

# EMF Databinding

Tom Schindl - BestSolution Systemhaus GmbH

---

*JAX 2011 - May 2nd 2011*

(c) Tom Schindl - BestSolution Systemhaus GmbH - Licensed under Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 Germany License.

# About Me

---

- ✧ CEO BestSolution Systemhaus GmbH
- ✧ Eclipse Committer
  - ✧ e4
  - ✧ Platform UI
  - ✧ EMF
- ✧ Projectlead: UFaceKit, Nebula
- ✧ Member of the Architectural Council



# What is databinding

- ❖ Synchronize attributes of 2 objects
  - ❖ Most of the time used to keep UI and Domain Model in sync

The screenshot shows a Java Swing window with two forms. The top form, titled 'Private', has fields for 'Firstname' (Ed), 'Lastname' (Merks), 'Street' (Mainstreet 10), 'ZIP/City' (9866 / Vancouver), 'Country' (Canada), and 'State' (British Columbia). The 'State' field is a dropdown menu with a list of Canadian provinces and territories. The bottom form, titled 'Business', has fields for 'Street', 'ZIP/City', 'Country', and 'State'. The 'Business' form is currently disabled, indicated by a greyed-out appearance.

Name	Value
▶ ● this	PersonForm (id=100)
▼ ○ person	PersonImpl (id=104)
▶ ◆ addresses	EObjectContainmentWithInverseEList<E> (id=126)
▶ ◆ eAdapters	BasicNotifierImpl\$EAdapterList<E> (id=148)
▶ ◆ eContainer	AddressBookImpl (id=151)
◆ eContainerFeatureID	3
◆ eFlags	1
▶ ◆ eProperties	BasicEObjectImpl\$PropertiesHolderImpl (id=154)
◆ firstname	"Ed" (id=159)
◆ lastname	"Merks" (id=163)

(c) Tom Schindl - BestSolution Systemhaus GmbH - Licensed under Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 Germany License.

# What is databinding

- ❖ Synchronize attributes of 2 objects
  - ❖ Most of the time used to keep UI and Domain Model in sync

Firstname: Ed  
Lastname: Merks  
Private:  
Street: Mainstreet 10  
ZIP/City: 9866 / Vancouver  
Country: Canada  
State: British Columbia  
Yukon  
Northwest Territories  
Nunavut  
Alberta  
Saskatchewan  
Manitoba  
Ontario  
Quebec  
New Brunswick  
Nova Scotia  
Prince Edward Island  
Newfoundland and Labrador  
Business:  
Street:  
ZIP/City:  
Country:  
State:

Name	Value
▶ this	PersonForm (id=100)
▼ person	PersonImpl (id=104)
▶ addresses	EObjectContainmentWithInverseEList<E> (id=126)
▶ eAdapters	BasicNotifierImpl\$EAdapterList<E> (id=148)
▶ eContainer	AddressBookImpl (id=151)
eContainerFeatureID	3
eFlags	1
▶ eProperties	BasicEObjectImpl\$PropertiesHolderImpl (id=154)
firstname	"Ed" (id=159)
lastname	"Merks" (id=163)



# What is databinding

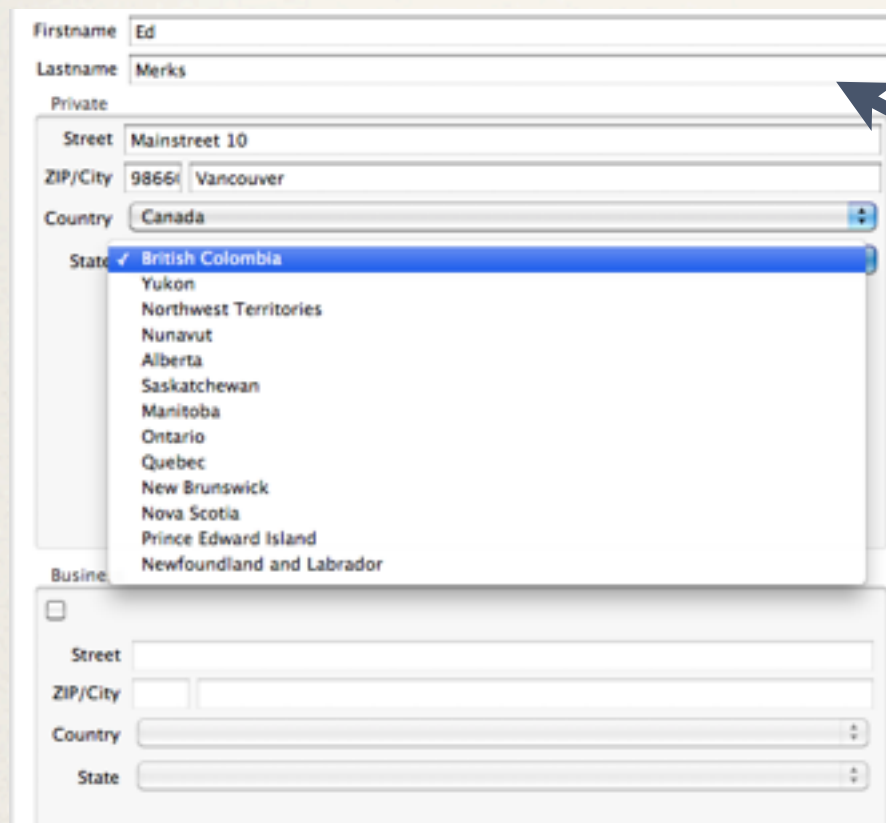
- ❖ Synchronize attributes of 2 objects
  - ❖ Most of the time used to keep UI and Domain Model in sync

Firstname: Ed  
Lastname: Merks  
Private: ☐  
Street: Mainstreet 10  
ZIP/City: 98661 Vancouver  
Country: Canada  
State: British Columbia  
Yukon  
Northwest Territories  
Nunavut  
Alberta  
Saskatchewan  
Manitoba  
Ontario  
Quebec  
New Brunswick  
Nova Scotia  
Prince Edward Island  
Newfoundland and Labrador  
Business: ☐  
Street:   
ZIP/City:   
Country:   
State:

Name	Value
▶ this	PersonForm (id=100)
▼ person	PersonImpl (id=104)
▶ addresses	EObjectContainmentWithInverseEList<E> (id=126)
▶ eAdapters	BasicNotifierImpl\$EAdapterList<E> (id=148)
▶ eContainer	AddressBookImpl (id=151)
eContainerFeatureID	3
eFlags	1
▶ eProperties	BasicEObjectImpl\$PropertiesHolderImpl (id=154)
firstname	"Ed" (id=159)
lastname	"Merks" (id=163)

# What is databinding

- ❖ Synchronize attributes of 2 objects
  - ❖ Most of the time used to keep UI and Domain Model in sync



A screenshot of a Java Swing form. The form has fields for Firstname (Ed), Lastname (Merks), Private (checkbox), Street (Mainstreet 10), ZIP/City (98661 Vancouver), Country (Canada), and State (British Columbia). Below these is a Business section with a checkbox and fields for Street, ZIP/City, Country, and State. A blue arrow points from the 'person' object in the domain model to the 'State' field in the form, indicating a databinding relationship.

Name	Value
▶ ● this	PersonForm (id=100)
▼ ○ person	PersonImpl (id=104)
▶ ◆ addresses	EObjectContainmentWithInverseEList<E> (id=126)
▶ ◆ eAdapters	BasicNotifierImpl\$EAdapterList<E> (id=148)
▶ ◆ eContainer	AddressBookImpl (id=151)
◆ eContainerFeatureID	3
◆ eFlags	1
▶ ◆ eProperties	BasicEObjectImpl\$PropertiesHolderImpl (id=154)
▶ ◆ firstname	"Ed" (id=159)
▶ ◆ lastname	"Merks" (id=163)



# What is databinding

- ❖ Synchronize attributes of 2 objects
  - ❖ Most of the time used to keep UI and Domain Model in sync

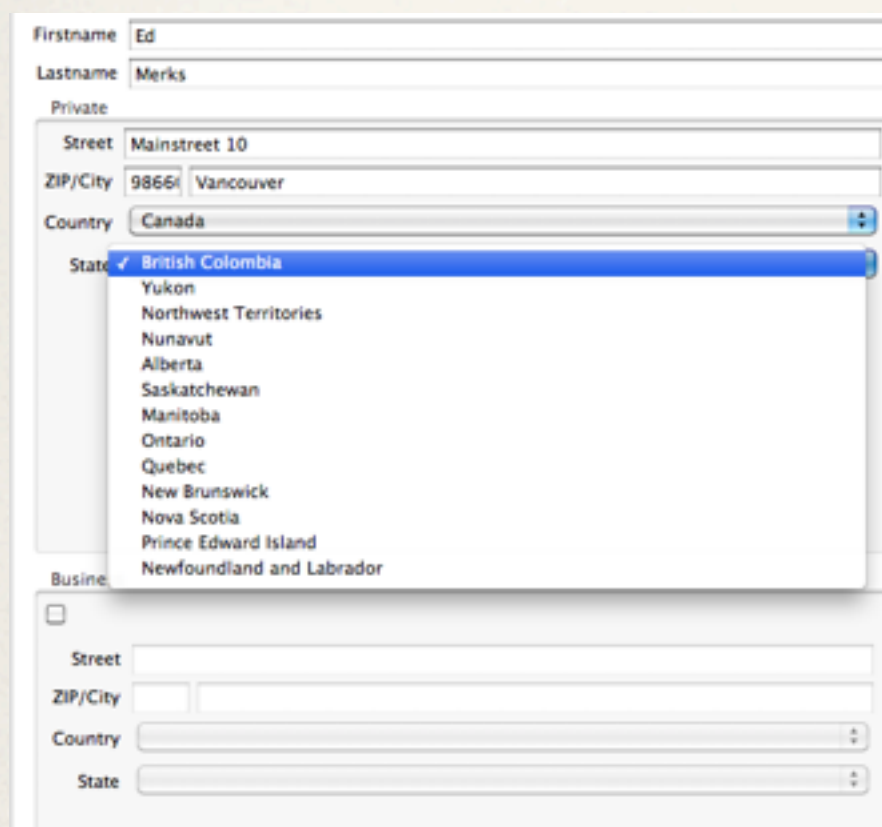
The screenshot shows a Java Swing window with two forms. The top form is a 'PersonForm' with fields for Firstname (Ed), Lastname (Merks), Private (checkbox), Street (Mainstreet 10), ZIP/City (9866 / Vancouver), Country (Canada), and State (British Columbia). The bottom form is a 'PersonImpl' with fields for Street, ZIP/City, Country, and State. The State dropdown menu is open, showing a list of Canadian provinces and territories. The form is demonstrating data binding between the UI and the domain model.

Name	Value
▶ ● this	PersonForm (id=100)
▼ ○ person	PersonImpl (id=104)
▶ ◆ addresses	EObjectContainmentWithInverseEList<E> (id=126)
▶ ◆ eAdapters	BasicNotifierImpl\$EAdapterList<E> (id=148)
▶ ◆ eContainer	AddressBookImpl (id=151)
◆ eContainerFeatureID	3
◆ eFlags	1
▶ ◆ eProperties	BasicEObjectImpl\$EPropertiesHolderImpl (id=154)
◆ firstname	"Ed" (id=159)
◆ lastname	"Merks" (id=163)

(c) Tom Schindl - BestSolution Systemhaus GmbH - Licensed under Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 Germany License.

# What is databinding

- ❖ Synchronize attributes of 2 objects
  - ❖ Most of the time used to keep UI and Domain Model in sync



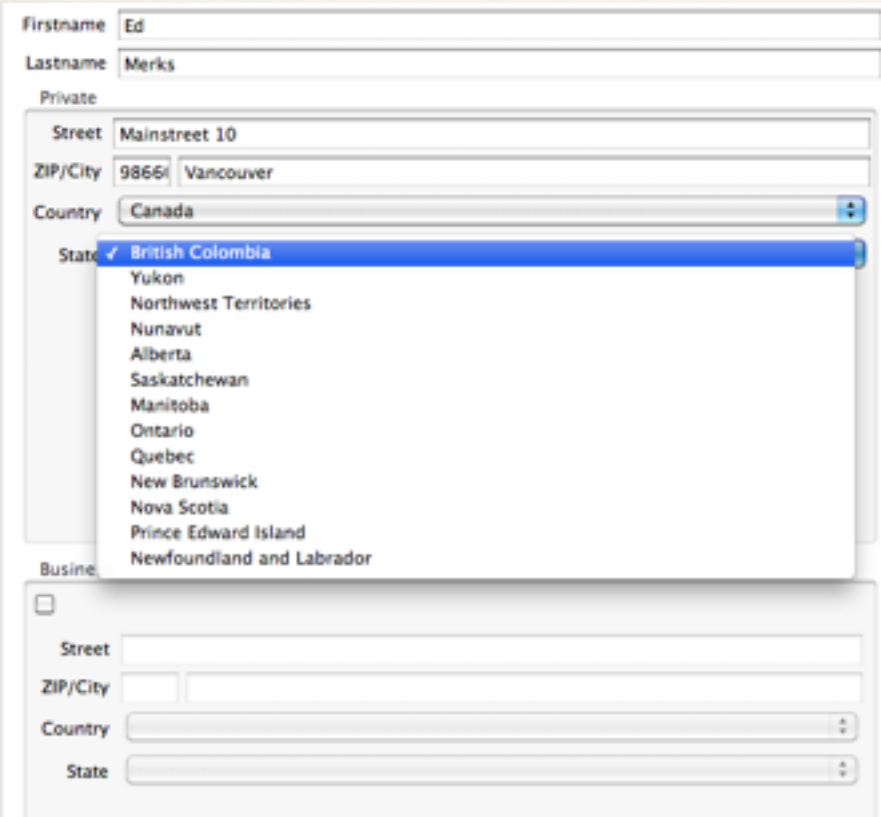
A screenshot of a Java Swing form. The form has several text fields: "Firstname" (containing "Ed"), "Lastname" (containing "Merks"), "Street" (containing "Mainstreet 10"), "ZIP/City" (containing "9866" and "Vancouver"), "Country" (containing "Canada"), and "State" (containing "British Colombia"). A blue arrow points from the word "text" to the "Firstname" text field. Below the "State" field, there is a list of Canadian provinces and territories, with "British Colombia" selected. At the bottom, there is a "Business" section with a checkbox and more text fields for "Street", "ZIP/City", "Country", and "State".

Name	Value
▶ ● this	PersonForm (id=100)
▼ ○ person	PersonImpl (id=104)
▶ ◆ addresses	EObjectContainmentWithInverseEList<E> (id=126)
▶ ◆ eAdapters	BasicNotifierImpl\$EAdapterList<E> (id=148)
▶ ◆ eContainer	AddressBookImpl (id=151)
◆ eContainerFeatureID	3
◆ eFlags	1
▶ ◆ eProperties	BasicEObjectImpl\$PropertiesHolderImpl (id=154)
◆ firstname	"Ed" (id=159)
◆ lastname	"Merks" (id=163)



# What is databinding

- ❖ Synchronize attributes of 2 objects
  - ❖ Most of the time used to keep UI and Domain Model in sync



A screenshot of a Java Swing form. The form has several text input fields: 'Firstname' (containing 'Ed'), 'Lastname' (containing 'Merks'), 'Street' (containing 'Mainstreet 10'), 'ZIP/City' (containing '9866' and 'Vancouver'), and 'Country' (containing 'Canada'). There is also a 'State' dropdown menu with 'British Colombia' selected. Below these is a 'Business' section with a checkbox and more text fields for 'Street', 'ZIP/City', 'Country', and 'State'. An arrow labeled 'text' points to the 'Firstname' field, and an arrow labeled 'selection' points to the 'State' dropdown menu.

Name	Value
▶ ● this	PersonForm (id=100)
▼ ○ person	PersonImpl (id=104)
▶ ◆ addresses	EObjectContainmentWithInverseEList<E> (id=126)
▶ ◆ eAdapters	BasicNotifierImpl\$EAdapterList<E> (id=148)
▶ ◆ eContainer	AddressBookImpl (id=151)
◆ eContainerFeatureID	3
◆ eFlags	1
▶ ◆ eProperties	BasicEObjectImpl\$PropertiesHolderImpl (id=154)
◆ firstname	"Ed" (id=159)
◆ lastname	"Merks" (id=163)

# What is databinding

- ❖ Synchronize attributes of 2 objects
  - ❖ Most of the time used to keep UI and Domain Model in sync

A screenshot of a Java Swing form. The form has several text input fields: 'Firstname' (containing 'Ed'), 'Lastname' (containing 'Merks'), 'Street' (containing 'Mainstreet 10'), 'ZIP/City' (containing '9866' and 'Vancouver'), and 'Country' (containing 'Canada'). There is a selection field for 'State' with a dropdown menu open, showing a list of Canadian provinces and territories, with 'British Colombia' selected. Below this is a 'Business' section with a checkbox and more text input fields for 'Street', 'ZIP/City', 'Country', and 'State'. Arrows point from the labels 'text', 'selection', and 'content' to the respective parts of the form.

Name	Value
▶ ● this	PersonForm (id=100)
▼ ○ person	PersonImpl (id=104)
▶ ◆ addresses	EObjectContainmentWithInverseEList<E> (id=126)
▶ ◆ eAdapters	BasicNotifierImpl\$EAdapterList<E> (id=148)
▶ ◆ eContainer	AddressBookImpl (id=151)
◆ eContainerFeatureID	3
◆ eFlags	1
▶ ◆ eProperties	BasicEObjectImpl\$PropertiesHolderImpl (id=154)
◆ firstname	"Ed" (id=159)
◆ lastname	"Merks" (id=163)



# What is databinding

- ❖ Synchronize attributes of 2 objects
  - ❖ Most of the time used to keep UI and Domain Model in sync

A screenshot of a Java Swing form with several text fields and a dropdown menu. Annotations with arrows point to specific parts of the form:

- text**: Points to the 'Firstname' field containing 'Ed'.
- selection**: Points to the 'Country' dropdown menu showing 'Canada'.
- content**: Points to the 'State' dropdown menu showing a list of Canadian provinces and territories, with 'British Colombia' selected.
- enabled**: Points to the 'Street' field in a disabled section at the bottom of the form.

Name	Value
▶ ● this	PersonForm (id=100)
▼ ○ person	PersonImpl (id=104)
▶ ◆ addresses	EObjectContainmentWithInverseEList<E> (id=126)
▶ ◆ eAdapters	BasicNotifierImpl\$EAdapterList<E> (id=148)
▶ ◆ eContainer	AddressBookImpl (id=151)
◆ eContainerFeatureID	3
◆ eFlags	1
▶ ◆ eProperties	BasicEObjectImpl\$EPropertiesHolderImpl (id=154)
◆ firstname	"Ed" (id=159)
◆ lastname	"Merks" (id=163)

# Basic Design of Eclipse DB

---

- ✦ Eclipse Databinding is built around the observer pattern

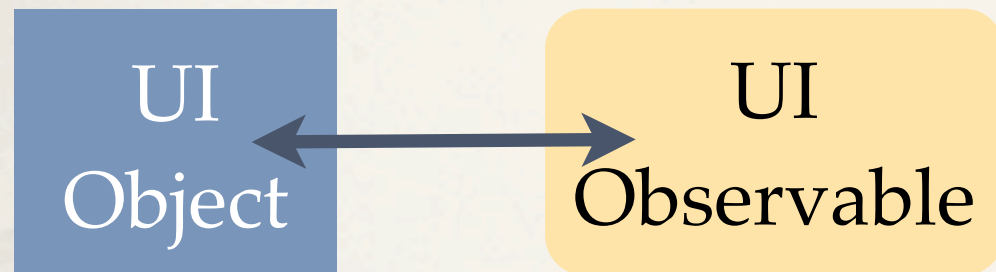


# Basic Design of Eclipse DB

---

- ✦ Eclipse Databinding is built around the observer pattern

Target

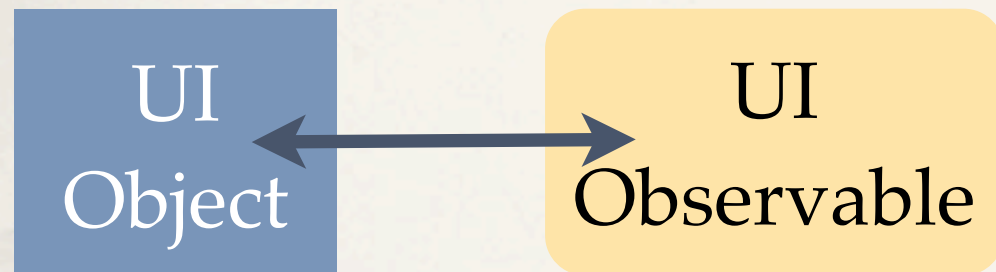


# Basic Design of Eclipse DB

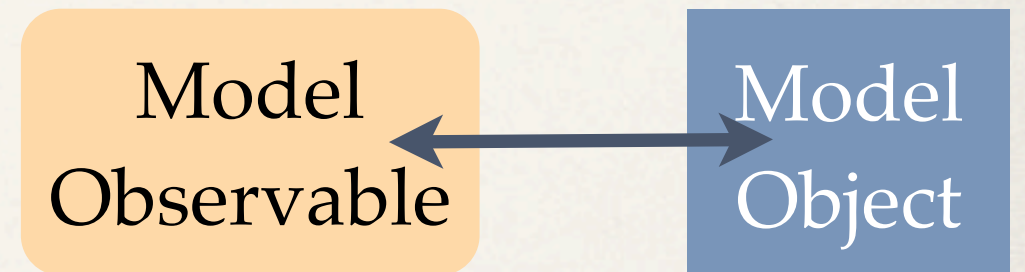
---

- ✦ Eclipse Databinding is built around the observer pattern

Target



Model





# Basic Design of Eclipse DB

---

- ✦ Eclipse Databinding is built around the observer pattern

Target

Model



# IProperty Interface

---

- ❖ IProperty is the domain independent abstraction of an attribute
  - ❖ IValueProperty: A single valued attribute
  - ❖ IListProperty: A multi valued attribute



# IProperty Interface

---

- ❖ IProperty is the domain independent abstraction of an attribute
  - ❖ IValueProperty: A single valued attribute
  - ❖ IListProperty: A multi valued attribute

## SWT

```
// Text text property
IWidgetValueProperty tProp =
    WidgetProperties.text(SWT.Modify);

// JFace Viewer Selection Property
IViewerValueProperty sProp =
    ViewerProperties.singleSelection();

// Button selection property
IWidgetValueProperty cProp =
    WidgetProperties.selection();
```

## Swing

```
// JTextField text property
IWidgetValueProperty tProp =
    SwingProperties.text(TextType.Modify);

// JComboBox selection Property
IWidgetValueProperty sProp =
    SwingProperties.singleSelectionValue();

// JButton selection property
IWidgetValueProperty cProp =
    SwingProperties.selection();
```



# IProperty Interface

---

- ❖ IProperty is the domain independent abstraction of an attribute
  - ❖ IValueProperty: A single valued attribute
  - ❖ IListProperty: A multi valued attribute

## EMF

```
// Single value property
IEMFValueProperty sProp = EMFProperties.value(
    AddressbookPackage.Literals.ADDRESS__STREET);

// Multi value property
IEMFListProperty mProp = EMFProperties.list(
    AddressbookPackage.Literals.COUNTRY__FEDERAL_STATES);
```

## JavaBean

```
// Single value property
IBeanValueProperty sProp =
    BeanProperties.value("street");

// Multi value property
IBeanListProperty mProp =
    BeanProperties.list("federalStates");
```



# Observable Creation - Model

---

## ❖ Simple observable creation

```
IEMFValueProperty prop = EMFProperties.value(  
    AddressbookPackage.Literals.PERSON__FIRSTNAME);  
  
IObservableValue v = prop.observe(person);
```

# Observable Creation - Model

---

## ❖ Simple observable creation

```
IEMFValueProperty prop = EMFProperties.value(  
    AddressbookPackage.Literals.PERSON__FIRSTNAME);  
  
IObservableValue v = prop.observe(person);
```

## ❖ Master-Detail observable creation

```
IObservableValue master = new WritableValue();  
  
IEMFValueProperty prop = EMFProperties.value(  
    AddressbookPackage.Literals.PERSON__FIRSTNAME);  
  
IObservableValue v = prop.observeDetail(master);  
  
master.setValue(person);
```



# Observable Creation - SWT/JFace

---

## ❖ WidgetProperties and ViewerProperties

```
IWidgetValueProperty tProp = WidgetProperties.text(SWT.Modify);  
IObservableValue v = tProp.observe(w_firstName);  
  
IViewerValueProperty sProp = ViewerProperties.singleSelection();  
IObservableValue v = sProp.observe(v_country);
```



# Observable Creation - SWT/JFace

---

## ❖ WidgetProperties and ViewerProperties

```
IWidgetValueProperty tProp = WidgetProperties.text(SWT.Modify);  
IObservableValue v = tProp.observe(w_firstName);  
  
IViewerValueProperty sProp = ViewerProperties.singleSelection();  
IObservableValue v = sProp.observe(v_country);
```

## ❖ Text-Widgets and SWT.Modify

```
IWidgetValueProperty tProp = WidgetProperties.text(SWT.Modify);  
IObservableValue v = tProp.observeDelayed(200, w_firstName);
```



# Databinding for Swing

---

## ❖ WidgetProperties from UFaceKit Project

```
IWidgetValueProperty tProp = SwingProperties.text(TextType.Modify);  
IObservableValue value = tProp.observe(w_firstName);  
  
IWidgetValueProperty sProp = SwingProperties.singleSelectionValue();  
IObservableValue value = sProp.observe(w_state)
```

# Connect Observables

---

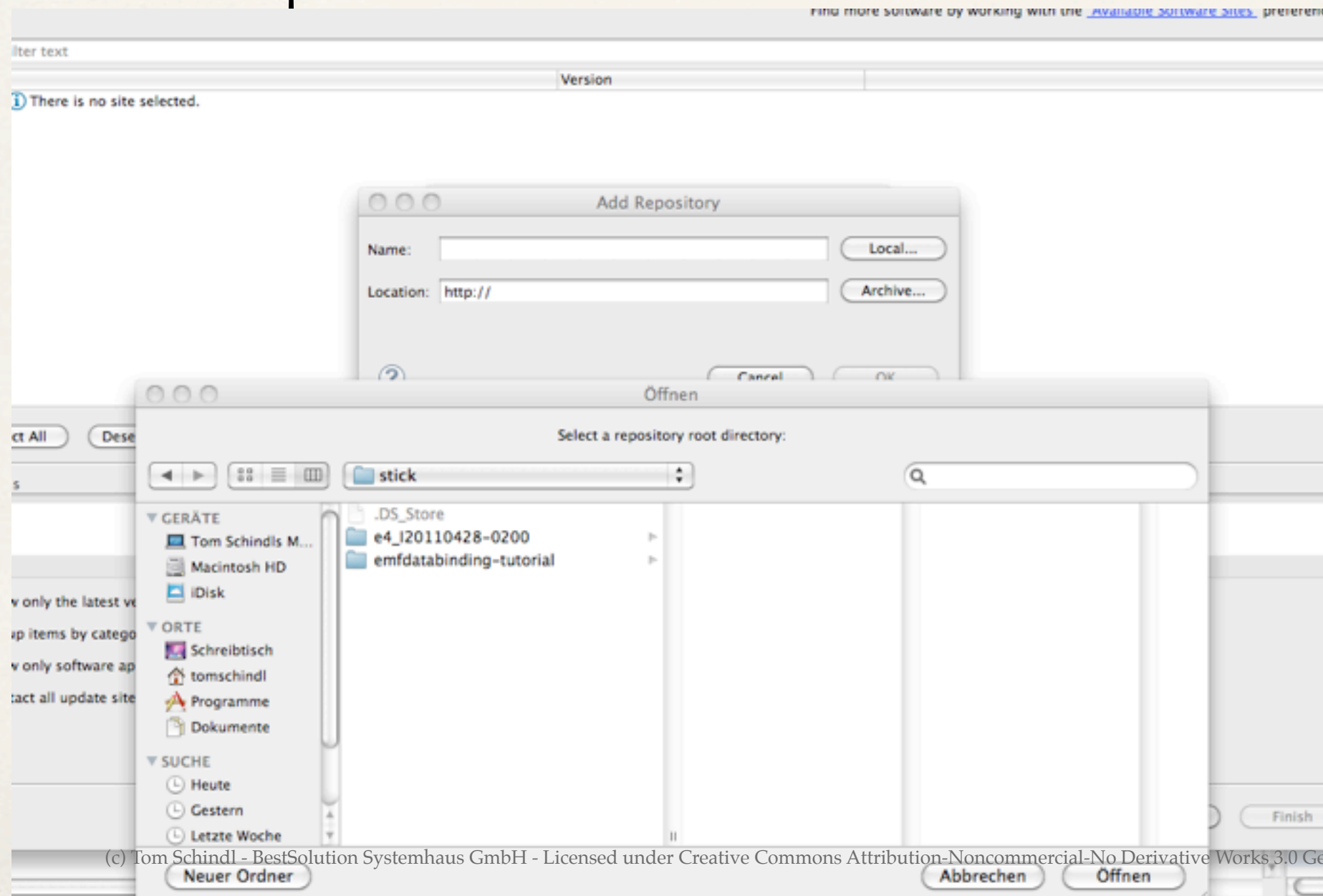
## ✧ Use EMFDataBindingContext

```
private void bindControls() {  
    EMFDataBindingContext dbc = new EMFDataBindingContext();  
  
    IWidgetValueProperty tProp = WidgetProperties.text(SWT.Modify);  
  
    {  
        IEMFValueProperty mProp = EMFProperties.value(  
            AddressbookPackage.Literals.PERSON__FIRSTNAME);  
  
        dbc.bindValue(tProp.observe(w_firstName), mProp.observeDetail(master));  
    }  
}
```



# Lab 1 - Eclipse Setup

- ✧ Add local e4 update-site



(c) Tom Schindl - BestSolution Systemhaus GmbH - Licensed under Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 Germany License.

# Lab 1 - Eclipse Setup

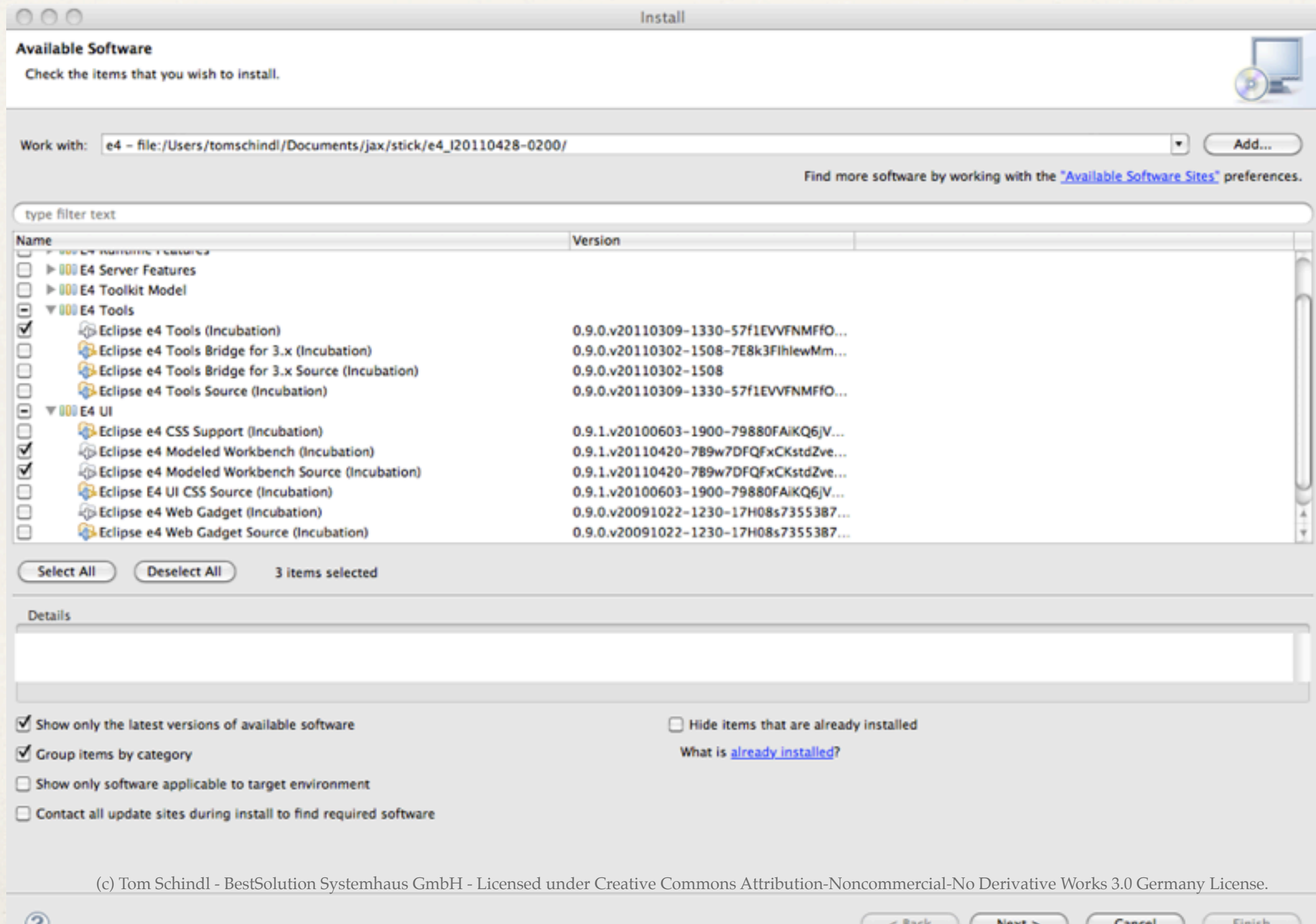
---

- ❖ Install the following projects from e4\_I20110428-0200
  - ❖ E4 Tools > Eclipse e4 Tools
  - ❖ E4 UI > Eclipse e4 Modeled Workbench
  - ❖ E4 UI > Eclipse e4 Modeled Workbench Source

**Important: Uncheck „Contact all update sites during install to find required software“**



# Lab 1 - Eclipse Setup



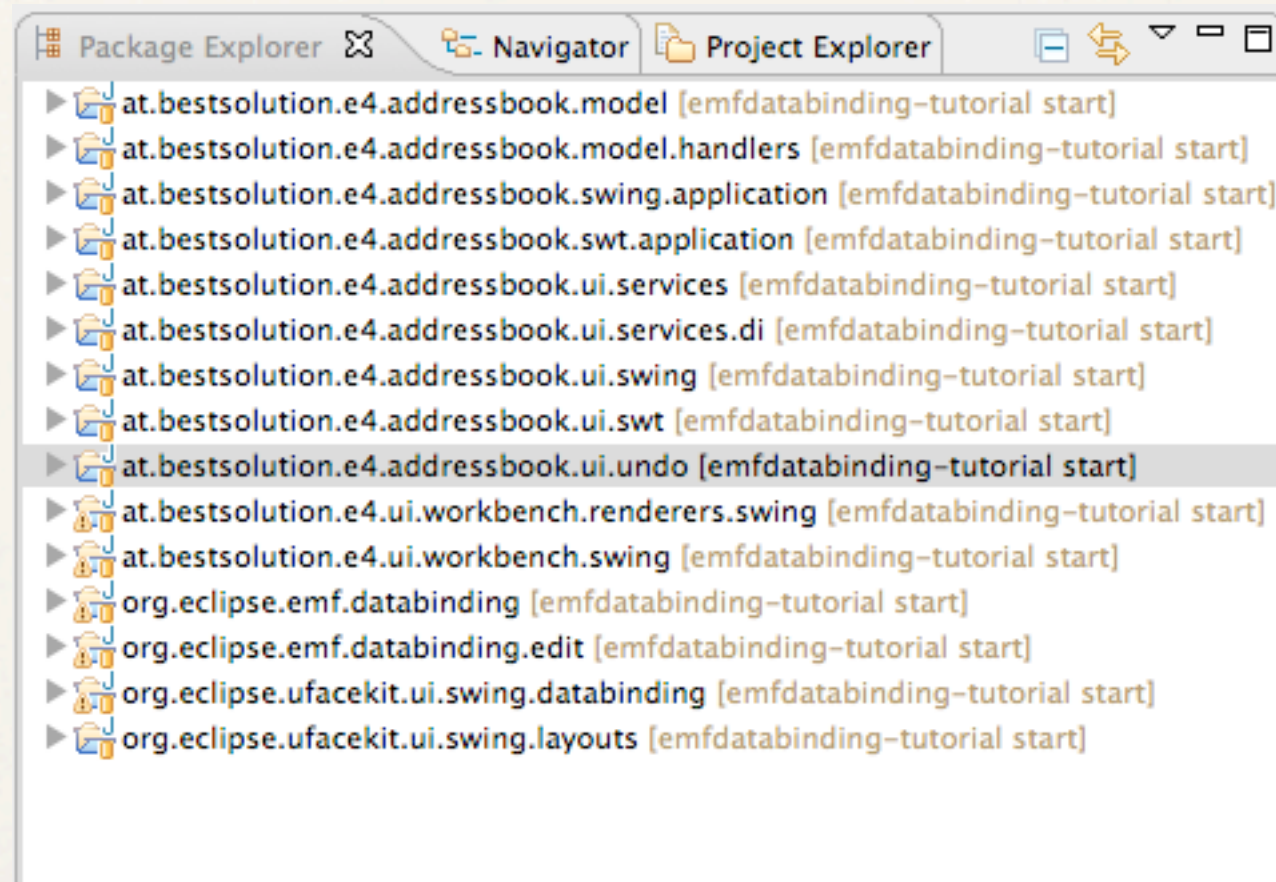
(c) Tom Schindl - BestSolution Systemhaus GmbH - Licensed under Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 Germany License.



# Lab 1 - Eclipse Setup

---

- ✧ Import projects from `emfdatatabinding-tutorial-init`





# Lab 1 for SWT follower

---

- ❖ Implement Text-Binding in `bindControls()` Method
  - ❖ `at.bestsolution.e4.addressbook.ui.swt.PersonForm`
  - ❖ `at.bestsolution.e4.addressbook.ui.swt.AddressForm`

# Lab 1 for Swing follower

---

- ❖ Implement JTextField-Binding in `bindControls()` Method
  - ❖ `at.bestsolution.e4.addressbook.ui.swing.PersonForm`
  - ❖ `at.bestsolution.e4.addressbook.ui.swing.AddressForm`



# JFace Viewers

---

- ❖ JFace Viewers
  - ❖ LabelProvider: Translate Domain Object into String / Image
  - ❖ ContentProvider: Converts input e.g. `java.util.List` into internal representation
  - ❖ Input: The input to pass to the content provider

# Databinding support for JFace

---

## ❖ ObservableListContentProvider

```
ItemProperty mProp = EMFProperties.list  
    (AddressbookPackage.Literals.ADDRESS_BOOK__COUNTRIES);  
  
v_country = new ComboViewer(w_country);  
  
ObservableListContentProvider cp = new ObservableListContentProvider();  
  
v_country.setContentProvider(cp);  
v_country.setInput(mProp.observe(book));
```



# Databinding support for JFace

---

## ❖ ObservableListContentProvider

```
IEMFListProperty mProp = EMFProperties.list  
    (AddressbookPackage.Literals.ADDRESS_BOOK__COUNTRIES);  
  
v_country = new ComboViewer(w_country);  
  
ObservableListContentProvider cp = new ObservableListContentProvider();  
  
v_country.setContentProvider(cp);  
v_country.setInput(mProp.observe(book));
```

## ❖ ObservableMapLabelProvider

```
IValueProperty props = EMFProperties.value(AddressbookPackage.Literals.COUNTRY__NAME);  
  
ObservableMapLabelProvider lp = new ObservableMapLabelProvider(  
    props.observeDetail(cp.getKnownElements()));  
v_country.setLabelProvider(lp);
```

(c) Tom Schindl - BestSolutionSystemhaus GmbH - Licensed under Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 Germany License.



# Databinding Support for Swing

## ❖ ObservableComboBoxModel and ObservableListModel

```
IEMFListProperty mProp = EMFProperties.list  
    (AddressbookPackage.Literals.COUNTRY__FEDERAL_STATES);  
  
IValueProperty[] props = {  
    EMFProperties.value(AddressbookPackage.Literals.FEDERAL_STATE__NAME)  
};  
  
ObservableComboBoxModel<FederalState> model = new ObservableComboBoxModel<FederalState>(  
    mProp.observeDetail(value), props);
```



# Databinding Support for Swing

## ❖ ObservableComboBoxModel and ObservableListModel

```
ITEMFListProperty mProp = EMFProperties.list  
    (AddressbookPackage.Literals.COUNTRY__FEDERAL_STATES);  
  
IValueProperty[] props = {  
    EMFProperties.value(AddressbookPackage.Literals.FEDERAL_STATE__NAME)  
};  
  
ObservableComboBoxModel<FederalState> model = new ObservableComboBoxModel<FederalState>(  
    mProp.observeDetail(value), props);
```

## ❖ ObservableListCellRenderer

```
w_state.setRenderer(new ObservableListCellRenderer<FederalState>(  
    model, new ILabelDelegate<FederalState>() {  
  
        @Override  
        public String getText(FederalState object,  
            IObservableMap[] maps) {  
            return object == null ? "" : object.getName();  
        }  
    }  
));
```

(c) Tom Schindl - BestSolutionSystemhaus GmbH - Licensed under Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 Germany License.



# Lab 2 for SWT follower

---

- ❖ `AddressForm#init`: Initialize Country viewer
- ❖ `AddressForm#bindControls`: Bind selection of country viewer
- ❖ `PersonList#init`: Initialize Person viewer



# Lab 2 for Swing follower

---

- ❖ `AddressForm#init`: Initialize Country JComboBox
- ❖ `AddressForm#bindControls`: Bind selection of country JComboBox
- ❖ `PersonList#init`: Initialize Person JList

# EMF Edit integration

---

- ❖ Simply use EMFEditProperties instead of EMFProperties

```
IEMFValueProperty prop = EMFProperties.value(  
    AddressbookPackage.Literals.PERSON__FIRSTNAME);
```

```
IObservableValue v = prop.observe(person);
```



# EMF Edit integration

---

- ❖ Simply use EMFEditProperties instead of EMFProperties

```
IEMFValueProperty prop = EMFEditProperties.value(editingDomain,  
    AddressbookPackage.Literals.PERSON__FIRSTNAME);
```

```
IObservableValue v = prop.observe(person);
```

# EMF Edit integration

---

- ❖ Simply use EMFEditProperties instead of EMFProperties

```
IEMFValueProperty prop = EMFEditProperties.value(editingDomain,  
    AddressbookPackage.Literals.PERSON__FIRSTNAME);
```

```
IObservableValue v = prop.observe(person);
```

**Don't forget: Use Delayed UI-Observables!**



# Lab 3 for SWT follower

---

- ❖ `PersonForm#bindControls(EditingDomain)`: Add binding for first and lastname
- ❖ Open & Launch:  
`at.bestsolution.e4.addressbook.swt.application/  
at.bestsolution.e4.addressbook.swt.application.prod  
uct`

# Lab 3 for Swing follower

---

- ❖ `PersonForm#bindControls(EditingDomain)`: Add binding for first and lastname
- ❖ Open & Launch:  
`at.bestsolution.e4.addressbook.swing.application/  
at.bestsolution.e4.addressbook.swing.application.pr  
oduct`



# Additional Information

---

- ❖ Blog: <http://tomsondev.bestsolution.at>
- ❖ Blog: <http://tomsondev.bestsolution.at/category/databinding/emf/>
- ❖ UFaceKit: <http://www.eclipse.org/ufacekit>
- ❖ e4: <http://wiki.eclipse.org/E4>
- ❖ <https://github.com/tomsontom/>