Guide complet : DesignGridLayout pour Java Swing

🧩 Introduction

DesignGridLayout est une bibliothèque Java Swing conçue pour simplifier la mise en page des composants dans les formulaires et interfaces graphiques. Elle remplace avantageusement `GridBagLayout` ou `MigLayout` en offrant une **API fluide, lisible et intuitive**.

Ce guide complet te montre **tout ce que tu dois savoir**, même si tu débutes totalement.

Installation

1. Télécharge le fichier : `designgridlayout-1.6.jar`
2. Place-le dans un dossier `lib/` à la racine de ton projet.
3. Ajoute-le au **classpath** :

```
javac -cp .:lib/designgridlayout-1.6.jar MonApp.java
java -cp .:lib/designgridlayout-1.6.jar:. MonApp
```

(Sous Linux, séparateur `:` ; sous Windows, `;`)

Importation

Avant d'utiliser DesignGridLayout, importe la classe suivante :

```
import net.java.dev.designgridlayout.DesignGridLayout;
```

Structure de base

Chaque interface s'organise ainsi :

```
JPanel panneau = new JPanel();
DesignGridLayout layout = new DesignGridLayout(panneau);
```

Ensuite, tu ajoutes des **lignes** :

```
layout.row().grid(new JLabel("Nom")).add(new JTextField(15));
```

Une ligne se compose généralement : - d'un **label à gauche** (`grid()`), - et d'un ou plusieurs **composants à droite** (`add()`).

🧩 Les bases des lignes (`row()`)

➤ Une simple ligne avec un label et un champ

```
layout.row().grid(new JLabel("Utilisateur")).add(new JTextField(15));
```

➤ Une ligne avec plusieurs champs

```
layout.row().grid(new JLabel("Nom complet"))
            .add(new JTextField(10))
            .add(new JTextField(10));
```

➤ Une ligne sans label

```
layout.row().center().add(new JButton("Envoyer"));
```

➤ Centrer, aligner ou étendre les composants

```
layout.row().center().add(new JLabel("Titre centré"));
layout.row().left().add(new JButton("Gauche"));
layout.row().right().add(new JButton("Droite"));
```

Largeurs et ajustements

DesignGridLayout aligne automatiquement les colonnes.

Mais tu peux aussi forcer la taille d'un champ :

```
JTextField champ = new JTextField();
champ.setColumns(30); // largeur logique
layout.row().grid(new JLabel("Adresse")).add(champ);
```

🧩 Les méthodes importantes

Méthode	Description
`.row()`	Crée une nouvelle ligne
`.grid(Component)`	Place un label ou un élément de référence (souvent à gauche)
`.add(Component...)`	Ajoute un ou plusieurs composants dans la ligne
`.spanRow(int)`	Fait occuper plusieurs lignes à un composant
`.center()`, `.left()`, `.right()`	Aligne les éléments dans la ligne
`.emptyRow()`	Crée un espace vide entre les lignes

Exemple complet – Formulaire de connexion

```
import javax.swing.*;
import net.java.dev.designgridlayout.DesignGridLayout;

public class FormulaireLogin {
    public static void main(String[] args) {
        JFrame fen = new JFrame("Connexion Oracle");
        fen.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel panneau = new JPanel();
        DesignGridLayout layout = new DesignGridLayout(panneau);

        JTextField champUser = new JTextField("scott", 15);
        JPasswordField champMdp = new JPasswordField("tiger", 15);
        JTextField champUrl = new JTextField("jdbc:oracle:thin:@//localhost:
1521/EE.oracle.docker", 25);

        layout.row().center().add(new JLabel("=== LOGIN ==="));
        layout.row().grid(new JLabel("Utilisateur")).add(champUser);
        layout.row().grid(new JLabel("Mot de passe")).add(champMdp);
        layout.row().grid(new JLabel("URL")).add(champUrl);
        layout.emptyRow(); // espace
        layout.row().center().add(new JButton("Connecter"));

        fen.add(panneau);
        fen.pack();
        fen.setLocationRelativeTo(null);
        fen.setVisible(true);
    }
}
```

Exemple avancé – Formulaire complexe (multi-colonnes)

```
layout.row().grid(new JLabel("Nom"))
            .add(new JTextField(10))
            .add(new JLabel("Prénom"))
            .add(new JTextField(10));

layout.row().grid(new JLabel("Email"))
            .add(new JTextField(25));

layout.row().grid(new JLabel("Adresse"))
            .add(new JTextArea(3, 20));

layout.row().grid(new JLabel("Sexe"))
            .add(new JComboBox<>(new String[]{"Homme", "Femme"}));

layout.row().grid(new JLabel("Langages connus"))
            .add(new JCheckBox("Java"))
            .add(new JCheckBox("C++"))
            .add(new JCheckBox("Python"));

layout.emptyRow();
layout.row().center().add(new JButton("Soumettre"));
```

Ce genre de structure permet de créer des **formulaires professionnels** sans avoir à gérer les contraintes de `GridBagLayout`.

🎨Combiner avec FlatLaf pour un look moderne

Ajoute `flatlaf-<version>.jar` et démarre ton interface ainsi :

```
import com.formdev.flatlaf.FlatLightLaf;
FlatLightLaf.setup(); // Thème clair moderne
```

Tu peux aussi tester : - `FlatDarkLaf.setup();` - `FlatIntelliJLaf.setup();` - `FlatDarculaLaf.setup();`

Bonnes pratiques

1. **Toujours créer un** `JPanel` **par écran ou section.**
2. **Utilise** `emptyRow()` pour espacer visuellement les groupes.
3. **Utilise** `.center()` **pour les boutons** (ex : OK, Annuler).
4. **Évite de mélanger plusieurs** `LayoutManager` dans un même panneau.
5. Combine DesignGridLayout avec **FlatLaf** pour un rendu moderne.

🖌️Conclusion

DesignGridLayout est un outil puissant et minimaliste pour créer des interfaces Swing **propres, lisibles et flexibles**, sans se battre avec la complexité des autres gestionnaires de layout.

Il s'adapte parfaitement à : - des **formulaires dynamiques** (login, inscription, configuration), - des **interfaces professionnelles**, - des **boîtes de dialogue** bien alignées.

Ressources utiles

- **Repo officiel GitHub** : https://github.com/mikaelgrev/designgridlayout
- **JavaDoc (anglais)** : dans `designgridlayout-1.6-javadoc.jar`
- **FlatLaf pour le style** : https://www.formdev.com/flatlaf/

🖋️Auteur : Goldman & GPT-5
📅Dernière mise à jour : Octobre 2025