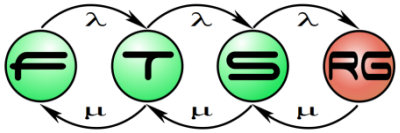
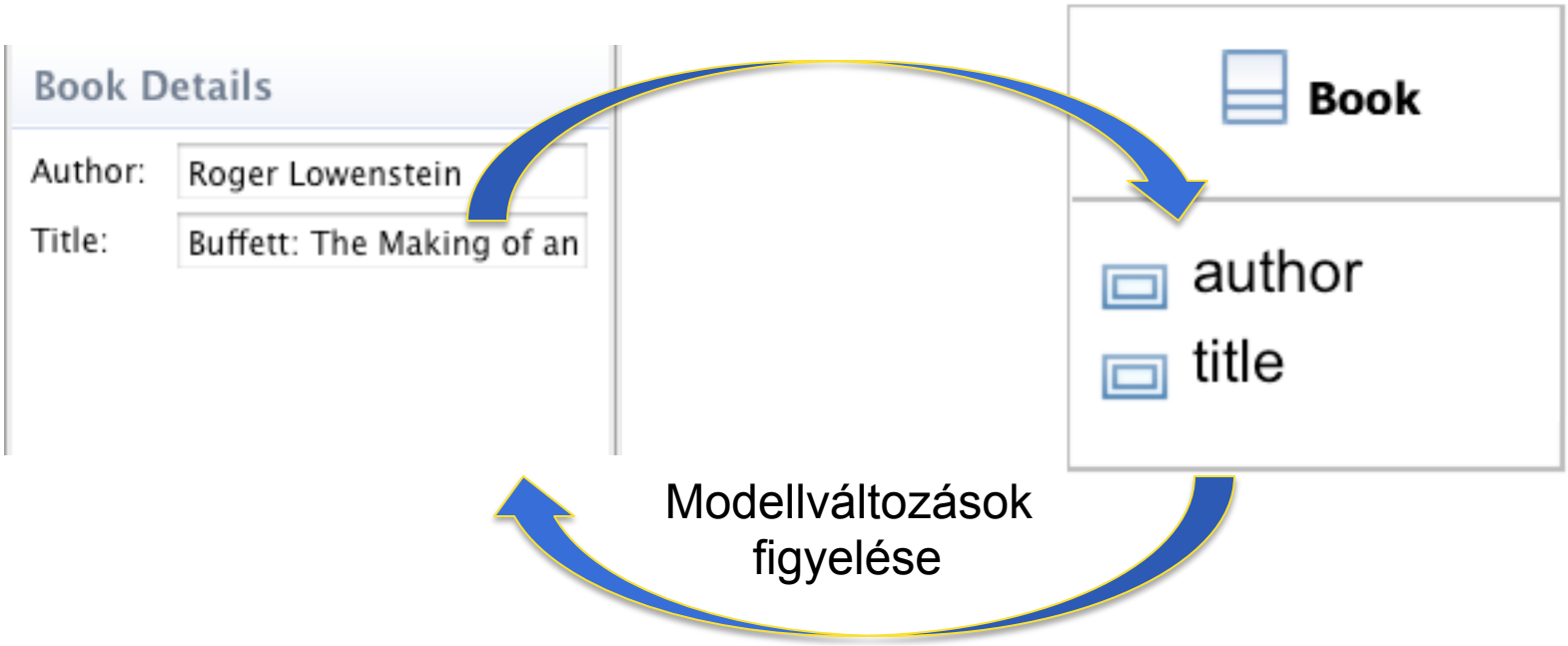
# Data Binding



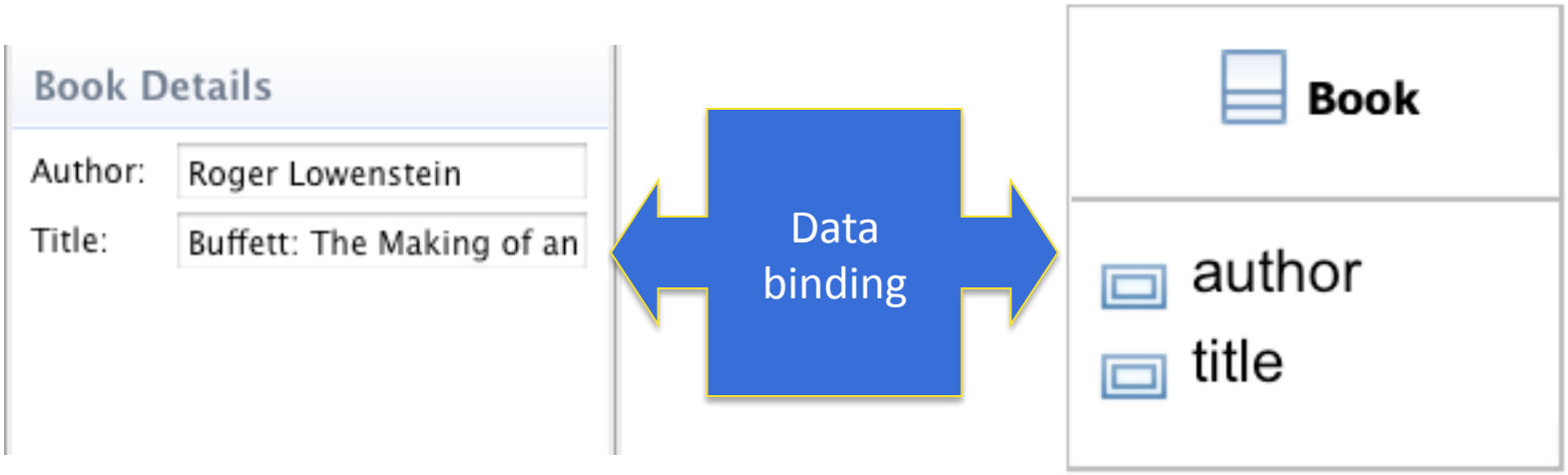
# JFace Data Binding

# Felhasználói felület és adatmodell




- Szinkronizáció: bonyolult, de mechanikus műveletek
- Könnyű hibázni
- Automatizálható?

# Felhasználói felület és adatmodell




- Adatkötés GUI elemekhez
  - Grafikus felület és reprezentált adat közötti szinkronizáció
- JFace data binding: Eclipse 3.3 óta
  - Modell és GUI objektumok megadása
  - Automatikus konverzió és validáció
  - Adatintenzív alkalmazásokban nagy segítség

# JFace Data Binding - Alapfogalmak

- Observable
  - Struktúra (pl. érték, lista, halmaz, stb.)
  - Megfigyelhető állapotváltozások
- Binding (kötés)
  - Kapcsolat két Observable között
  - Egy- vagy kétirányú szinkronizáció
- Data binding context
  - A kötések tárolója
- Realm
  - Observable hozzáférések sorosítása
  - RCP alkalmazás, ill. Eclipse plug-in fejlesztéskor a platform létrehozza

# Data Binding függőségek

- org.eclipse.core.databinding,
- org.eclipse.core.databinding.beans,
- org.eclipse.jface.databinding,
- com.ibm.icu

# Data Binding példa

```
public void createPartControl(Composite parent){
  parent.setLayout(new GridLayout());
  Text text = new Text(parent, SWT.BORDER);
  Label label = new Label(parent, SWT.NONE);
  Button button = new Button(parent, SWT.PUSH);
  button.setText("Double!");
  button.addSelectionListener(new SelectionAdapter() {
    public void widgetSelected(SelectionEvent e){
     model.setAmount(model.getAmount() * 2);
    }
  });
  DataBindingContext dbc = new DataBindingContext();
  IObservableValue modelObservable =
   BeansObservables.observeValue(model, "amount");
  dbc.bindValue(SWTObservables.observeText(text, SWT.Modify),
    modelObservable, null, null);
  dbc.bindValue(SWTObservables.observeText(label),
    modelObservable, null, null);
}
```

# Data Binding példa

```
public void createPartControl(Composite parent){
  parent.setLayout(new GridLayout());
  Text text = new Text(parent, SWT.BORDER);
  Label label = new Label(parent, SWT.NONE);
  Button button = new Button(parent, SWT.PUSH);
  button.setText("Double!");
  button.addSelectionListener(new SelectionAdapter() {
    public void widgetSelected(SelectionEvent e){
     model.setAmount(model.getAmount() * 2);
    }
  });
  DataBindingContext dbc = new DataBindingContext();
  IObservableValue modelObservable =
   BeansObservables.observeValue(model, "amount");
  dbc.bindValue(SWTObservables.observeText(text, SWT.Modify),
    modelObservable, null, null);
  dbc.bindValue(SWTObservables.observeText(label),
    modelObservable, null, null);
}
```

Form összeállítása

# Data Binding példa

```
public void createPartControl(Composite parent){
  parent.setLayout(new GridLayout());
  Text text = new Text(parent, SWT.BORDER);
  Label label = new Label(parent, SWT.NONE);
  Button button = new Button(parent, SWT.PUSH);
  button.setText("Double!");
  button.addSelectionListener(new SelectionAdapter() {
    public void widgetSelected(SelectionE
     model.setAmount(model.getAmount() *
    }
  });
  DataBindingContext dbc = new DataBindingContext();
  IObservableValue modelObservable =
   BeansObservables.observeValue(model, "amount");
  dbc.bindValue(SWTObservables.observeText(text, SWT.Modify),
    modelObservable, null, null);
  dbc.bindValue(SWTObservables.observeText(label),
    modelObservable, null, null);
}
```

Modell írása

# Data Binding példa

```
public void createPartControl(Composite parent){
  parent.setLayout(new GridLayout());
  Text text = new Text(parent, SWT.BORDER);
  Label label = new Label(parent, SWT.NONE);
  Button button = new Button(parent, SWT.PUSH);
  button.setText("Double!");
  button.addSelectionListener(new SelectionAdapter() {
    public void widgetSelected(SelectionEver
     model.setAmount(model.getAmount() * 2)
    }
  });
  DataBindingContext dbc = new DataBindi
  IObservableValue modelObservable =
   BeansObservables.observeValue(model, "amount");
  dbc.bindValue(SWTObservables.observeText(text, SWT.Modify),
    modelObservable, null, null);
  dbc.bindValue(SWTObservables.observeText(label),
    modelObservable, null, null);
}
```

A model objektum amount tulajdonságát figyeljük

# Data Binding példa

```
public void createPartControl(Composite parent){
  parent.setLayout(new GridLayout());
  Text text = new Text(parent, SWT.BORDER);
  Label label = new Label(parent, SWT.NONE);
  Button button = new Button(parent, SWT.PUSH);
  button.setText("Double!");
  button.addSelectionListener(new SelectionAdapter() {
    public void widgetSelected(SelectionEvent e){
     model.setAmount(model.getAmount() * 2);
    }
  });
  DataBindingContext dbc = new DataBindingContext();
  IObservableValue modelObservable =
   BeansObservables.observeValue(model, "amoun
  dbc.bindValue(SWTObservables.observeText(tex
    modelObservable, null, null);
  dbc.bindValue(SWTObservables.observeText(label
    modelObservable, null, null);
}
```

Szövegmező kötése

# Data Binding példa

```
public void createPartControl(Composite parent){
  parent.setLayout(new GridLayout());
  Text text = new Text(parent, SWT.BORDER);
  Label label = new Label(parent, SWT.NONE);
  Button button = new Button(parent, SWT.PUSH);
  button.setText("Double!");
  button.addSelectionListener(new SelectionAdapter() {
    public void widgetSelected(SelectionEvent e){
     model.setAmount(model.getAmount() * 2);
    }
  });
  DataBindingContext dbc = new DataBindingContext();
  IObservableValue modelObservable =
   BeansObservables.observeValue(model, "amount");
  dbc.bindValue(SWTObservables.observeText(text, SWT.Modify),
    modelObservable, null, null);
  dbc.bindValue(SWTObservables.observeTe
    modelObservable, null, null);
}
```

Címke kötése

# Observable előállítása: SWT widgetek

- SWTObservables osztály
  - Factory SWT tulajdonságok vizsgálatára
- Példa:
  - Szöveg:
    - SWTObservables.observeText()
  - Engedélyezettség:
    - SWTObservables.observeEnabled()

# Observable előállítása: Modell

- PropertyChangeSupport objektum használata
  - Listener kezelésre
- Szükséges
  - Publikus lekérdező és beállító metódusok
  - PropertyChangeSupport értesítése változásról

# Modell osztály

```
public class Model {
  private PropertyChangeSupport changeSupport = new PropertyChangeSupport
   (this);
  public void addPropertyChangeListener(String propertyName,
   PropertyChangeListener listener {
     changeSupport.addPropertyChangeListener(propertyName, listener);
  }
  public void removePropertyChangeListener(String propertyName,
   PropertyChangeListener listener) {
    changeSupport.removePropertyChangeListener(propertyName, listener);
  }
  private int amount = 0;
  public void setAmount(int newAmount) {
   int oldAmount = this.amount;
   this.amount = newAmount;
   changeSupport.firePropertyChange("amount", oldAmount, newAmount);
 }
 public int getAmount() { return amount; }
}
```

# Modell osztály

```
public class Model {
  private PropertyChangeSupport changeSupport = new PropertyChangeSupport
   (this);
  public void addPropertyChangeListener(String propertyName,
   PropertyChangeListener listener {
     changeSupport.addPropertyChangeListener(propertyName, listener);
  }
  public void removePropertyChangeListener(String propertyName,
   PropertyChangeListener listener) {
    changeSupport.removePropertyChangeListener(propertyName, listener);
  }
  private int amount = 0;
  public void setAmount(int newAmount) {
   int oldAmount = this.amount;
   this.amount = newAmount;
   changeSupport.firePropertyChange("amount", oldAmount, newAmount);
 }
 public int getAmount() { return amount; }
}
```

# Modell osztály

```
public class Model {
  private PropertyChangeSupport changeSupport =                pport
   (this);
  public void addPropertyChangeListener(String
   PropertyChangeListener listener {
     changeSupport.addPropertyChangeListener(propertyName, listener);
  }
  public void removePropertyChangeListener(String propertyName,
   PropertyChangeListener listener) {
    changeSupport.removePropertyChangeListener(propertyName, listener);
  }
  private int amount = 0;
  public void setAmount(int newAmount) {
   int oldAmount = this.amount;
   this.amount = newAmount;
   changeSupport.firePropertyChange("amount", oldAmount, newAmount);
 }
 public int getAmount() { return amount; }
}
```

Eseménykezelés támogatása

# Modell osztály

```
public class Model {
  private PropertyChangeSupport changeSupport = new PropertyChangeSupport
   (this);
  public void addPropertyChangeListener(String propertyName,
   PropertyChangeListener listener {
     changeSupport.addPropertyChangeListener(prop
  }
  public void removePropertyChangeListener(Strinc
   PropertyChangeListener listener) {
    changeSupport.removePropertyChangeListener(pr
  }
  private int amount = 0;
  public void setAmount(int newAmount) {
   int oldAmount = this.amount;
   this.amount = newAmount;
   changeSupport.firePropertyChange("amount", oldAmount, newAmount);
 }
 public int getAmount() { return amount; }
}
```

Eseménykezelők hozzáadása, eltávolítása

# Modell osztály

```
public class Model {
  private PropertyChangeSupport changeSupport = new PropertyChangeSupport
   (this);
  public void addPropertyChangeListener(String propertyName,
   PropertyChangeListener listener {
     changeSupport.addPropertyChangeListener(propertyName, listener);
  }
  public void removePropertyChangeListener(String propertyName,
   PropertyChangeListener listener) {
    changeSupport.removePropertyChangeListener(propertyName, listener);
  }
  private int amount = 0;
  public void setAmount(int newAmount) {
   int oldAmount = this.amount;
   this.amount = newAmount;
   changeSupport.firePropertyChange("amount", oldAmount, newAmount);
 }
 public int getAmount() { return amount; }
}
```

Változás értesítés

# Validáció

- A példakód szövegbeviteli mezőből számot állít elő
  - Alapértelmezett konverzió és validáció van megadva

```
dbc.bindValue(SWTObservables.observeText(text, SWT.Modify),

    modelObservable, null, null);
```

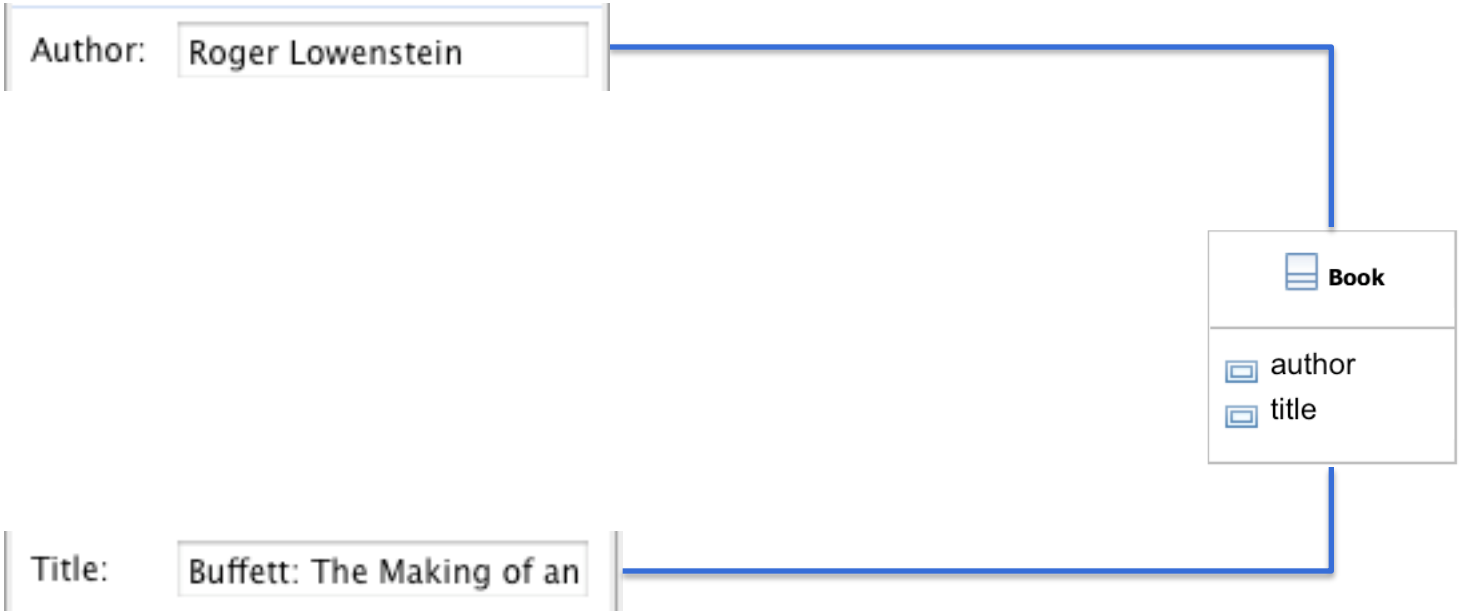
Saját validátor is megadható

```
dbc.bindValue
  (SWTObservables.observeText(text, SWT.Modify),
   modelObservable,
   // UI to model
   new UpdateValueStrategy().
     setAfterConverValidator(anIntValidator),
   //model to UI
   new UpdateValueStrategy.
     setConverter(anIntToStringConverter);
```

# UpdateValueStrategy

- A validáció/konverzió/update összefogása
- Fázisok
  - Validate after get
    - Az érték forrásból kiolvasása utáni validáció
  - Konverzió
    - Az érték átalakítása a forrás domainből a cél domainbe
  - Validate after conversion
    - A konverzió utáni ellenőrzés
  - Validate before set
    - A cél érték beállítása előtti validáció
  - Value set
    - A célérték beállítása a cél objektumon

# Adatkötés folyamata - Példa




Mindkét szövegmező kétirányú kötést létesít a modellel

# Adatkötés folyamata – Példa

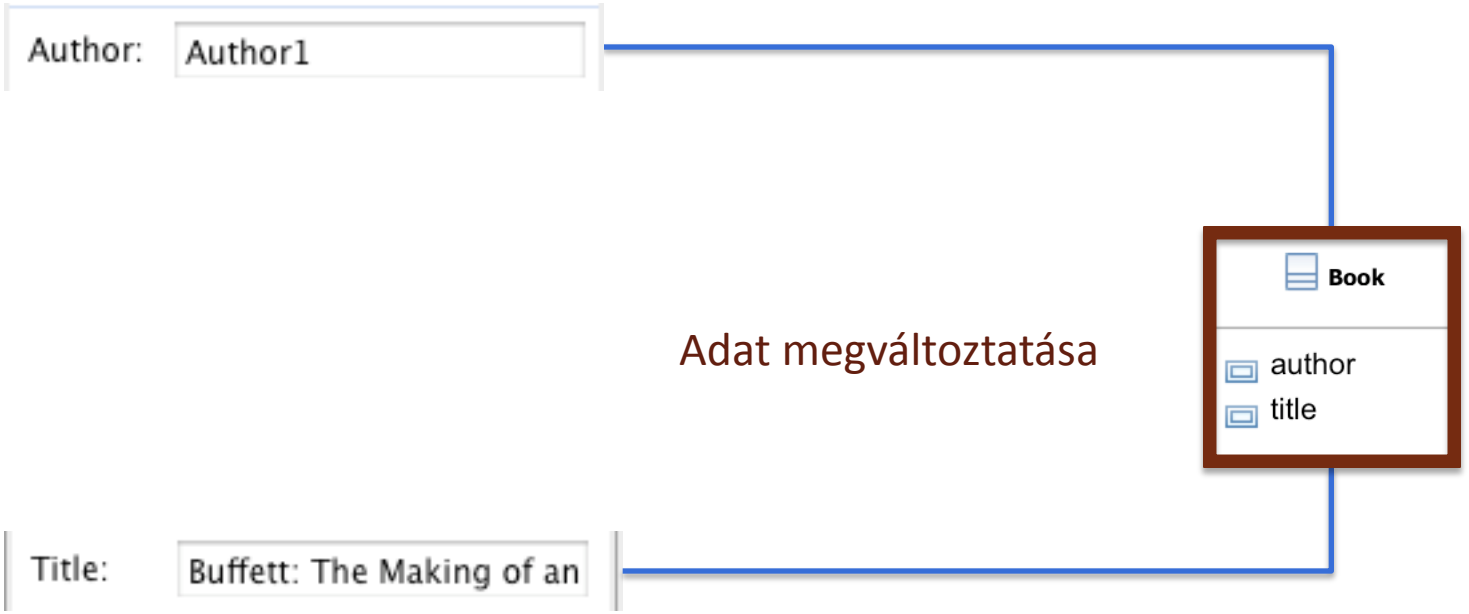

Szerző megváltozott

# Adatkötés folyamata - Példa

# Adatkötés folyamata - Példa

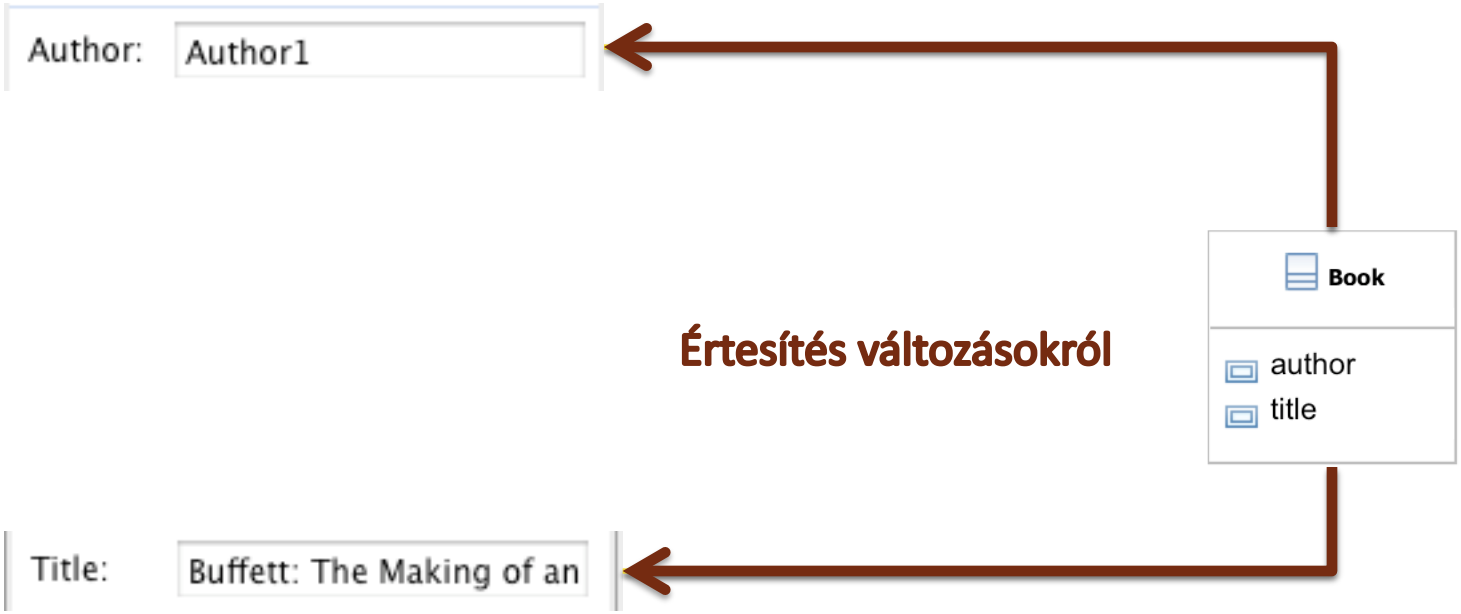
# Adatkötés folyamata - Példa

# Adatkötés folyamata – Példa

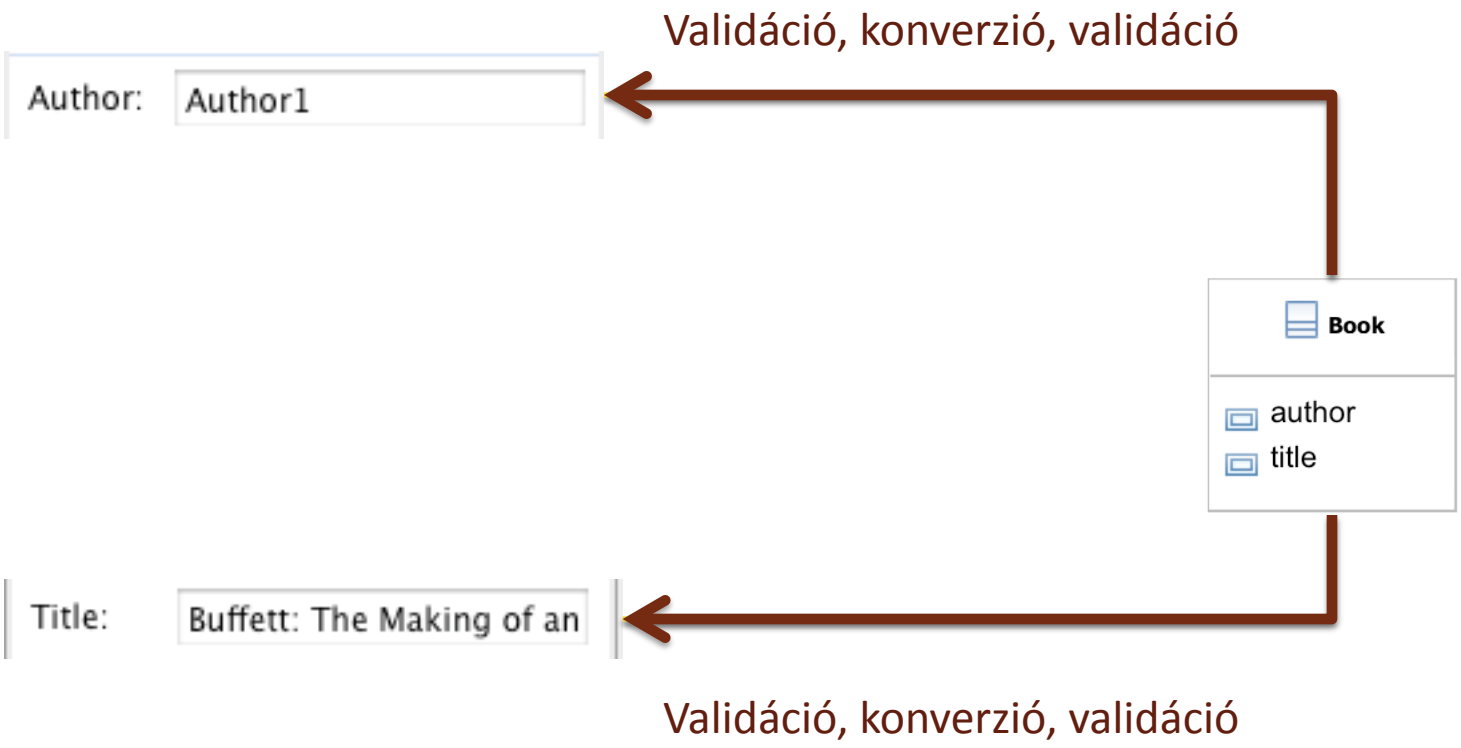



Szerző megváltozott

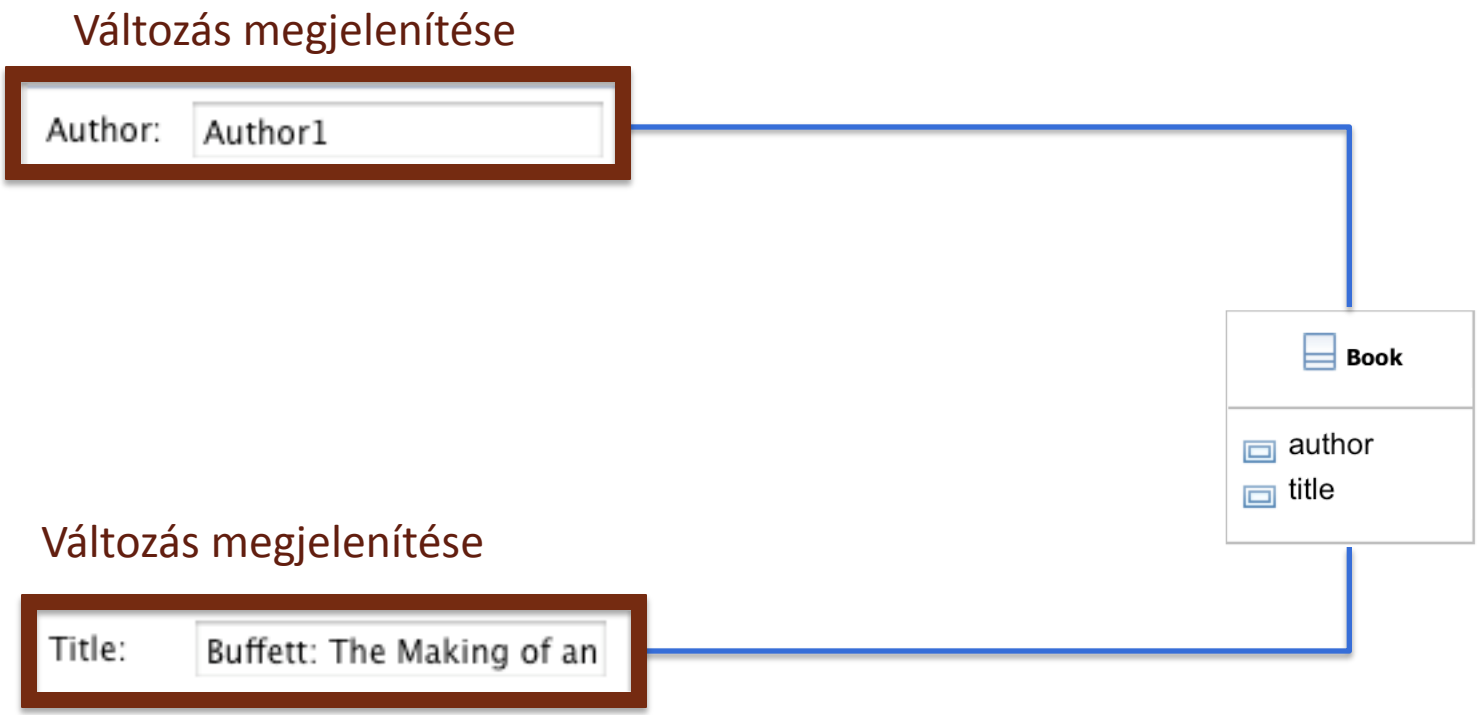
# Adatkötés folyamata - Példa



# Adatkötés folyamata - Példa



# Adatkötés folyamata - Példa



# Validátorok írása

- IValidator interfész használható
- Visszatérési érték előállítása:
  - ValidationStatus.ok()
  - ValidationStatus.info(String message)
  - ValidationStatus.warning(String message)
  - ValidationStatus.error(String message)

# Konverter írása

- IConverter interfész
- Forrás és cél osztály megadása
- Tényleges átalakítás
- Számok és string közti átalakításra:
  - NumberToStringConverter
  - StringToNumberConverter

# Validációs hibák megjelenítése

- Hibainformációk
  - Nem IStatus formátumban
  - IObservable -> bindelhető
- Minden elem hibája:
  - `new AggregateValidationStatus (dbc.getBindings(), AggregateValidationStatus.MAX_SEVERITY), null, null);`
- Egy b nevű változóban tárolt kötés hibája
  - `b.getValidationStatus()`
- Ezek teljesen analóg módon köthetőek ki a felületre

```
dbc.bindValue(SWTObservables.observeText
   (individualErrorLabel), b.getValidationStatus, null,
   null);
```

# Függő és számolt Observable értékek

- Minden attribútum olvasást is figyel a rendszer
  - Függő elemek nyilvántarthatóak
  - Számolt attribútumok (ComputedValue, ComputedList) automatikusan frissülnek
- Példa számolt attribútumra

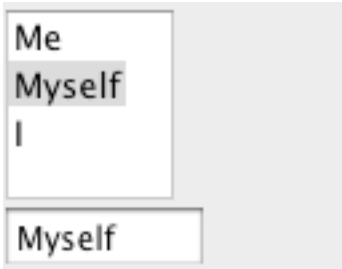
```
final IObservableValue firstName = SWTObservables.observeText
   (firstNameField, SWT.Modify);

final IObservableValue lastName = SWTObservables.observeText
   (lastNameField, SWT.Modify);

 IObservableValue formattedName = new ComputedValue() {
    protected Object calculate() {
       return lastName.getValue() + firstName.getValue();
    }
 };
```

# Master-Detail adatkötés

- Feladat:
  - Kijelölés figyelése (és kötése)
  - Ha változik a kijelölés
    - Meglevő kötést eldobni
    - Új kötést létrehozni
- Támogatás: master-detail adatkötés
  - Figyelő hozzáadása a kijelöléshez (Master Observable)
  - Kijelölés vizsgálata (observeDetailValue)

# Master-Detail példa

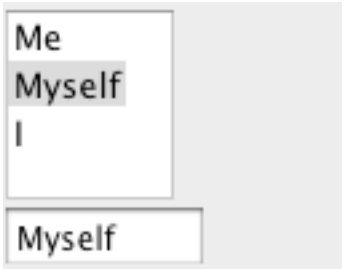
```
// 1. Observe changes in selection.
IObservableValue selection =
  ViewersObservables.observeSingleSelection(viewer);
// 2. Observe the name property of the current
  selection.
IObservableValue detailObservable =
  BeansObservables .observeDetailValue(selection,
  "name", String.class);
// 3. Bind the Text widget to the name detail
  (selection's name).
new DataBindingContext().bindValue(
  SWTObservables.observeText(name, SWT.None),
  detailObservable,
  new UpdateValueStrategy(false,
    UpdateValueStrategy.POLICY_NEVER),
  null);
```



# Master-Detail példa

```
// 1. Observe changes in selection.
IObservableValue selection =
  ViewersObservables.observeSingleSelection(viewer);
// 2. Observe the name property of the current
  selection.
IObservableValue detailObservable =
  BeansObservables .observeDetailVa
  "name", String.class);
// 3. Bind the Text widget to the name detail
  (selection's name).
new DataBindingContext().bindValue(
  SWTObservables.observeText(name, SWT.None),
  detailObservable,
  new UpdateValueStrategy(false,
    UpdateValueStrategy.POLICY_NEVER),
  null);
```

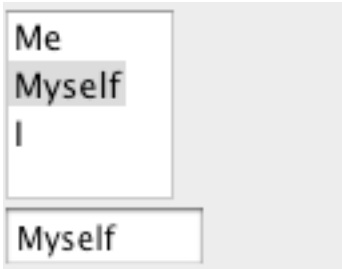
Observer készítése egyszeres kijelöléshez (JFace Viewer)



# Master-Detail példa

```
// 1. Observe changes in selection.
IObservableValue sel
  ViewersObservable                eSelection(viewer);
// 2. Observe the na                the current
  selection.
IObservableValue detailObservable =
  BeansObservables .observeDetailValue(selection,
  "name", String.class);
// 3. Bind the Text widget to the name detail
  (selection's name).
new DataBindingContext().bindValue(
  SWTObservables.observeText(name, SWT.None),
  detailObservable,
  new UpdateValueStrategy(false,
    UpdateValueStrategy.POLICY_NEVER),
  null);
```

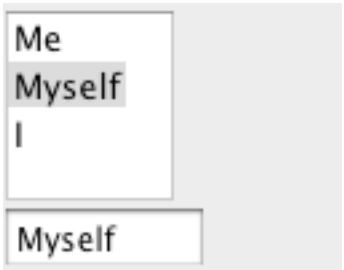
Detail Observable létrehozása



# Master-Detail példa

```
// 1. Observe changes in selection.
IObservableValue selection =
  ViewersObservables.observeSingleSelection(viewer);
// 2. Observe the name property of the current
  selection.
IObservableValue detailObservable =
  BeansObservables .observeDetailValue(selection,
  "name", String.class);
// 3. Bind the Text widget to the
  (selection's name).
new DataBindingContext().bindValue(
  SWTObservables.observeText(name, SWT.None),
  detailObservable,
  new UpdateValueStrategy(false,
    UpdateValueStrategy.POLICY_NEVER),
  null);
```

Adatkötés



# JFace Databinding és Viewerek

- Segédfüggvények adatfeltöltéshez
  - ViewerSupport osztály
  - Kötés lista és fastruktúra számára
    - WritableList: megfigyelhető osztálylista
    - LabelProperty: címkestruktúra

```
bookList = new WritableList(
  manager.getAllBooks(), Book.class);
ViewerSupport.bind(viewer, bookList,
  BeanProperties.value(Book.class,
  "description"));
```

# Összefoglalás

- Magas szintű adat-GUI szinkronizáció
- Gyorsan fejlődik
- Részletesebb dokumentáció az Eclipse wikin:
  - **http://wiki.eclipse.org/index.php/JFace_Data_Binding**

# EMF Data Binding

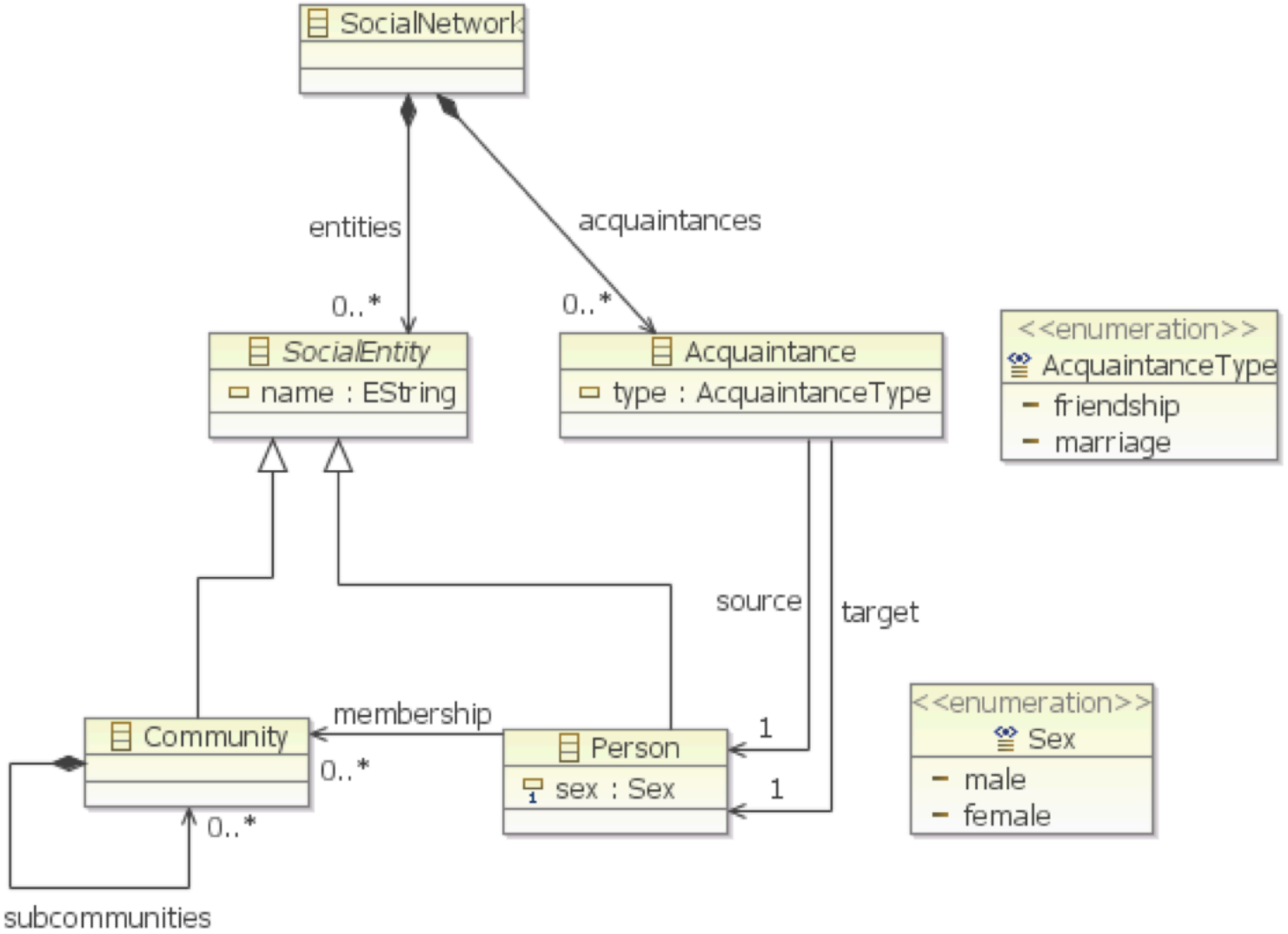
# Adatkötés EMF modellek esetén

- Miben más?
  - Bejárandó struktúra
    - Ismert (metamodell)
  - EMF objektumok
    - Más IValidator/Converter
  - Rögzített notifikációs mechanizmus
    - Nem Java bean alapú

# Eclipse 3.4-ig

- Java objektumokhoz hasonló felület
  - (Java bean)
  - EMFObservablesFactory osztály

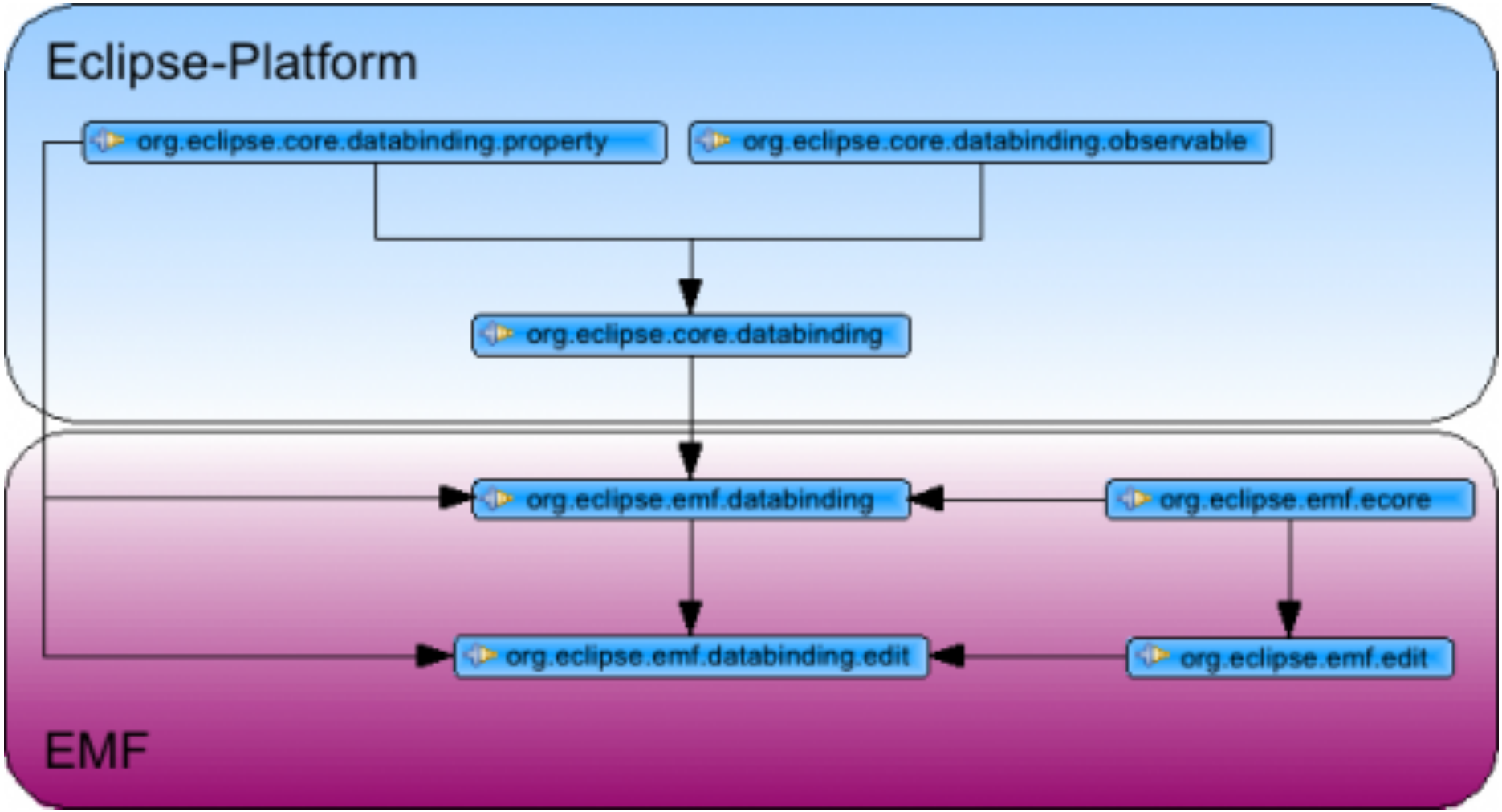
# Példa: Kapcsolati háló



# Properties API (Eclipse 3.5-től)

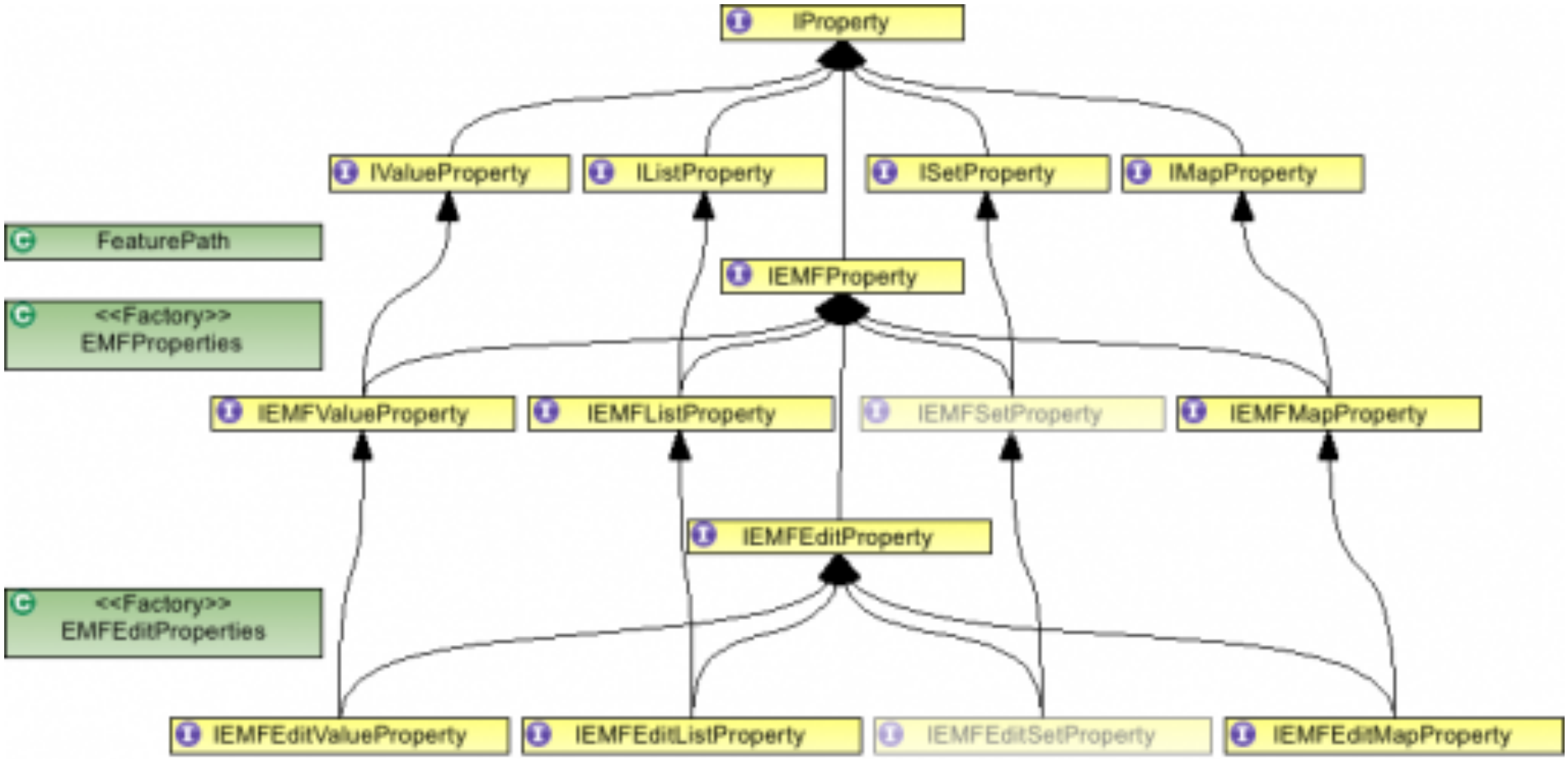
- Modelltulajdonságok
  - Reprezentáció
  - Bejárás
  - +Observable készítés

# Függőségek




Forrás: http://tomsondev.bestsolution.at/2009/06/07/galileo-emf-databinding-part-2/

# Properties API




Forrás: http://tomsondev.bestsolution.at/2009/06/07/galileo-emf-databinding-part-2/

# Tulajdonságok elérése – 1.

- Kiindulás
  - Egy EMF modell objektum
- Közvetlen tulajdonságelérés
  - Típus alapján
    - Package definiált literál alapján
  - Példa: Személy/Közösség neve
    - `SocialnetworkPackage.Literals.SOCIAL_ENTITY__NAME`

# Tulajdonságok elérése – 2.

- Közvetett tulajdonságelérés
  - Több lépésből érhető el
    - Referenciák felsorolása
  - Példa: ismerős neve

```
FeaturePath path = FeaturePath.fromList(
  SocialnetworkPackage.Literals.ACQUAINTANCE__TARGET,
  SocialnetworkPackage.Literals.SOCIAL_ENTITY__NAME);
```

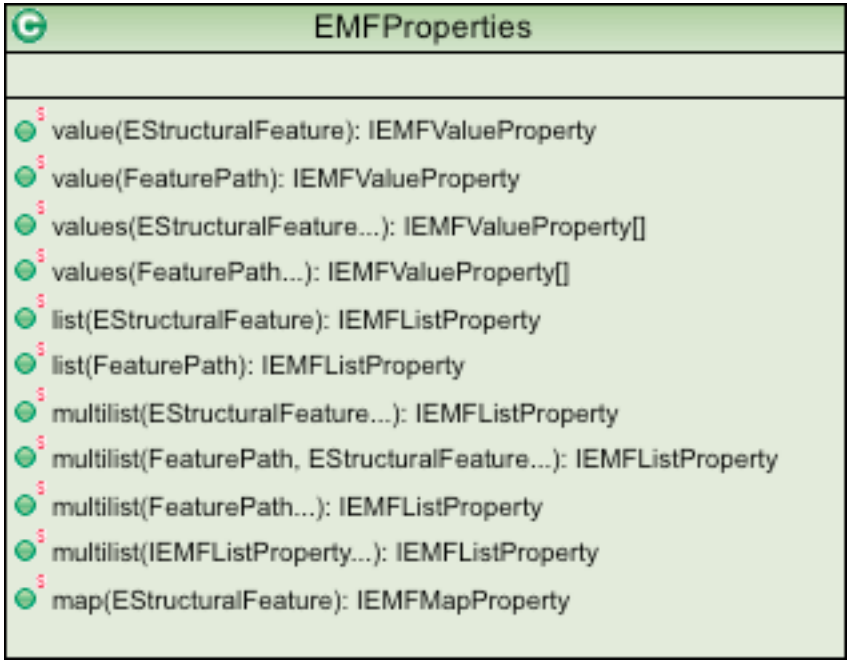
# Háttér

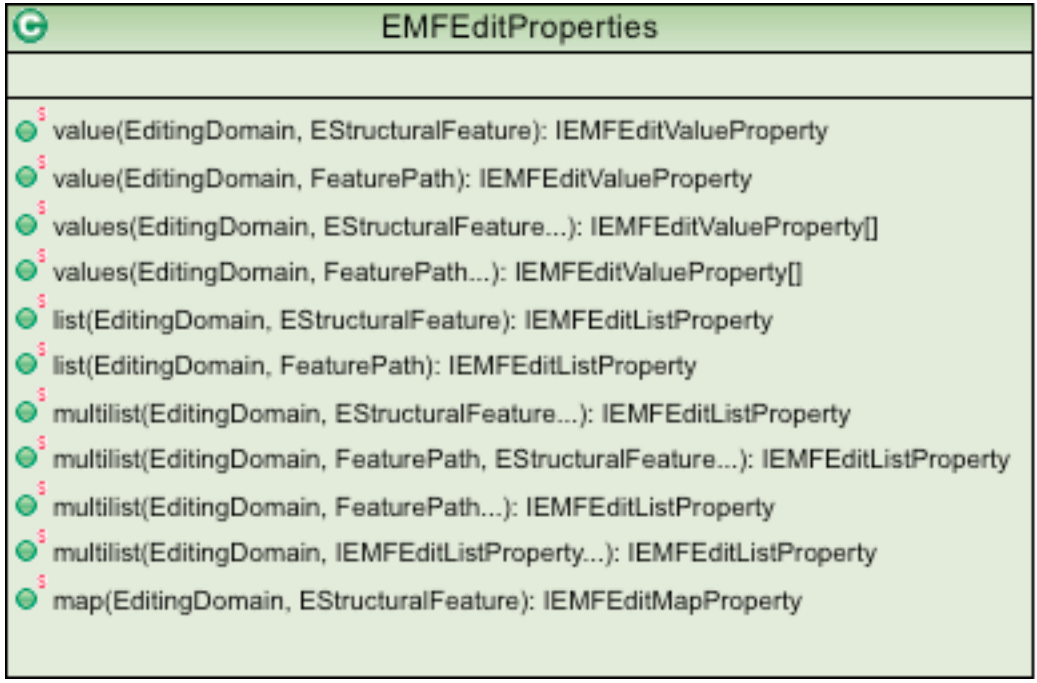
- EMF reflektív API
  - Relációk lekérése típus szerint

```
public String getName(Person person) {
  return (String) person.eGet
(SocialnetworkPackage.Literals.SOCIAL_ENTITY__NAME);
}
```

# Property osztályok előállítása

- Factory osztályok
  - EMFProperties
  - EMFEditProperties




Forrás: http://tomsondev.bestsolution.at/2009/06/07/galileo-emf-databinding-part-2/

# Property osztály példányosítás

```
IEMFValueProperty pName = EMFProperties.value(
SocialnetworkPackage.Literals.SOCIAL_ENTITY__NAME);
```

```
IEMFValueProperty pContactName =
EMFProperties.value(
  FeaturePath.fromList(
SocialnetworkPackage.Literals.ACQUAINTANCE__TARGET,
SocialnetworkPackage.Literals.SOCIAL_ENTITY__NAME))
;
```

# Observable előállítása

- IProperty leszármazottak
  - observe metódus
    - Paraméter: kiinduló objektum
    - Visszatérési érték: IObservable

# IObservable osztály példányosítás

```
IEMFValueProperty pName = EMFProperties.value(
SocialnetworkPackage.Literals.SOCIAL_ENTITY__NAME);
IObservableValue oName = pName.observe(person);

IEMFValueProperty pContactName =
EMFProperties.value(
  FeaturePath.fromList(
SocialnetworkPackage.Literals.ACQUAINTANCE__TARGET,
SocialnetworkPackage.Literals.SOCIAL_ENTITY__NAME))
;
IObservableValue oCName = pContactName.observe
(aquaintance);
```

# IObservable példányosítás – Példák

```
// 1. Use case - observe the subprojects public


IObservableList uc1(Project p) {
  IEMFListProperty prop = EMFProperties.list(
   ProjectPackage.Literals.PROJECT__SUBPROJECTS );
  return prop.observe(p);
}
```

# IObservable példányosítás - Példák

```
// 2. Use case - observe the nested list
// The list of all subprojects of the projects parent
public IObservableList uc2(Project p) {
  IEMFListProperty prop = EMFProperties.list(
    FeaturePath.formList(
      ProjectPackage.Literals.PROJECT__PARENT,
ProjectPackage.Literals.PROJECT__SUBPROJECTS ) );
  return prop.observe(p);
}
```

# IObservable példányosítás - Példák

```
// 3. Use case - Observe a detail list public


IObservableList uc3(IObservableValue master) {
  IEMFListProperty prop = EMFProperties.list(
   ProjectPackage.Literals.PROJECT__SUBPROJECTS );
  return prop.observeDetail(master);
}
```

# IObservable példányosítás - Példák

```
// 4. Use case - Combine to lists into one


public IObservableList uc4(Project p) {
  IEMFListProperty prop = EMFProperties.multiList(
   ProjectPackage.Literals.PROJECT__SUBPROJECTS,
   ProjectPackage.Literals.PROJECT__COMMITTERS );
  return prop.observe(p);
}
```

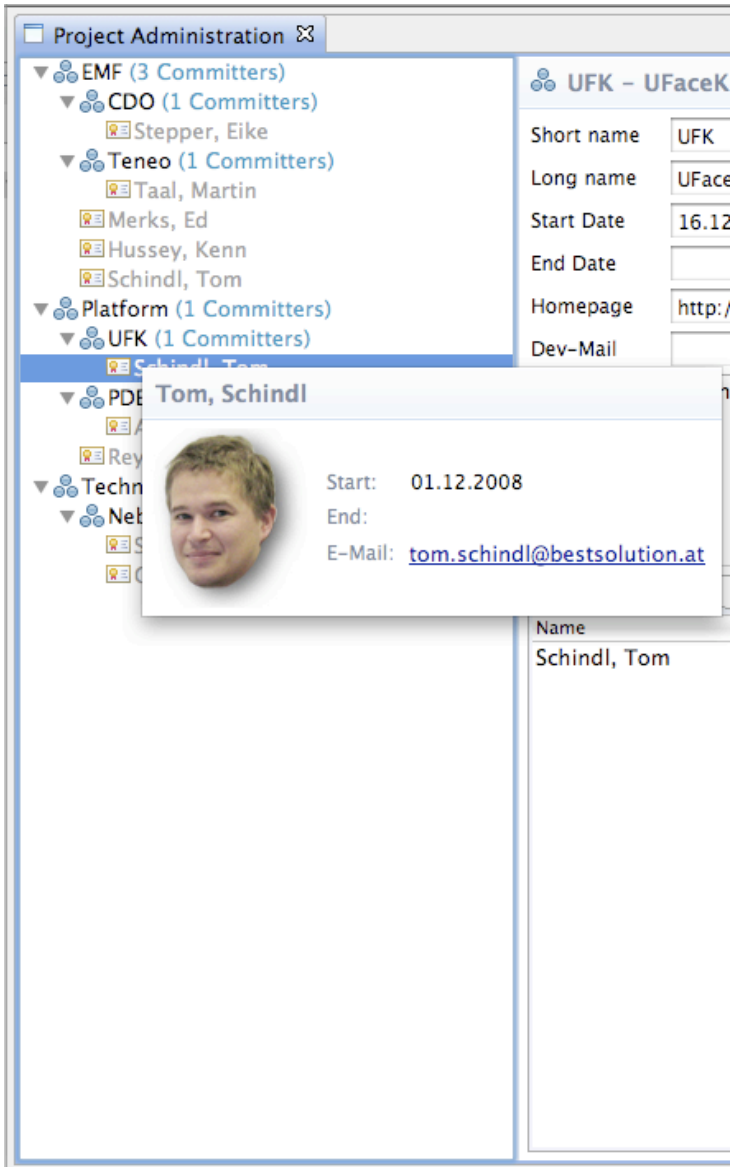
# TreeViewer és EMF

- ContentProvider
  - ObservableListTreeContentProvider
    - Generikus implementáció
    - Kiegészítendő
      - TreeFactory
        - » Gyermekelemek lekérése
        - » IObservableList létrehozás
      - TreeStructureAdvisor
        - » Kiegészítő információk
        - » Szülő
        - » Vannak-e gyermekelemek

# TreeViewer és EMF

- Label Provider
  - CellLabelProvider ajánlott
  - Lehet helyette StyledCellLabelProvider

# Label Provider példa



# Kódrészletek - TreeFactory

```
public class SocialnetworkTreeFactory implements IObservableFactory {

        private IEMFListProperty multi = EMFProperties.multiList(
                SocialnetworkPackage.Literals.COMMUNITY__CHILDREN,
                SocialnetworkPackage.Literals.COMMUNITY__MEMBERS);
        private IEMFListProperty communities = EMFProperties
                .list(SocialnetworkPackage.Literals.SOCIAL_ENTITY__NAME);

    @Override
    public IObservable createObservable(Object target) {
        if (target instanceof IObservableList) {
                return (IObservable) target;
        } else if (target instanceof Community) {
                return multi.observe(target);
        } else if (target instanceof SocialNetwork) {
                return communities.observe(target);
        }
        return null;
    }

}
```

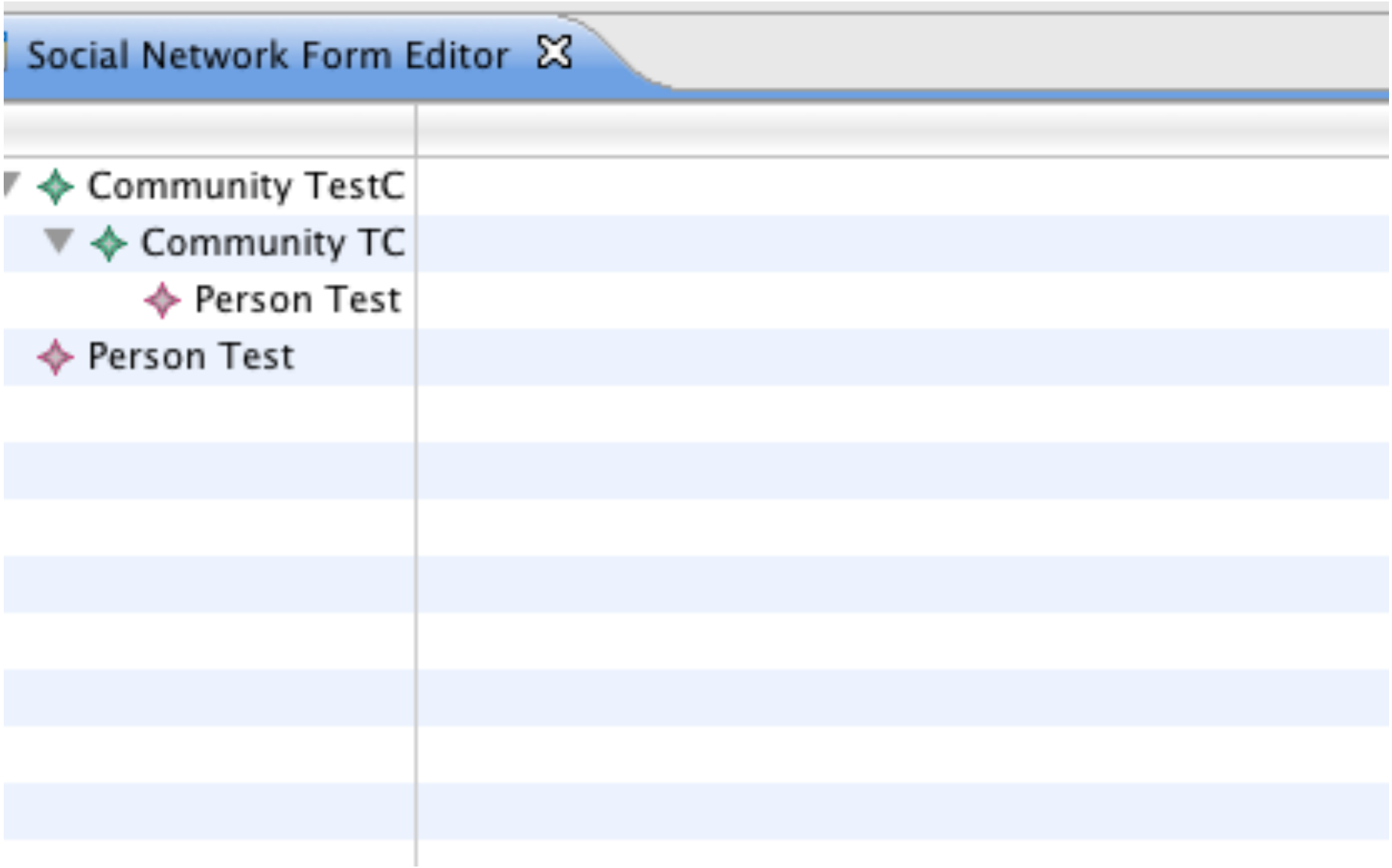
# Kódrészletek - TreeStructureAdvisor

```
public class SocialnetworkStructureAdvisor extends TreeStructureAdvisor {

    @Override
    public Object getParent(Object element) {
         if (element instanceof Community) {
           return ((Community) element).getParent();
         } else if (element instanceof Person) {
           return ((Person) element).getMembership().get(0);
         }
         return null;
    }

    @Override
    public Boolean hasChildren(Object element) {
         if (element instanceof Community
                   && (!((Community) element).getChildren().isEmpty()
                  || !((Community) element).getMembers().isEmpty())) {
            return Boolean.TRUE;
         }
         return super.hasChildren(element);
    }
}
```

# Kódrészletek - Felhasználás

```
// Setting up tree viewer
ObservableListTreeContentProvider cp =
  new ObservableListTreeContentProvider(
      new SocialnetworkTreeFactory(),
      new SocialnetworkStructureAdvisor());
treeViewer.setContentProvider(cp);
// Generic EMF Edit label provider
SocialnetworkItemProviderAdapterFactory factory =
  new SocialnetworkItemProviderAdapterFactory();
treeViewer.setLabelProvider(
  new AdapterFactoryLabelProvider(factory));

IEMFListProperty communities = EMFProperties.list
(SocialnetworkPackage.Literals.SOCIAL_NETWORK__ENTITIES)
;
treeViewer.setInput(communities.observe(network));
```

# A Tree Viewer



# Összegzés

- Egyszerű bejárás gyorsan leírható
  - Deklaratív definíció
- Viszonylag összetett technológia
  - Tanulni nehéz