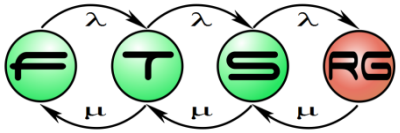
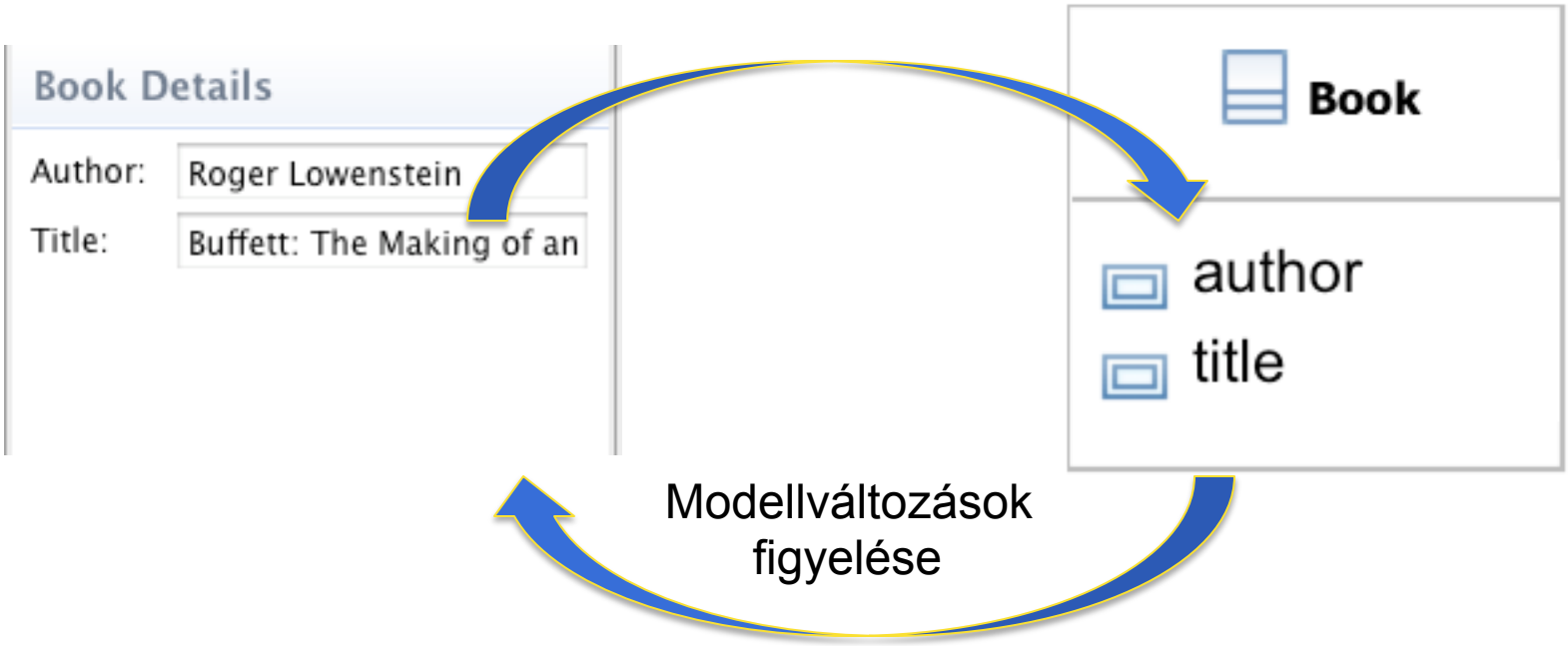
JFace Data Binding

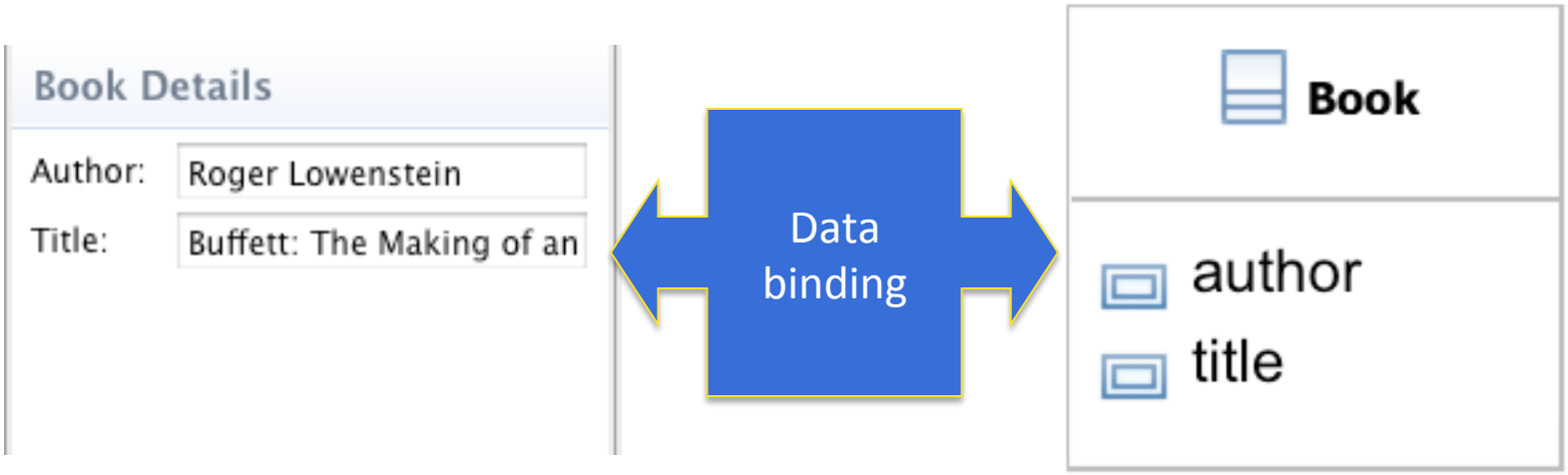


Felhasználói felület és adatmodell



- 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
   (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(pr
  }
  public void removePropertyChangeListener(
   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; }
}
```

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);
```

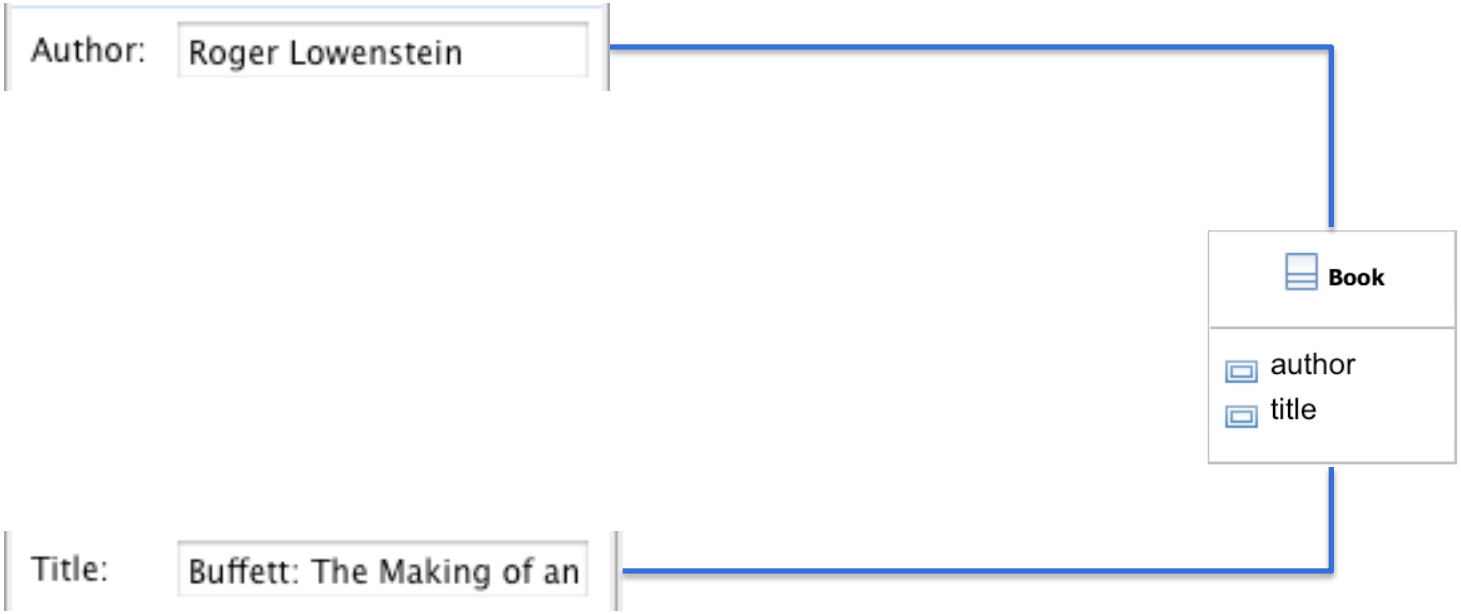
Saját validátor is megadható

```
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

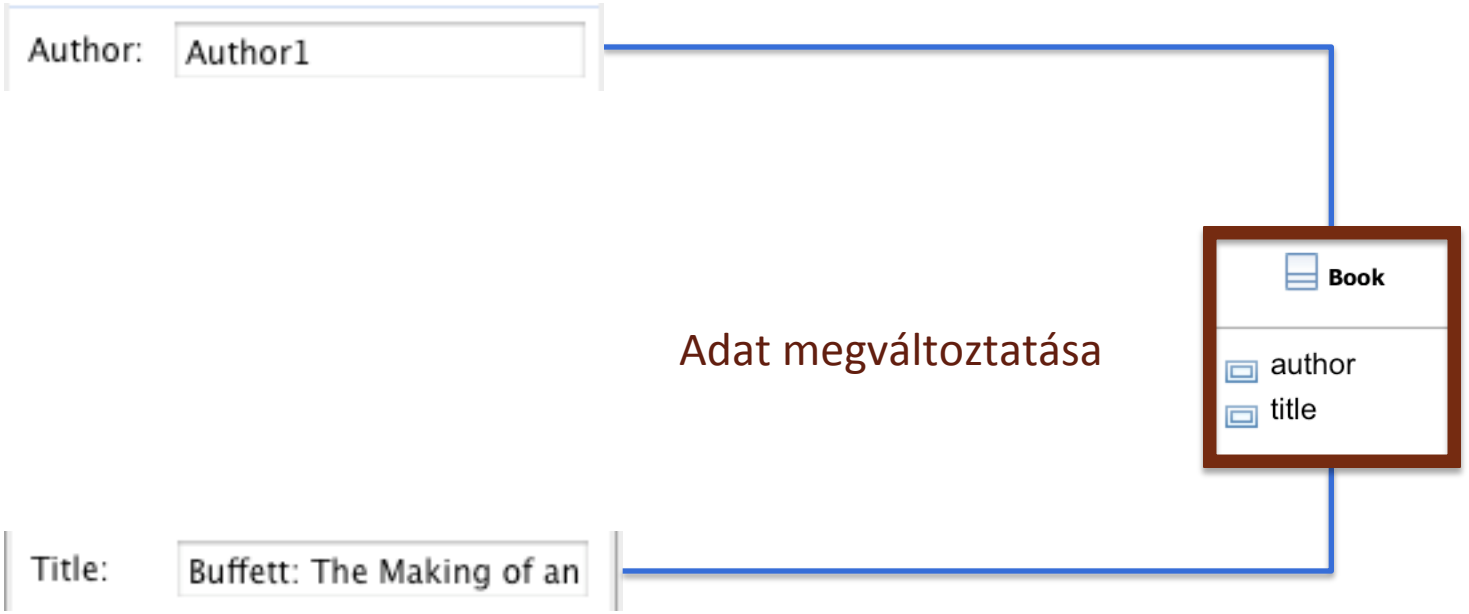
Szerző megváltozott

Adatkötés folyamata - Példa

Adatkötés folyamata - Példa

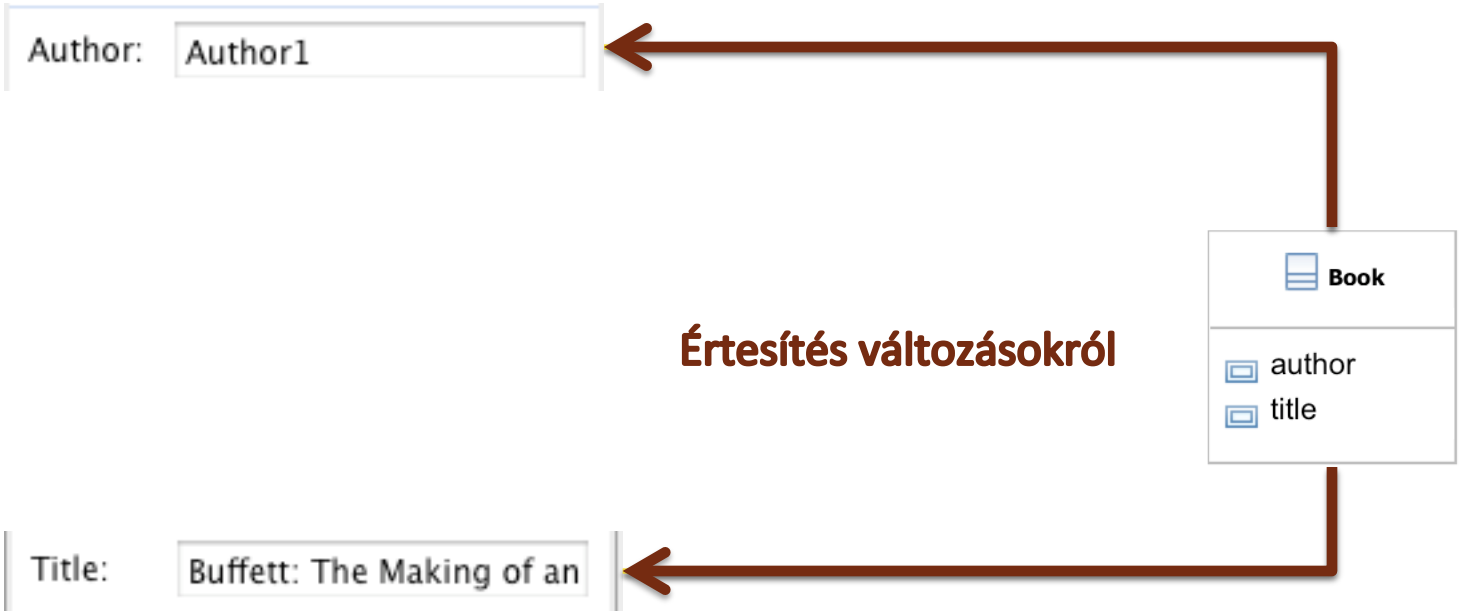
Adatkötés folyamata - Példa

Adatkötés folyamata – Példa

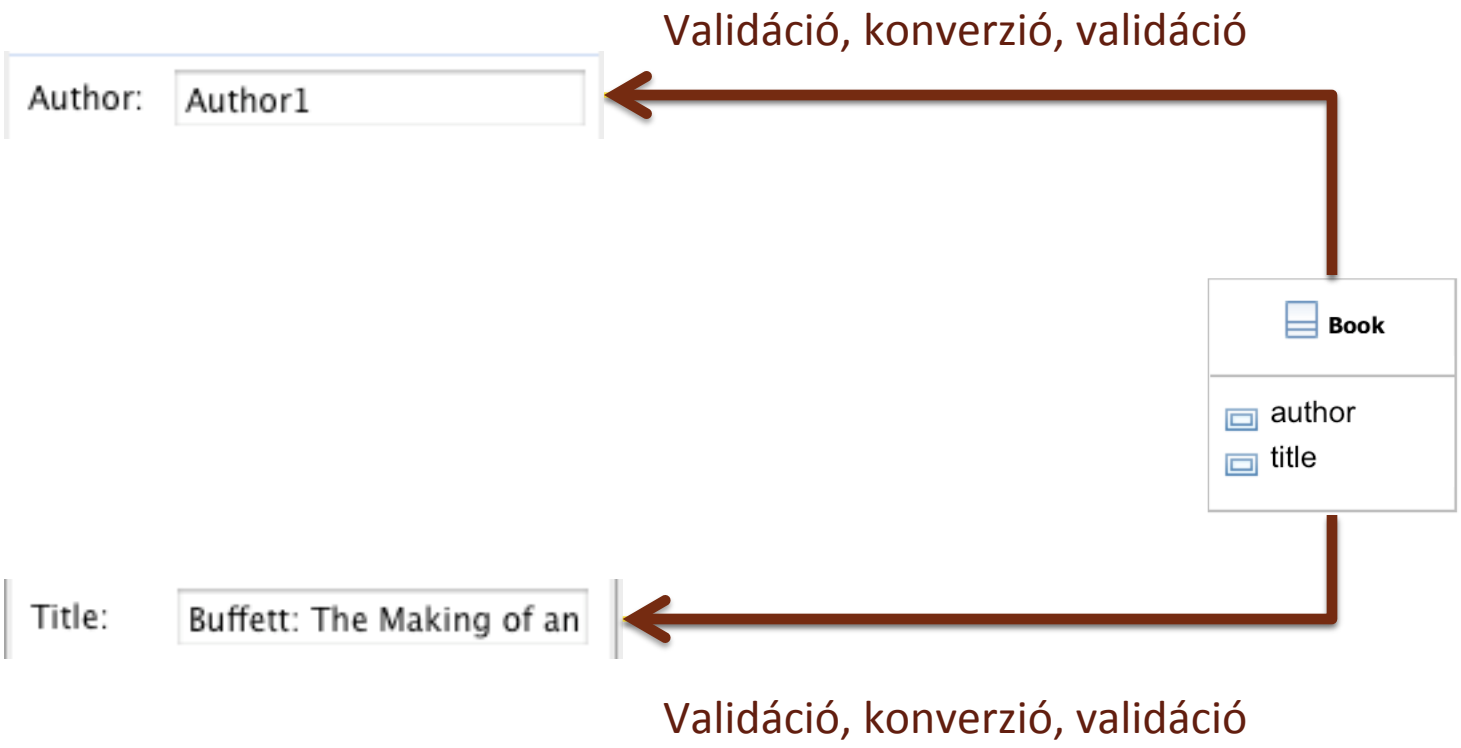


Szerző megváltozott

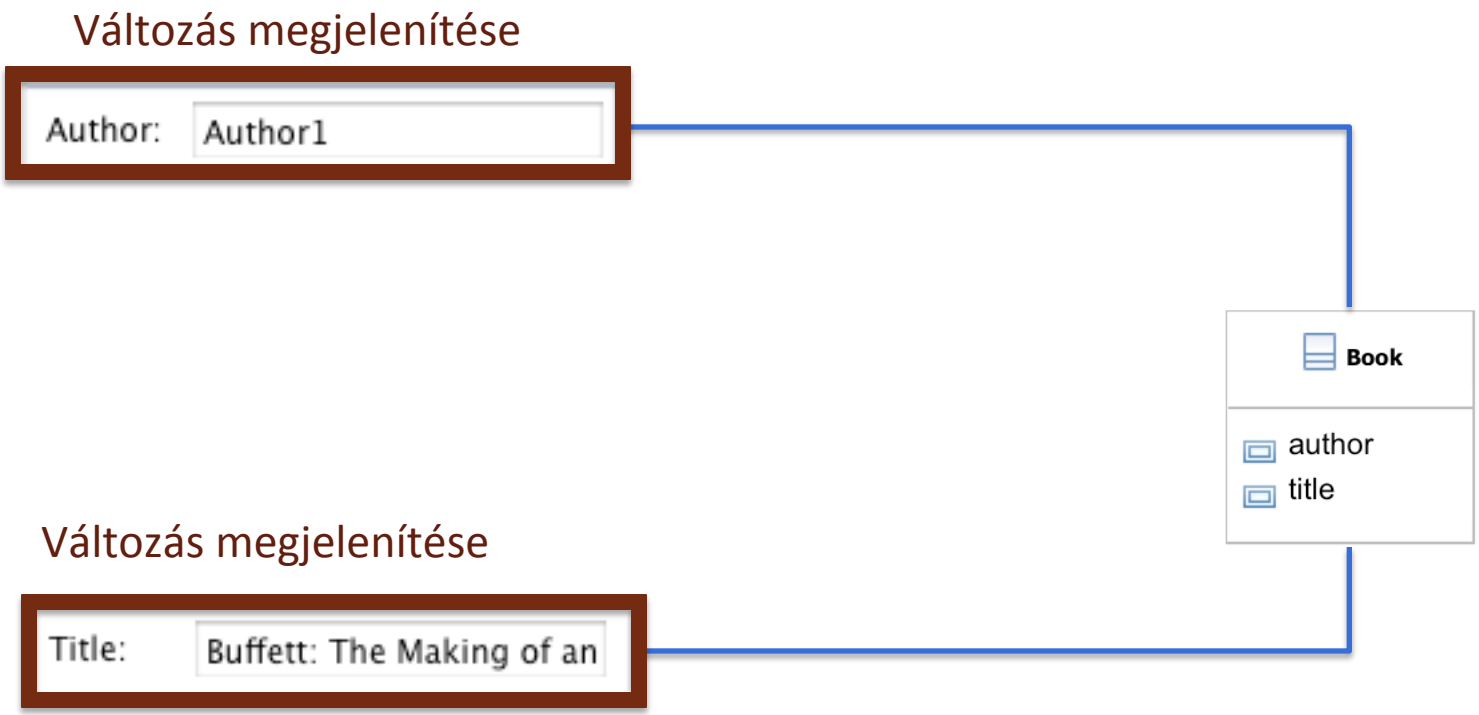
Adatkötés folyamata - Példa



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

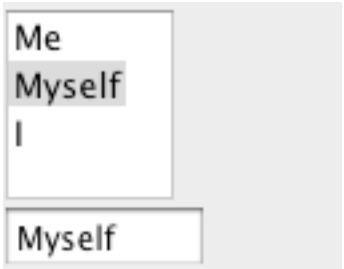
```
final IObservableValue firstName = SWTObservables.observeText
   (firstNameField, SWT.Modify);

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

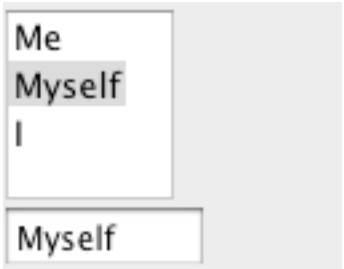
```
// 1. Observe changes in selection.
IObservableValue selection =
   ViewersObservables.observeSingleSelection(viewer);
// 2. Observe the name property of the current selection.
IObservableValue detailObservable =
   BeansObservables .observeDetailValue(selection, "name",
   String.class);
// 3. Bind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

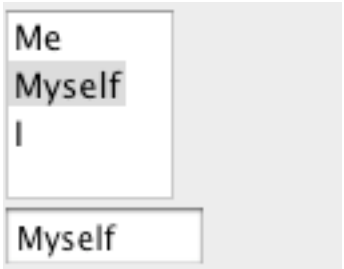
Observer készítése egyszeres kijelöléshez (JFace Viewer)



Master-Detail példa

```
// 1. Observe changes in selection
IObservableValue sele
   ViewersObservables.                lection(viewer);
// 2. Observe the nam                 e current selection.
IObservableValue detailO servable =
   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);
```

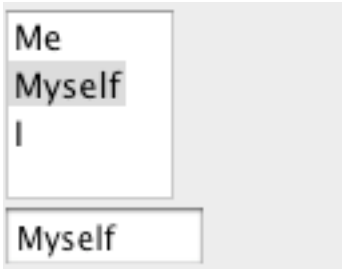
Detail Observable létrehozása



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 nam                tion's
   name).
new DataBindingContext().bindValue(
  SWTObservables.observeText(name, SWT.   one),
  detailObservable,
  new UpdateValueStrategy(false,
    UpdateValueStrategy.POLICY_NEVER),
  null);
```

Adatkötés



Összefoglalás

- Magas szintű adat-GUI szinkronizáció
- Gyorsan fejlődik
  - Van EMF-specifikus API hozzá
- Részletesebb dokumentáció az Eclipse wikin:

❖ **http://wiki.eclipse.org/index.php/JFace_Data_Binding**