

## P2 – JDBC et architecture d'application – eVente (3)

### Quelques composants supplémentaires de recherche et de sélection

*Nous continuons le développement de composants visuels utiles à la couche 'UI' de eVente.*

#### SÉLECTION DANS UNE LISTE

---

Certains objets devront pouvoir être recherchés au travers de critères complexes et l'utilisateur aura à sélectionner un objet parmi un ensemble qui lui sera présenté.

Le composant Swing `JTable` est adapté à ce type de fonctionnalité et nous allons l'utiliser pour nous doter de composants plus spécialisés.

#### MAJTABLE

---

Nous vous fournissons en annexe le code de la classe `MaJTable<T>` étendant `JPanel` et reprenant une `JTable` dans un `JScrollPane`. Nous vous fournissons aussi `MonModele<T>` étendant `AbstractTableModel` nécessaire à `MaJTable`.

Ce `MaJTable<T>`, un peu comme `MaJComboBox<T>`, sera utilisé pour en dériver des composants spécialisés dans l'affichage de certains objets de l'application eVente.

#### MAJTABLECLIENT

---

Ce composant nous permettra d'afficher un ensemble de `ProduitDto` et d'en sélectionner un.

```
public class MaJTableClient extends MaJTable<ClientDto> {
    private Collection<ClientDto> data;
    private String[] titres = {"id", "UId", "Nom", "Prénom", "Mail"};
    private String[] methodes =
        {"getId", "getIdentifiant", "getNom", "getPrenom", "getMail"};
    private int[] largeurs = {15, 40, 200, 100, 300};

    public MaJTableClient() {
        super();
    }

    public MaJTableClient(Collection<ClientDto> data) {
        super();
        setData(data);
    }

    public void setPresentation(
        String[] titres, String[] methodes, int[] largeurs ){
        this.titres=titres;
        this.methodes=methodes;
        this.largeurs=largeurs;
        if (data!=null){
            setData(data);
        }
    }
}
```

```

    }
}

public void setData(Collection<ClientDto> col){
    this.data=col;
    setModel(new MonModele<ClientDto>(col,titres,methodes));
    setColumnWidth(largeurs);
}
}

```

## MISE EN ŒUVRE

Écrivez un `JDialog` nommé **RechercheClient** se présentant de manière analogue à l'exemple ci-dessus et offrant une méthode `public ClientDto getClient()`. Veillez à ce que le bouton 'Sélectionner' ferme le `JDialog` et ne soit actif que si un `ClientDto` est sélectionné dans la `JTable`. Si aucun `ClientDto` n'est sélectionné la méthode `getClient()` retournera `null`.

id	Uid	Nom	Prénom	Mail
1	aaa	Dupont	Charles	a@dupont.be
2	nvc	NvClient	SonPrenom	sonprenom.nvclient@mail.be

Remarques:

- x le double-clic sur un client devrait offrir le même effet que la pression sur le bouton 'Sélectionner'.
- x Le bouton 'Sélectionner' ne devrait être actif que lorsqu'un client de la `JTable` est sélectionné.

## EXERCICE

Réaliser un `JDialog` analogue pour **Produit**.

## AFFICHAGE ET SÉLECTION RAPIDE

---

L'application eVente proposera plusieurs fenêtres nécessitant la sélection de plusieurs objets. Nous allons veiller à concevoir des composants favorisant la recherche la plus souvent sollicitée et prenant un minimum de surface.

### RECHERCHE CLIENT

---

S'il apparaît que les recherches de Client se font majoritairement sur l'acronyme (attribut 'identifiant') nous concevrons un composant '**SelectionClient**' étendant un `JPanel` apparaissant comme ci-dessous:



Un `JTextField` permettra de saisir l'acronyme du client et l'appui de la touche 'Enter' et/ou la perte de focus lancera la recherche et l'affichage de l'élément trouvé (nous aurons à prévoir une représentation en chaîne de caractère courte et significative pour un client). Ce composant devra aussi présenter une méthode `getClient()` et éventuellement une méthode `setClient(ClientDto cli)`.

Si la recherche n'aboutit pas, nous le représenterons de manière claire, par exemple comme suit:



Ce composant offre la recherche la plus souvent réalisée mais doit aussi permettre de rechercher un client au travers d'autres critères. Pour répondre à ce besoin, le bouton 'R' ouvrira '**RechercheClient**'.

Remarques:

- x Veillez à offrir des méthodes classiques d'un tel composant (permettre d'activer-désactiver le composant, ...)
- x Attention, faites en sorte que des recherches ne soient pas lancées inutilement (appui de 'Enter' et ensuite perte de focus, ...)
- x Offrez une méthode `refresh()`

### EXERCICE

---

Développez un composant analogue pour **Produit**.