. Initially for all required fields on which a required validator is registered are marked this way.
x-Tag (⊗, ⊗)	Error state in the field and therefore formpart has also an error
yellow background (🟨)	Visual representation for a mandatory or recommended fields
star (*)	Decoration for mandatory fields
bulb (💡)	Debug Decoration for bindings. Tooltip explains binding type and model target. Enable it by using a jvm property: `-Dnet.sf.rcpforms.debug.bindings=true`

Table 1 Explanation some common ui elements

Appendix A: SimpleSample Template installation

Make sure to have RCPForms v 0.8 installed (either in workspace or target platform). For installation procedure please have a look at the RCPForms-Wiki.

The template is located in this plugin under "docs" folder.

To get the full functional simple sample you should add some code wherever this statement occurs:

```
// TODO add code here
```

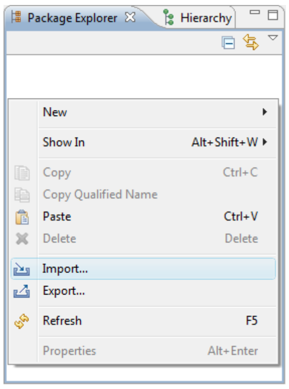


Figure 11 Eclipse import wizard

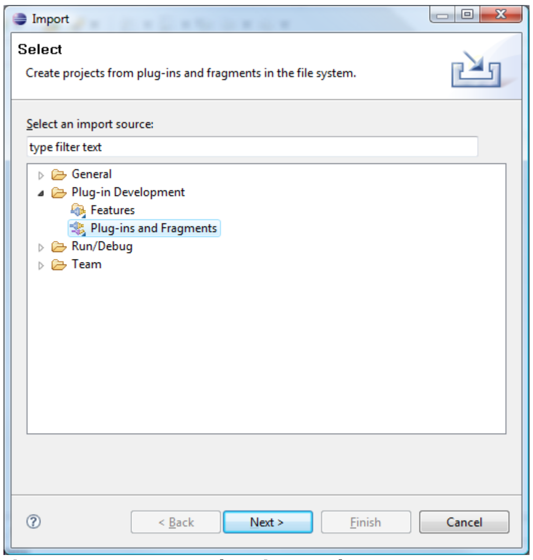
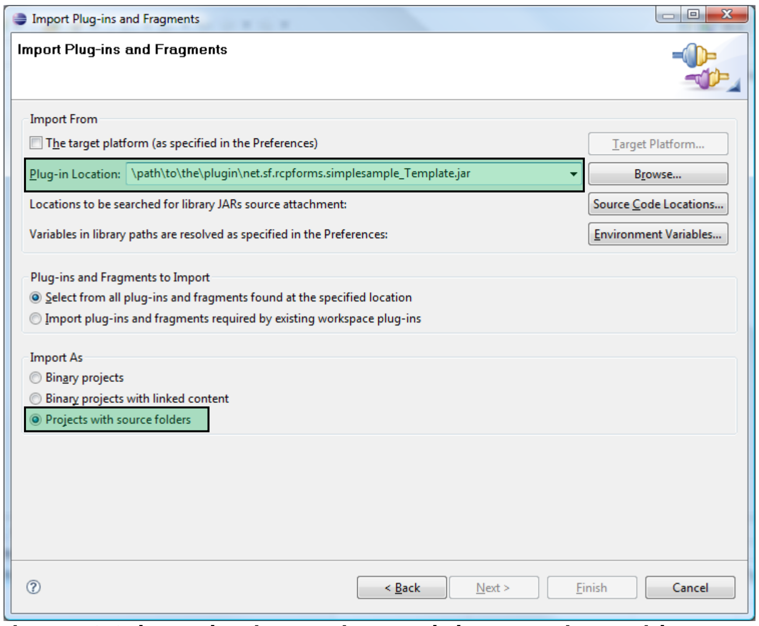


Figure 12 Import „Plug-ins and Fragments"



Import Plug-ins and Fragments

Import Plug-ins and Fragments

Import From

☐ The target platform (as specified in the Preferences)

Plug-in Location: \path\to\the\plugin\net.sf.rcpforms.simplesample_Template.jar

Locations to be searched for library JARs source attachment:

Variables in library paths are resolved as specified in the Preferences:

Plug-ins and Fragments to Import

● Select from all plug-ins and fragments found at the specified location

○ Import plug-ins and fragments required by existing workspace plug-ins

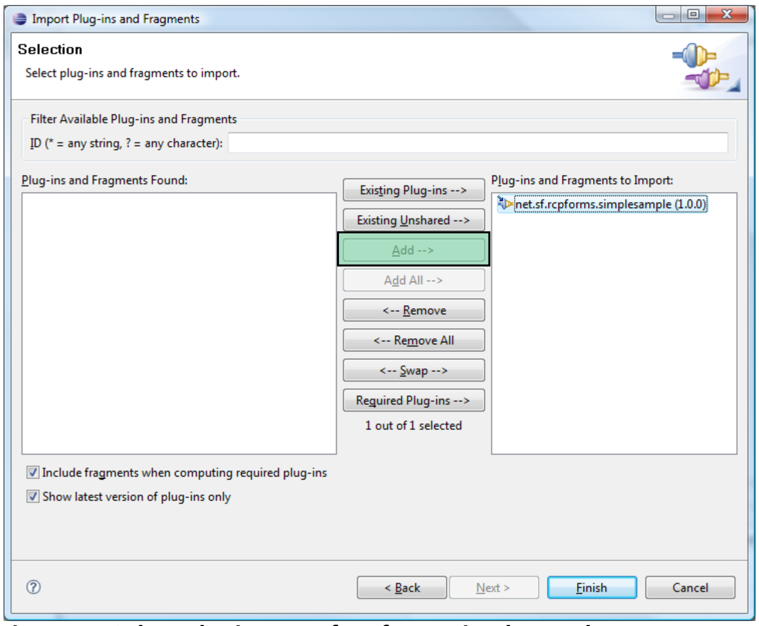
Import As

○ Binary projects

○ Binary projects with linked content

● Projects with source folders

Figure 13 Select „Plug-in Location“ and chose „Projects with source folders“



Import Plug-ins and Fragments

Selection

Select plug-ins and fragments to import.

Filter Available Plug-ins and Fragments

ID (* = any string, ? = any character):

Plug-ins and Fragments Found:

Plug-ins and Fragments to Import:

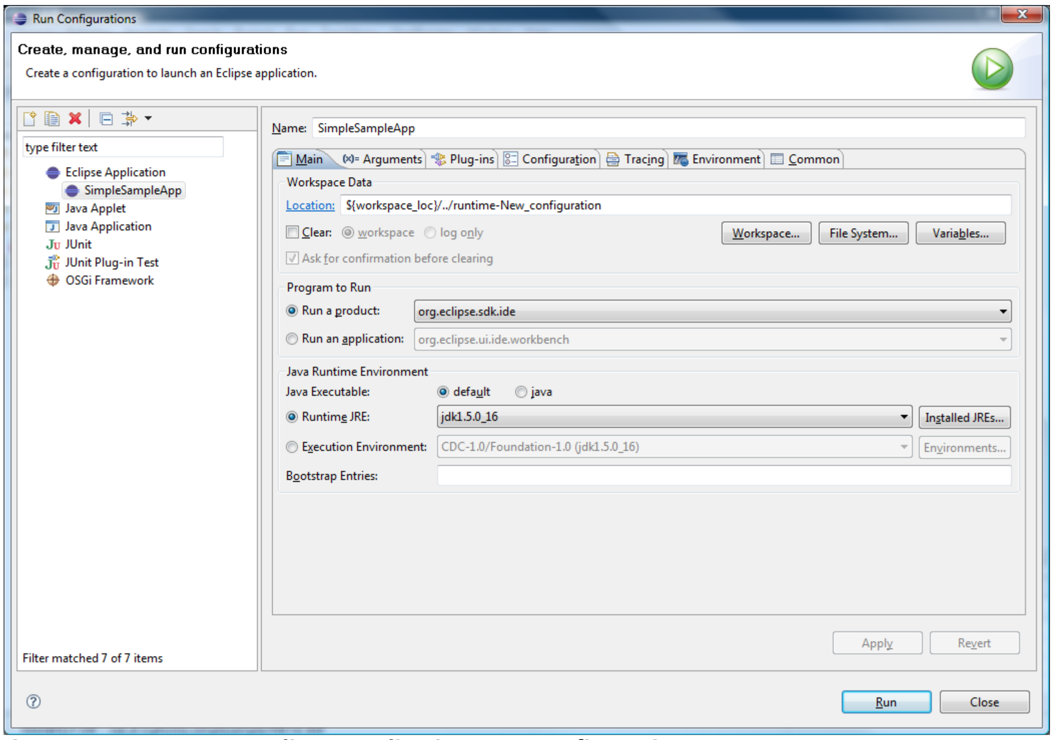
net.sf.rcpforms.simplesample (1.0.0)

1 out of 1 selected

☑ Include fragments when computing required plug-ins

☑ Show latest version of plug-ins only

Figure 14 Select plugin „net.sf.rcpforms.simplesample“


Run Configurations

Create, manage, and run configurations
Create a configuration to launch an Eclipse application.

type filter text
- Eclipse Application
  - SimpleSampleApp
- Java Applet
- Java Application
- JUnit
- JUnit Plug-in Test
- OSGi Framework

Filter matched 7 of 7 items

Name: SimpleSampleApp

Main | Arguments | Plug-ins | Configuration | Tracing | Environment | Common

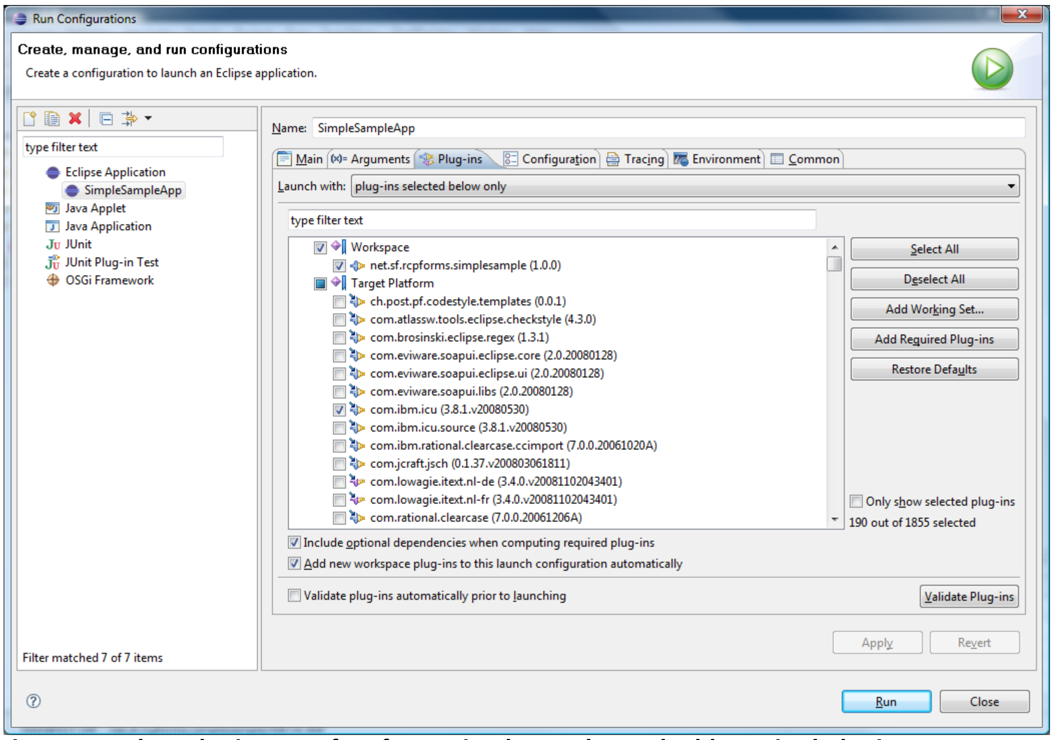
Workspace Data
Location: ${workspace_loc}/../runtime-New_configuration
☐ Clear: ● workspace ○ log only — Workspace... File System... Variables...
☑ Ask for confirmation before clearing

Program to Run
● Run a product: org.eclipse.sdk.ide
○ Run an application: org.eclipse.ui.ide.workbench

Java Runtime Environment
Java Executable: ● default ○ java
● Runtime JRE: jdk1.5.0_16 — Installed JREs...
○ Execution Environment: CDC-1.0/Foundation-1.0 (jdk1.5.0_16) — Environments...
Bootstrap Entries:

Apply Revert
Run Close

Figure 15 Create new "Eclipse Application" run configuration


Run Configurations

Create, manage, and run configurations
Create a configuration to launch an Eclipse application.

type filter text
- Eclipse Application
  - SimpleSampleApp
- Java Applet
- Java Application
- JUnit
- JUnit Plug-in Test
- OSGi Framework

Filter matched 7 of 7 items

Name: SimpleSampleApp

Main | Arguments | Plug-ins | Configuration | Tracing | Environment | Common

Launch with: plug-ins selected below only

type filter text
- ☑ Workspace
  - ☑ net.sf.rcpforms.simplesample (1.0.0)
- ▪ Target Platform
  - ☐ ch.post.pf.codestyle.templates (0.0.1)
  - ☐ com.atlassw.tools.eclipse.checkstyle (4.3.0)
  - ☐ com.brosinski.eclipse.regex (1.3.1)
  - ☐ com.eviware.soapui.eclipse.core (2.0.20080128)
  - ☐ com.eviware.soapui.eclipse.ui (2.0.20080128)
  - ☐ com.eviware.soapui.libs (2.0.20080128)
  - ☑ com.ibm.icu (3.8.1.v20080530)
  - ☐ com.ibm.icu.source (3.8.1.v20080530)
  - ☐ com.ibm.rational.clearcase.ccimport (7.0.0.20061020A)
  - ☐ com.jcraft.jsch (0.1.37.v200803061811)
  - ☐ com.lowagie.itext.nl-de (3.4.0.v20081102043401)
  - ☐ com.lowagie.itext.nl-fr (3.4.0.v20081102043401)
  - ☐ com.rational.clearcase (7.0.0.20061206A)

Select All | Deselect All | Add Working Set... | Add Required Plug-ins | Restore Defaults
☐ Only show selected plug-ins
190 out of 1855 selected

☑ Include optional dependencies when computing required plug-ins
☑ Add new workspace plug-ins to this launch configuration automatically
☐ Validate plug-ins automatically prior to launching — Validate Plug-ins

Apply Revert
Run Close

Figure 16 Select plugin „net.sf.rcpforms.simplesample" and add required plugins

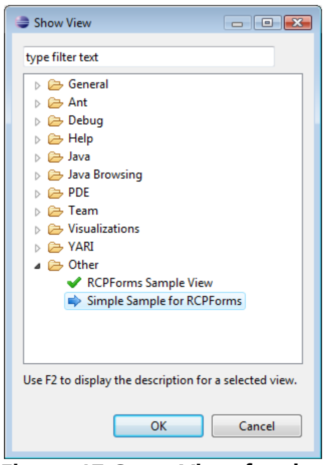
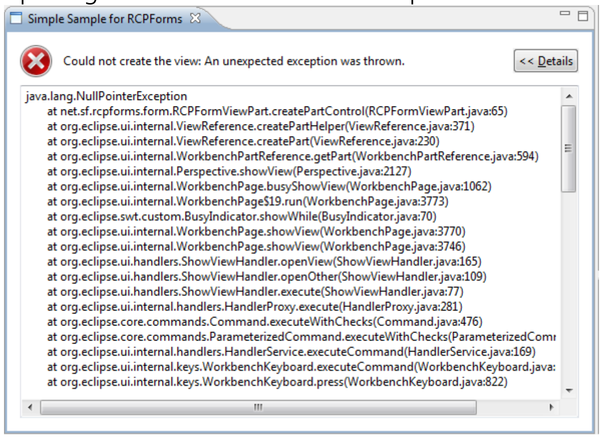


Figure 17 Open View for the Simple Sample

Opening the view based on the template will throw the following error:



Could not create the view: An unexpected exception was thrown.

```
java.lang.NullPointerException
    at net.sf.rcpforms.form.RCPFormViewPart.createPartControl(RCPFormViewPart.java:65)
    at org.eclipse.ui.internal.ViewReference.createPartHelper(ViewReference.java:371)
    at org.eclipse.ui.internal.ViewReference.createPart(ViewReference.java:230)
    at org.eclipse.ui.internal.WorkbenchPartReference.getPart(WorkbenchPartReference.java:594)
    at org.eclipse.ui.internal.Perspective.showView(Perspective.java:2127)
    at org.eclipse.ui.internal.WorkbenchPage.busyShowView(WorkbenchPage.java:1062)
    at org.eclipse.ui.internal.WorkbenchPage$19.run(WorkbenchPage.java:3773)
    at org.eclipse.swt.custom.BusyIndicator.showWhile(BusyIndicator.java:70)
    at org.eclipse.ui.internal.WorkbenchPage.showView(WorkbenchPage.java:3770)
    at org.eclipse.ui.internal.WorkbenchPage.showView(WorkbenchPage.java:3746)
    at org.eclipse.ui.handlers.ShowViewHandler.openView(ShowViewHandler.java:165)
    at org.eclipse.ui.handlers.ShowViewHandler.openOther(ShowViewHandler.java:109)
    at org.eclipse.ui.handlers.ShowViewHandler.execute(ShowViewHandler.java:77)
    at org.eclipse.ui.internal.handlers.HandlerProxy.execute(HandlerProxy.java:281)
    at org.eclipse.core.commands.Command.executeWithChecks(Command.java:476)
    at org.eclipse.core.commands.ParameterizedCommand.executeWithChecks(ParameterizedComı
    at org.eclipse.ui.internal.handlers.HandlerService.executeCommand(HandlerService.java:169)
    at org.eclipse.ui.internal.keys.WorkbenchKeyboard.executeCommand(WorkbenchKeyboard.java:
    at org.eclipse.ui.internal.keys.WorkbenchKeyboard.press(WorkbenchKeyboard.java:822)
```

Figure 18 Opening the template will cause an error

```
java.lang.NullPointerException
      at
net.sf.rcpforms.form.RCPFormViewPart.createPartControl(RCPFormViewPart.java
:65)
      at
org.eclipse.ui.internal.ViewReference.createPartHelper(ViewReference.java:3
71)
      at
org.eclipse.ui.internal.ViewReference.createPart(ViewReference.java:230)
      at
org.eclipse.ui.internal.WorkbenchPartReference.getPart(WorkbenchPartReferen
ce.java:594)
      at
   org.eclipse.ui.internal.Perspective.showView(Perspective.java:2127)
[...]
```

☞	Go ahaed with chapter 2 Create ViewPart to “fix” the error(s).

Appendix B: Architecture Overview

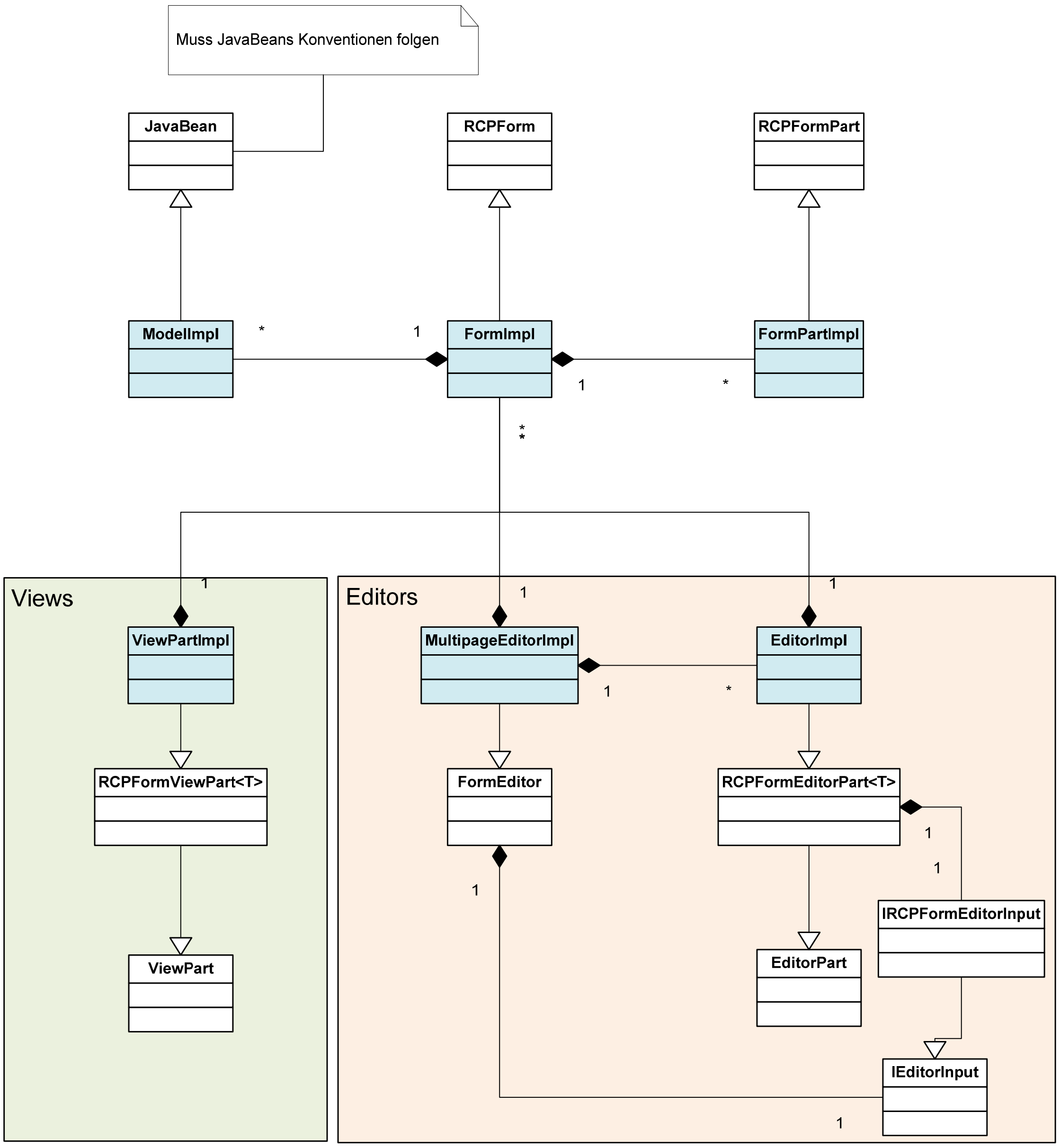


Figure 19 Architecture overview

Appendix C: Sequence Diagram

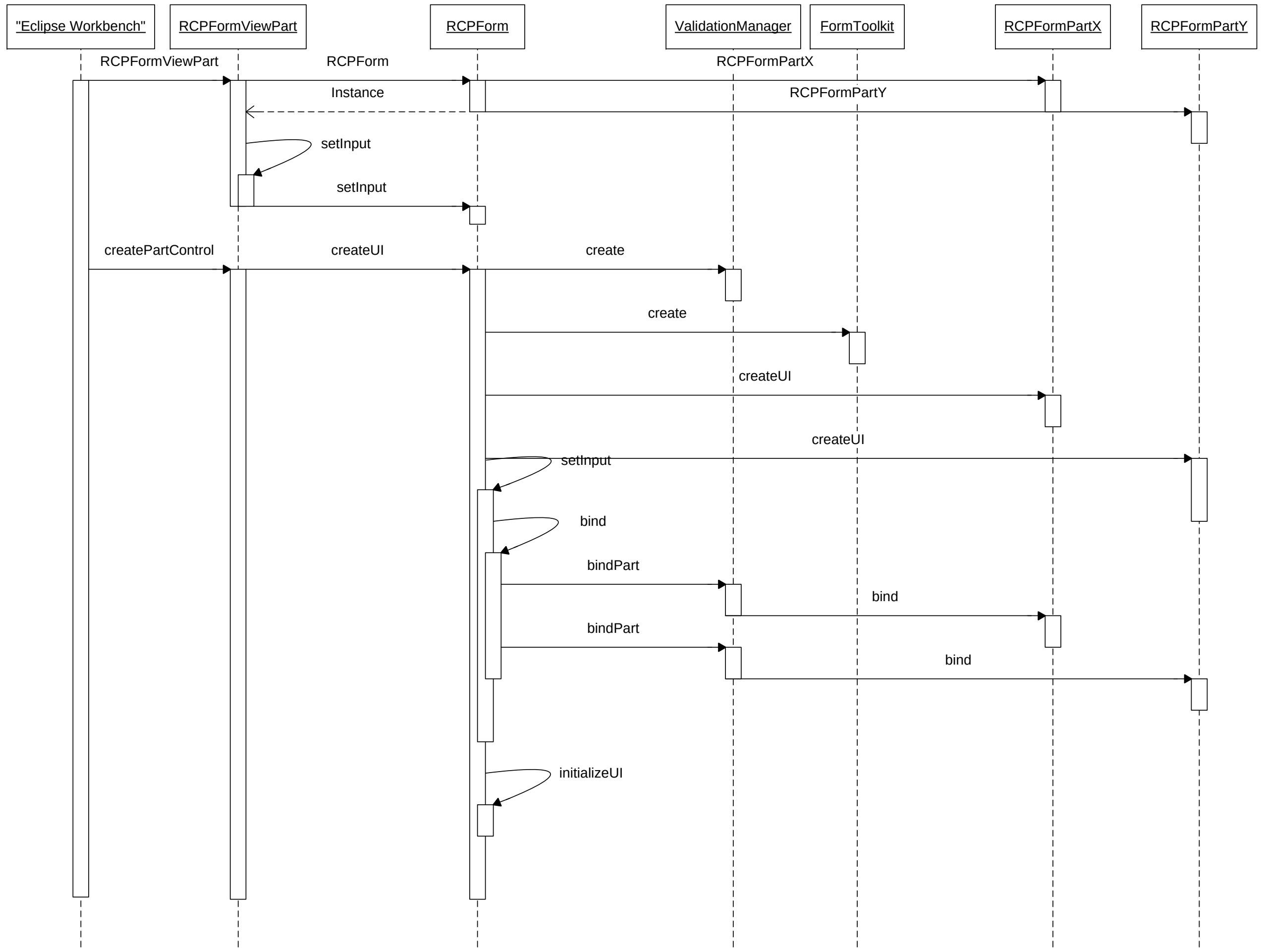


Figure 20 Sequence diagram for the creation of a RCPForm