Eclipse RCP Part XI

Automotive Financial Services Insurance Life Science & Healthcare Public Sector
Telecommunications & Media Travel & Logistics Utilities Automotive Financial
Services Insurance Life Science & Healthcare Public Sector Telecommunications
& Media Travel & Logistics Utilities Automotive Financial Services Insurance
Life Science & Healthcare Public Sector Telecommunications & Media Travel
& Logistics Utilities Automotive Financial Services Insurance Life Science
Healthcare Public Sector Telecommunications & Media Travel & Logistics
Utilities Automotive Financial Services Life Science & Healthcare Public
Sector Telecommunications & Media Travel & Logistics Utilities Automotive
Financial Services Insurance Life Science & Healthcare Public Sector
Telecommunications & Media Travel & Logistics Utilities Automotive
Financial Services Insurance Life Science & Healthcare Telecommunications
& Media Travel & Logistics Utilities Automotive Financial Services Insurance
Life Science & Healthcare Public Sector Telecommunications & Media Travel &
Logistics Utilities Automotive Financial Services Insurance Life Science &
Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities
Automotive Financial Services Insurance Life Science & Healthcare Public Sector

.consulting .solutions .partnership

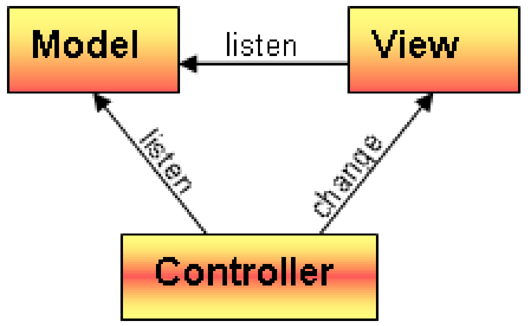
.msg systems

Objective

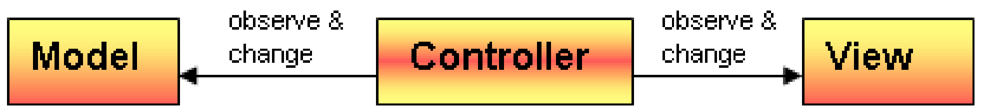
- JFace Data Binding

Eclipse Databinding

Traditional Model-View-Controller



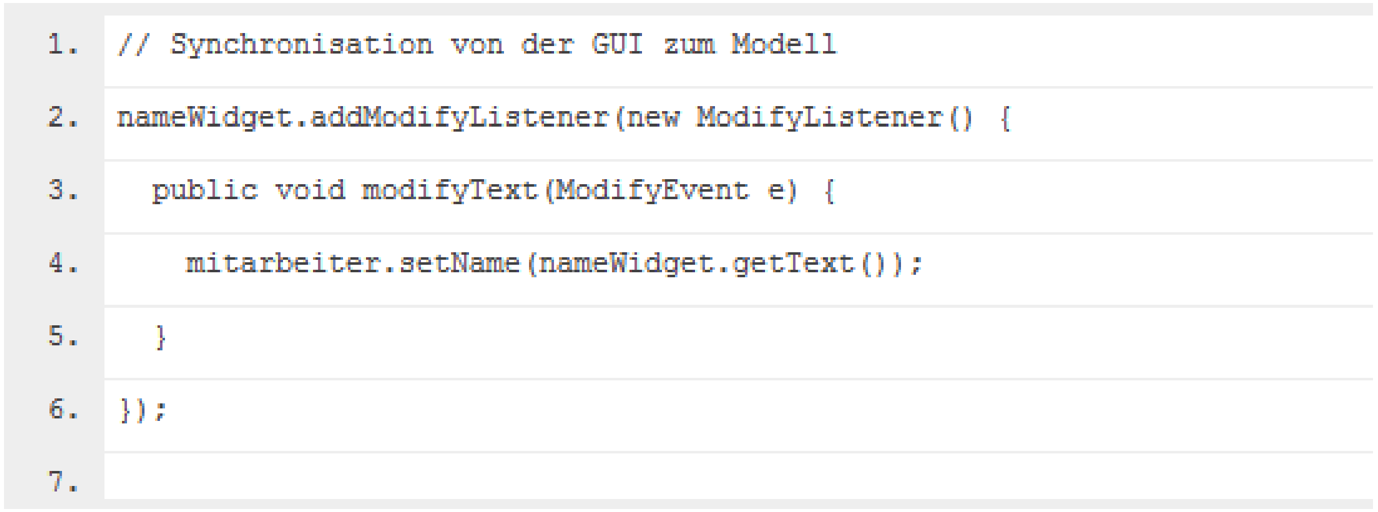
Eclipse DataBinding



Java Beans PropertyChange Support

```
public class Mitarbeiter  {
  String name = "Huber";
  private PropertyChangeSupport propertyChangeSupport =
    new PropertyChangeSupport(this);

  public void addPropertyChangeListener(String propertyName,
    PropertyChangeListener listener) {
  propertyChangeSupport.addPropertyChangeListener(propertyName,
   listener);
  }

  public String getName() {
    return name;
  }

  public void setName(String name) {
    propertyChangeSupport.firePropertyChange("name",
        this.name, this.name = name);
  }

}

```

Widget → Model

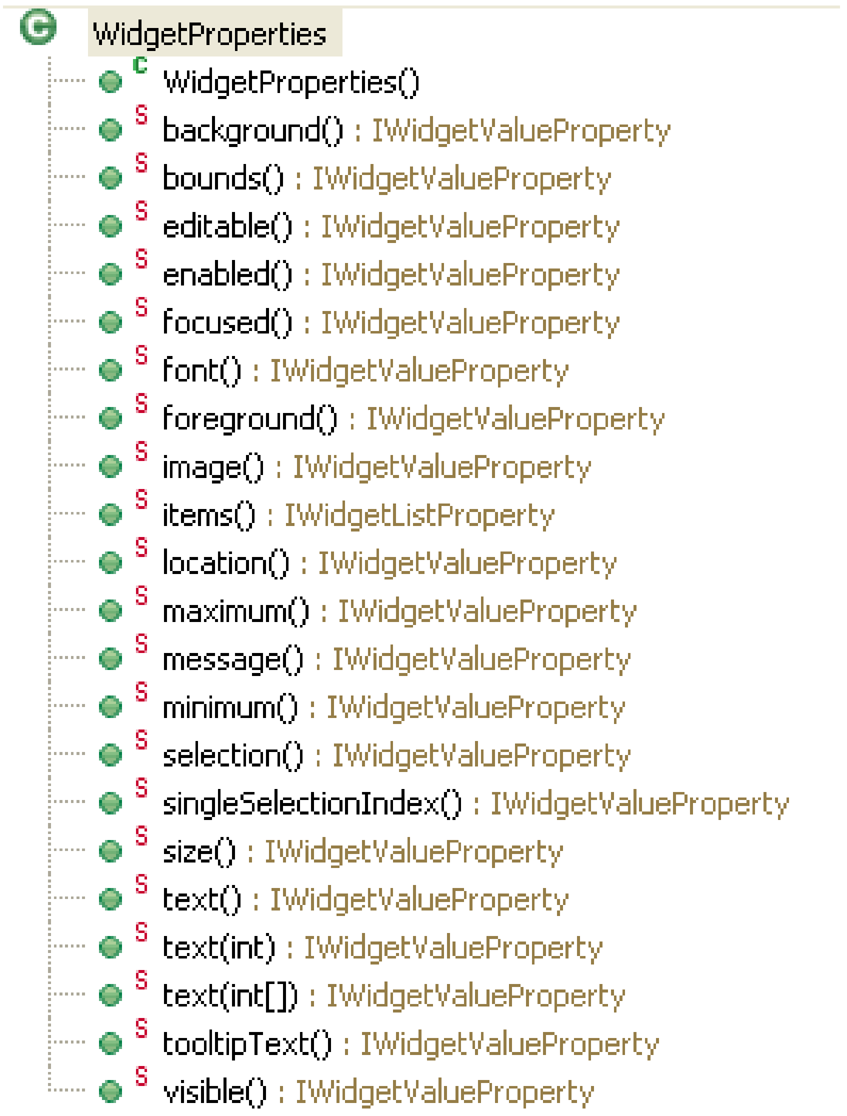


```java
// Synchronisation von der GUI zum Modell
nameWidget.addModifyListener(new ModifyListener() {
  public void modifyText(ModifyEvent e) {
    mitarbeiter.setName(nameWidget.getText());
  }
});

```

JFace Databinding

```
Text nameText = new Text(composite, SWT.None);
…
DataBindingContext bindingContext = new DataBindingContext();

IWidgetValueProperty textProperty =
   WidgetProperties.text(SWT.Modify);
IObservableValue textValue = textProperty.observe(nameText);

IValueProperty nameProperty = BeanProperties.value(“name”);
IObservableValue nameValue = nameProperty.observe(person);

bindingContext.bindValue(textProperty.observe(nameText),
   nameValue);
```

WidgetProperties

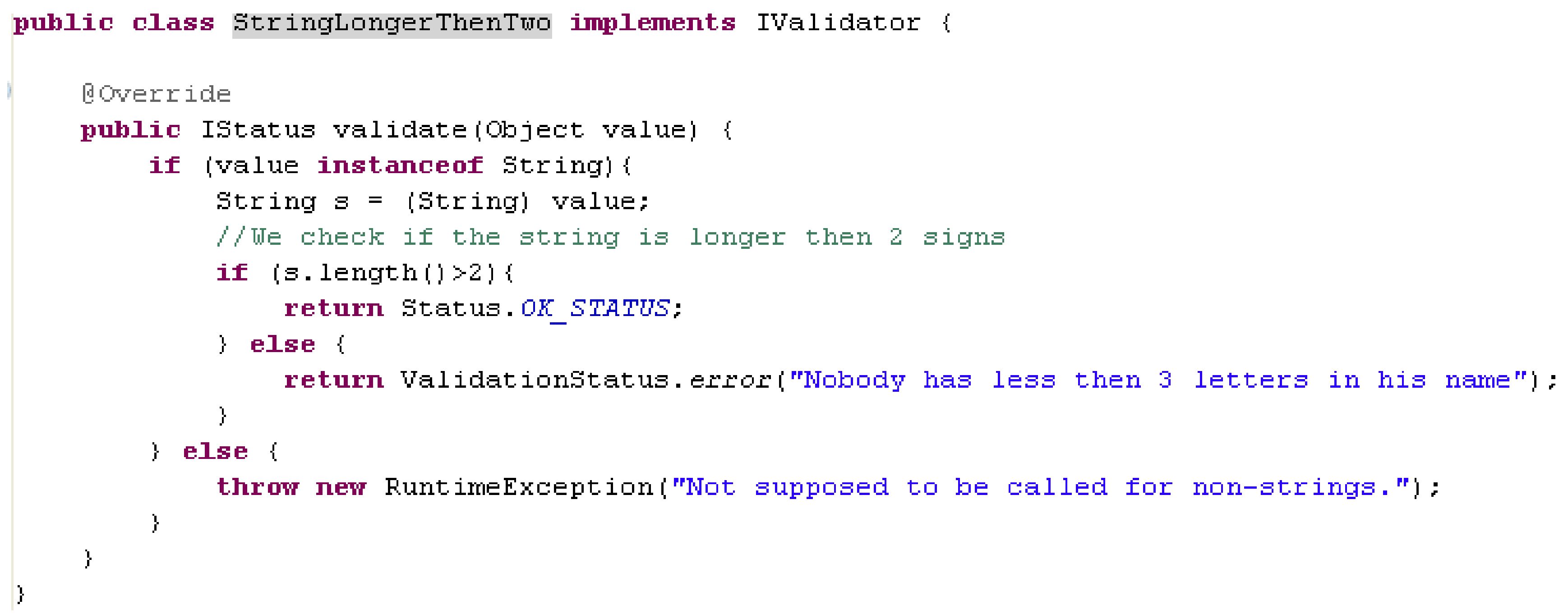


```
WidgetProperties
    WidgetProperties()
    background() : IWidgetValueProperty
    bounds() : IWidgetValueProperty
    editable() : IWidgetValueProperty
    enabled() : IWidgetValueProperty
    focused() : IWidgetValueProperty
    font() : IWidgetValueProperty
    foreground() : IWidgetValueProperty
    image() : IWidgetValueProperty
    items() : IWidgetListProperty
    location() : IWidgetValueProperty
    maximum() : IWidgetValueProperty
    message() : IWidgetValueProperty
    minimum() : IWidgetValueProperty
    selection() : IWidgetValueProperty
    singleSelectionIndex() : IWidgetValueProperty
    size() : IWidgetValueProperty
    text() : IWidgetValueProperty
    text(int) : IWidgetValueProperty
    text(int[]) : IWidgetValueProperty
    tooltipText() : IWidgetValueProperty
    visible() : IWidgetValueProperty
```

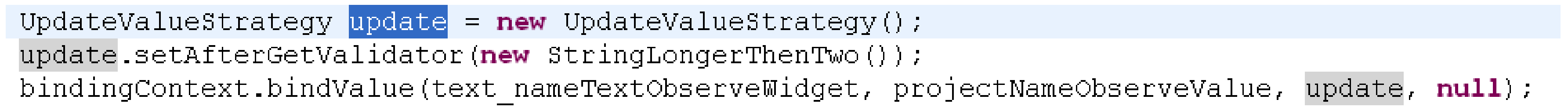
Customizing

- Customizes a Binding between two observable values. The following behaviors can be customized via the strategy:
- Validation:
  Validators validate the value in the update process.
- Conversion:
  A converter will convert the value from the type of the source observable into the type of the destination.

Sample Validator



```
public class StringLongerThenTwo implements IValidator {

    @Override
    public IStatus validate(Object value) {
        if (value instanceof String){
            String s = (String) value;
            //We check if the string is longer then 2 signs
            if (s.length()>2){
                return Status.OK_STATUS;
            } else {
                return ValidationStatus.error("Nobody has less then 3 letters in his name");
            }
        } else {
            throw new RuntimeException("Not supposed to be called for non-strings.");
        }
    }
}
```



```
UpdateValueStrategy update = new UpdateValueStrategy();
update.setAfterGetValidator(new StringLongerThenTwo());
bindingContext.bindValue(text_nameTextObserveWidget, projectNameObserveValue, update, null);
```

Lab

- Add a Text-Widget for both firstName and lastName to your editor
- Bind the widgets to the person referenced by the EditorInput-object using JFace Databinding
- When a change is made to the model return an appropriate value for isDirty() and call firePropertyChange(PROP_DIRTY)
- Add a Validator to the binding(s)

Vielen Dank für Ihre Aufmerksamkeit

.consulting .solutions .partnership