

EMF-Data binding

JAX 2010
Monday, May 3rd, 2010

EMF-Databinding

- **Founder and Owner of BestSolution.at**
- **Eclipse Committer**
 - EMF
 - e4
 - Platform UI
- **Projectlead**
 - Nebula
 - UFaceKit

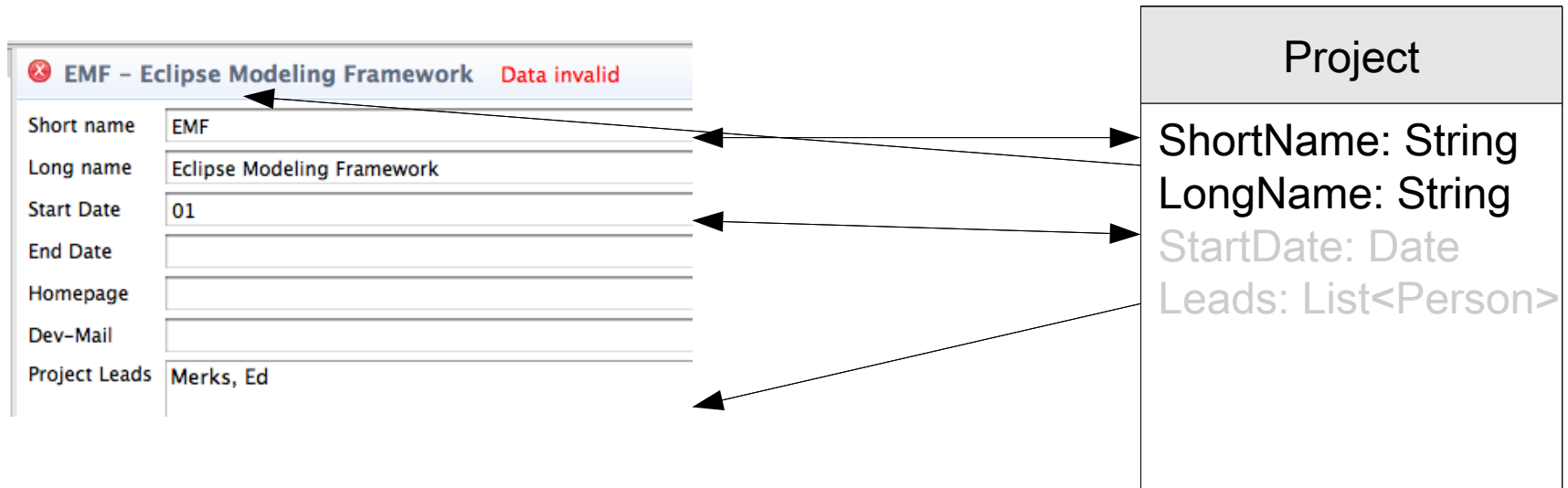
EMF-Databinding

- **Eclipse-Databinding**
 - Part of Eclipse SDK since 3.2
 - Major new API since 3.5
- **EMF-Databinding**
 - Available since EMF 2.4
 - Still marked as provisional though API and implementation are production ready

EMF-Databinding

■ What is Databinding

Synchronizing the attributes of 2 objects



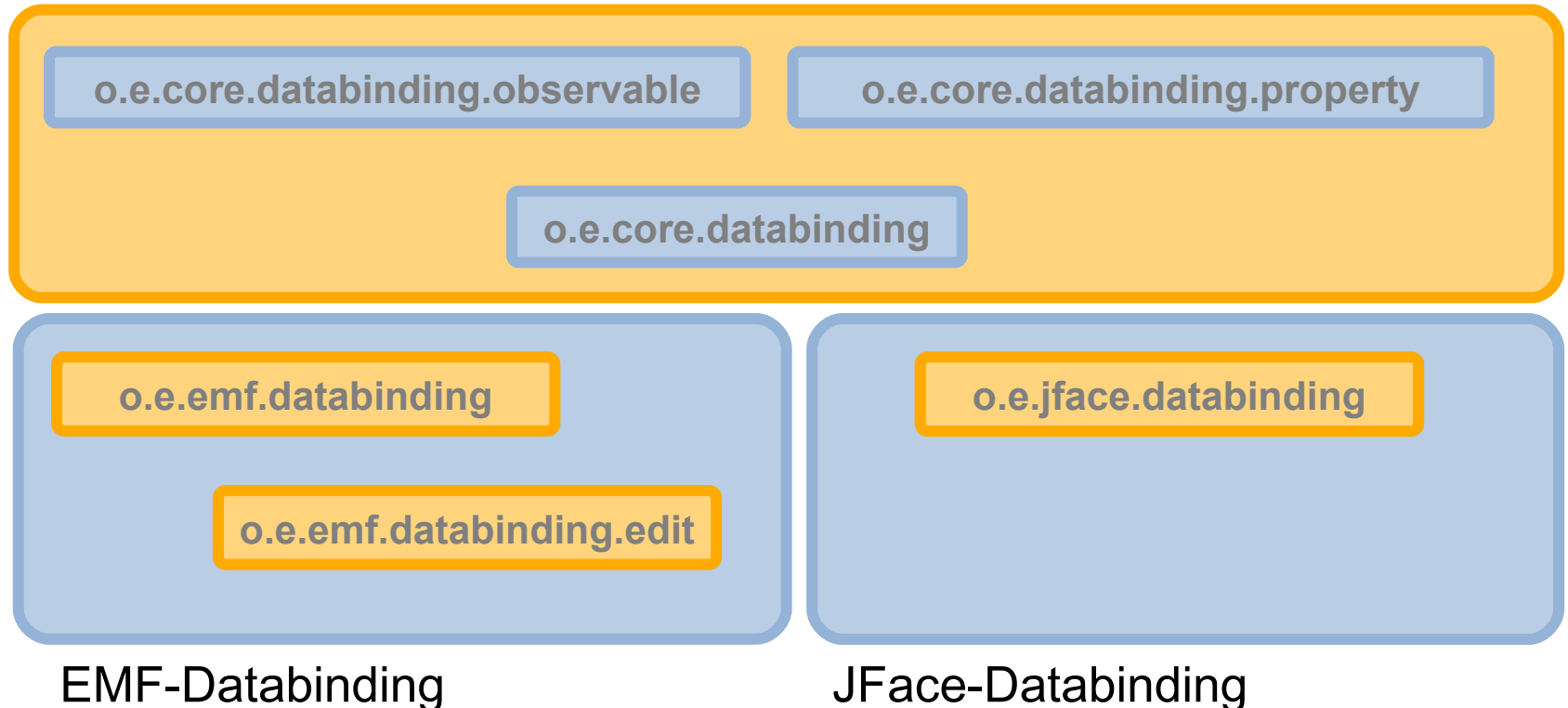
EMF-Databinding

- **Why use Databinding**
 - UI-Code not cluttered with Listeners
 - Automatic Thread syncing
 - Master-Detail
 - Conversion/Validation-Support
- **Why EMF-Databinding**
 - Advanced Features of EMF (e.g. Undo/Redo)

EMF-Databinding

Package Overview and Relation

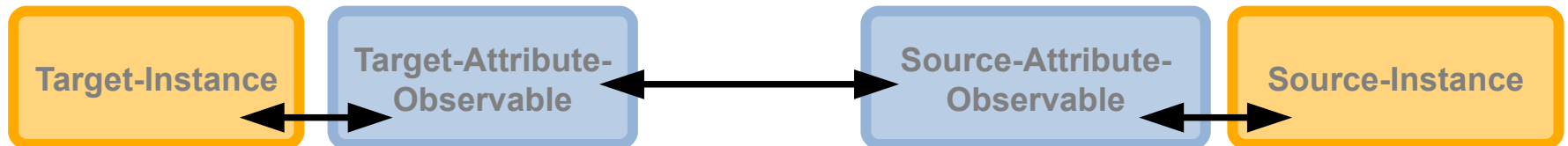
Eclipse-Databinding



EMF-Databinding

▪ Binding Attributes

binding properties is done by creating observers and bind them to each other

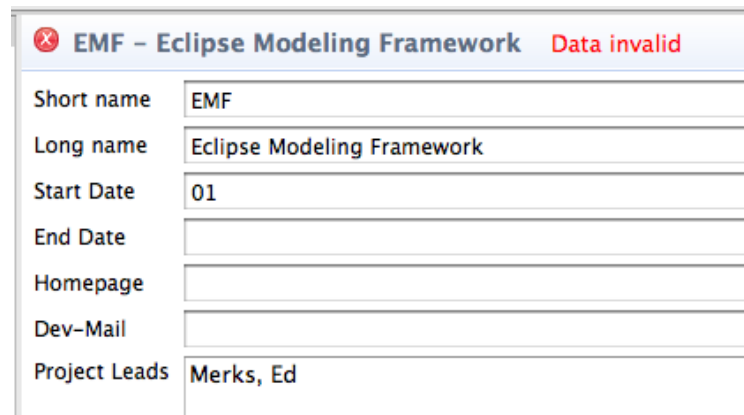


EMF-Databinding

- **Simple-Binding-Code**
 - Use Factories to create PropertyObservers
 - WidgetProperties for SWT/JFace-Observers
 - EMFProperties for EObject-Observers
 - Use DatabindingContext to connect

EMF-Databinding

■ Simple-Binding-Code



Short name	EMF
Long name	Eclipse Modeling Framework
Start Date	01
End Date	
Homepage	
Dev-Mail	
Project Leads	Merks, Ed

```
Project project = ...
```

```
// Create Target Observable
```

```
Text t = new Text(comp, SWT.BORDER);  
IValueProperty prop = WidgetProperties.text(SWT.Modify);  
IObservableValue tObs = prop.observe(t);
```

```
// Create Source Observable
```

```
IValueProperty shortProp = EMFProperties.value(ProjectPackage.Literals.PROJECT__SHORTNAME);  
IObservableValue sObs = shortProp.observe(project);
```

```
// Bind the Observables
```

```
EMFDataBindingContext ctx = new EMFDataBindingContext();  
ctx.bindValue(tObs, sObs);
```

EMF-Databinding

- **Simple-Binding-Customization**
 - Update policy (never, on-request, ...)
 - Custom Conversion between types
 - Custom validation

```
IObservableValue tObs = ...  
IObservableValue sObs = ...
```

```
EMFUpdateValueStrategy t2s = new EMFUpdateValueStrategy();  
StringToDateConverter c = new StringToDateConverter();  
t2s.setConverter(c);
```

```
EMFUpdateValueStrategy s2t = new EMFUpdateValueStrategy();  
DateToStringConverter c = new DateToStringConverter();  
s2t.setConverter(c);
```

```
EMFDataBindingContext ctx = new EMFDataBindingContext();  
ctx.bindValue(tObs, sObs, t2s, s2t);
```

EMF-Databinding

- **Structured-Control-Binding**
 - Wrap the control into a JFace-Viewer
 - Use utility classes coming with JFace-Databinding (Label and ContentProviders)
 - Pass in an Observable as input

EMF-Databinding

■ Structured-Control-Binding

```
// Create Properties
IListProperty mProp = EMFProperties.list(
    ProjectPackage.Literals.PROJECT__PROJECTLEADS);
IValueProperty fProp = EMFProperties.value(
    ProjectPackage.Literals.PERSON__FIRSTNAME);
IValueProperty lProp = EMFProperties.value(
    ProjectPackage.Literals.PERSON__LASTNAME);

// Create Observable
IObservableList input = mProp.observe(project);

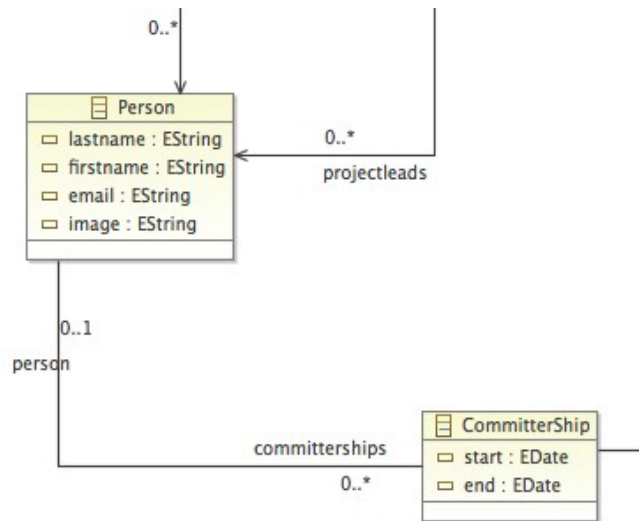
ObservableListContentProvider cp = new ObservableListContentProvider();
IObservableMap[] attributeMap = new IObservableMap[] {
    fProp.observeDetail(cp.getKnownElements()),
    lProp.observeDetail(cp.getKnownElements())
};

// Setup a TableViewer
TableViewer tv = new TableViewer(comp);
tv.setContentProvider(cp);
tv.setLabelProvider(new GenericMapCellLabelProvider("{0}, {1}", attributeMap));
tv.setInput(input);
```

EMF - Eclipse Modeling Framework Data invalid	
Short name	EMF
Long name	Eclipse Modeling Framework
Start Date	01
End Date	
Homepage	
Dev-Mail	
Project Leads	Merks, Ed

EMF-Databinding

Support for nested Attributes



```
CommitterShip committerShip = ...
```

```
// Create a Path to the feature
FeaturePath path = FeaturePath.fromList(
    ProjectPackage.Literals.COMMITTER_SHIP__PERSON,
    ProjectPackage.Literals.PERSON__LASTNAME)
);
```

```
IValueProperty fProp = EMFProperties.value(path);
IObservableValue mObs = fProp.observe(committerShip);
```

EMF-Databinding

■ Binding-Computed Values

```
Project project = ...
```

```
// Create Properties
```

```
IValueProperty sProp = EMFProperties.value(  
    ProjectPackage.Literals.PROJECT__SHORTNAME);  
IValueProperty lProp = EMFProperties.value(  
    ProjectPackage.Literals.PROJECT__LONGNAME);
```

```
// Create Computed Value
```

```
IObservableValue mObs == new ComputedValue()  
{  
    private IObservableValue shortname = sProp.observe(project);  
    private IObservableValue longname = lProp.observe(project);  
  
    @Override  
    protected Object calculate()  
    {  
        return shortname.getValue() + " - " + longname.getValue();  
    }  
};
```

The screenshot shows a dialog box titled "EMF - Eclipse Modeling Framework" with a red "Data invalid" status bar. The dialog contains several fields:

Short name	EMF
Long name	Eclipse Modeling Framework
Start Date	01
End Date	
Homepage	
Dev-Mail	
Project Leads	Merks, Ed

EMF-Databinding

- **Master-Detail**
 - Master Value must be an IObservableValue
 - Simply use observeDetail instead of observe

```
IObservableValue project = ...  
  
// Create Target Observable  
Text t ≡ new Text(comp, SWT.BORDER);  
IValueProperty prop ≡ WidgetProperties.text(SWT.Modify);  
IObservableValue tObs ≡ prop.observe(t);  
  
// Create Source Observable  
IValueProperty shortProp = EMFProperties.value(ProjectPackage.Literals.PROJECT__SHORTNAME);  
IObservableValue sObs ≡ shortProp.observeDetail(project);  
  
// Bind the Observables  
EMFDataBindingContext ctx = new EMFDataBindingContext();  
ctx.bindValue(tObs, sObs);
```

EMF-Databinding

- **EMF-Edit integration**
 - Use `EMFEditProperties` and changes go through the `CommandStack`

```
Project project = ...
```

```
// Create Target Observable
```

```
Text t = new Text(comp, SWT.BORDER);
```

```
IWidgetValueProperty prop = WidgetProperties.text(SWT.Modify);
```

```
IObservableValue tObs = prop.observeDelayed(400, t);
```

```
// Create Source Observable
```

```
IValueProperty sProp = EMFEditProperties.value(eDom, ProjectPackage.Literals.PROJECT__SHORTNAME);
```

```
IObservableValue sObs = sProp.observe(project);
```

```
// Bind the Observables
```

```
EMFDataBindingContext ctx = new EMFDataBindingContext();
```

```
ctx.bindValue(tObs, sObs);
```

EMF-Databinding

- **Supported Technologies**
 - Domain-Model
 - JavaBean, POJO – Eclipse Core Databinding
 - EObject – EMF Databinding
 - UBean – Eclipse UFaceKit
 - UI
 - SWT/JFace – Core Databinding
 - Swing – Eclipse UFaceKit
 - Qt – Eclipse UFaceKit
 - GWT – Eclipse UFaceKit
 - QxWT – UFaceKit.org
 - Android – UFaceKit.org (no code available in repository yet)

EMF-Databinding

■ Resources

- Personal Blog <http://tomsondev.bestsolution.at/>
- Eclipse Wiki
http://wiki.eclipse.org/JFace_Data_Binding
- The RCP Book <http://eclipsercp.org/>

THE END