

JFace Data binding

JFace permet de mettre en place un système permettant de mettre à jour les interfaces sur modification du modèle, et de mettre à jour le modèle, depuis la modification d'une interface :

-- Crédit à la modélisation

Sur le principe, chaque modèle va émettre un événement sur modification de ses membres, événement communiqué aux listeners associés au modèle.

-- Classe de base AbstractModel

La classe **AbstractModel** va permettre d'ajouter des listeners associés à des membres, et de déclencher des événements. Toute classe intégrée au modèle et utilisant le Data binding devra hériter de AbstractModel.

```
public abstract class AbstractModelObject {  
    private final PropertyChangeSupport propertyChangeSupport = new  
    PropertyChangeSupport(  
        this);  
  
    public void addPropertyChangeListener(PropertyChangeListener listener) {  
        propertyChangeSupport.addPropertyChangeListener(listener);  
    }  
  
    public void addPropertyChangeListener(String propertyName,  
        PropertyChangeListener listener) {  
        propertyChangeSupport.addPropertyChangeListener(propertyName, listener);  
    }  
  
    public void removePropertyChangeListener(PropertyChangeListener listener) {  
        propertyChangeSupport.removePropertyChangeListener(listener);  
    }  
  
    public void removePropertyChangeListener(String propertyName,  
        PropertyChangeListener listener) {  
        propertyChangeSupport.removePropertyChangeListener(propertyName,  
            listener);  
    }  
  
    protected void firePropertyChange(String propertyName, Object oldValue,  
        Object newValue) {  
        propertyChangeSupport.firePropertyChange(propertyName, oldValue,  
            newValue);  
    }  
}
```

-- Crédit à la modélisation

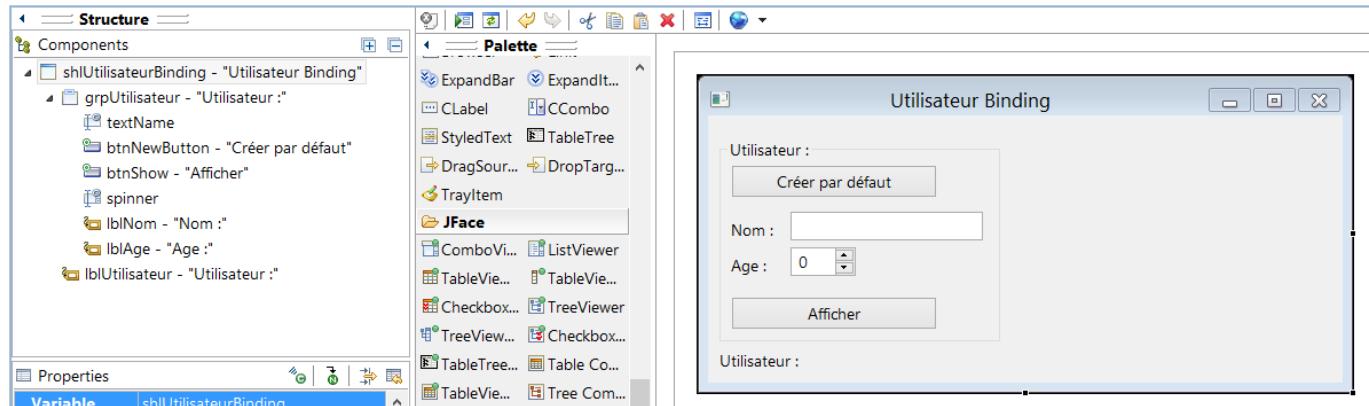
Création d'une classe Utilisateur :

- Utilisateur doit être un Bean (constructeur+getters/setters)
- Hériter de **AbstractModel**
- Appeler **firePropertyChange** sur la modification de ses membres

```
public class Utilisateur extends AbstractModelObject {  
    private String nom;  
    private int age;  
  
    public Utilisateur() {}  
  
    public Utilisateur(String nom, int age) {  
        this.nom = nom;  
        this.age = age;  
    }  
  
    /**  
     * @return the nom  
     */  
    public String getNom() {  
        return nom;  
    }  
  
    /**  
     * @param nom  
     *          the nom to set  
     */  
    public void setNom(String nom) {  
        String oldValue = this.nom;  
        this.nom = nom;  
        firePropertyChange("nom", oldValue, nom);  
    }  
  
    /**  
     * @return the age  
     */  
    public int getAge() {  
        return age;  
    }  
  
    /**  
     * @param age  
     *          the age to set  
     */  
    public void setAge(int age) {  
        int oldValue = this.age;  
        this.age = age;  
        firePropertyChange("age", oldValue, age);  
    }  
}
```

-- Création d'une vue

Créer la classe FormUtilisateur, à partir de l'assistant **SWT/window Application**



-- Implémentation du code

Implémenter les méthodes suivantes, puis tester l'interface

```
btnNewButton.addSelectionListener(new SelectionAdapter() {
    @Override
    public void widgetSelected(SelectionEvent e) {
        user.setNom("DUPONT-LAJOIE");
        user.setAge(25);
    }
});
```

```
btnAfficher.addSelectionListener(new SelectionAdapter() {
    @Override
    public void widgetSelected(SelectionEvent e) {
        lblUtilisateur.setText(user.toString());
    }
});
```

-- Définition du binding

Ajouter les bindings suivants :

Bound Properties:

Target	Model	Target Strategy	Model Strategy
textName.text(SWT.Modify)	user.nom	POLICY_UPDATE	POLICY_UPDATE
spinnerAge.selection	user.age	POLICY_UPDATE	POLICY_UPDATE

Target (Widgets): shlUtilisateurBinding.grpUtilisateur.textName.text

Model (Beans): user.nom

Properties:

The screenshot shows the Eclipse JFace Binding editor interface. At the top, there's a table titled 'Bound Properties' with four columns: Target, Model, Target Strategy, and Model Strategy. Two bindings are listed: one for 'textName.text(SWT.Modify)' mapping to 'user.nom' with 'POLICY_UPDATE' strategy, and another for 'spinnerAge.selection' mapping to 'user.age' with 'POLICY_UPDATE' strategy. Below this are two large panes: 'Target (Widgets)' on the left and 'Model (Beans)' on the right. The 'Target (Widgets)' pane lists various UI components like 'textName', 'btnNewButton', etc. The 'Model (Beans)' pane lists bean properties like 'user - Utilisateur', 'spinnerAge', etc. At the bottom, there are tabs for 'Source', 'Design', and 'Bindings', with 'Bindings' being the active tab.

Code généré :

```

protected DataBindingContext initDataBindings() {
    DataBindingContext bindingContext = new DataBindingContext();
    //
    IObserveValue observeTextTextNameObserveWidget =
WidgetProperties.text(SWT.Modify).observe(textName);
    IObserveValue namePersonObserveValue =
BeanProperties.value("nom").observe(user);
    bindingContext.bindValue(observeTextTextNameObserveWidget,
namePersonObserveValue, null, null);
    //
    IObserveValue observeSelectionSpinnerObserveWidget =
WidgetProperties.selection().observe(spinnerAge);
    IObserveValue ageUserObserveValue =
BeanProperties.value("age").observe(user);
    bindingContext.bindValue(observeSelectionSpinnerObserveWidget,
ageUserObserveValue, null, null);
    //
    return bindingContext;
}

```

Tester l'interface créée et son comportement

-- Binding sur listes d'objets

-- création du model

Implémenter le model Utilisateurs, permettant de gérer une liste d'instances d'Utilisateur :

```
public class Utilisateurs extends AbstractModelObject {
    private final List<Utilisateur> items = new ArrayList<>();

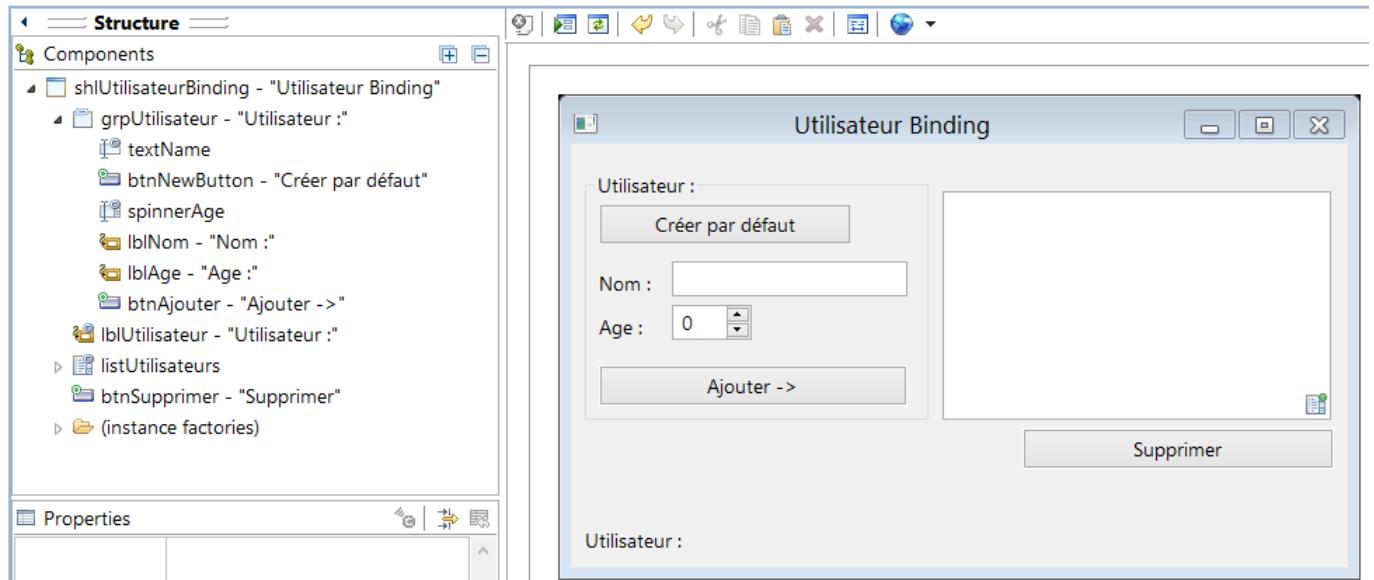
    public void add(Utilisateur utilisateur) {
        items.add(utilisateur);
        firePropertyChange("items", null, items);
    }

    public void remove(Utilisateur utilisateur) {
        items.remove(utilisateur);
        firePropertyChange("items", null, items);
    }

    /**
     * @return the items
     */
    public List<Utilisateur> getItems() {
        return items;
    }
}
```

-- Modification de la vue

Ajouter à la vue les éléments manquants :



-- Implémentation du comportement

From:

<http://slamwiki2.kobject.net/> - Broken SlamWiki 2.0

Permanent link:

<http://slamwiki2.kobject.net/slam4/gui/jfacebinding?rev=1364431289>

Last update: **2019/08/31 14:39**

