Rendenrer in JTable

Biểu diễn dữ liệu trên JTable theo yêu cầu về mầu sắc, căn lề

DefaultTableCellRenderer mặc định sử dụng JLabel để render ô của bảng. Để tạo một renderer có tác dụng đổi màu text trong JTable rất đơn giản. Bạn chỉ cần lợi dụng phương thức setForeground của JLabel.

```
1 public static class ColorRenderer extends DefaultTableCellRenderer {
2
3     @Override
4     public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) {
5         setForeground(new Color(153, 0, 0));
6         super.getTableCellRendererComponent(table, value, isSelected,
7                 hasFocus, row, column);
8         return this;
9     }
10}
```

Căn lề cho cột

Bạn có thể căn lề trái, giữa, phải cho các cột trong JTable tùy ý.

```
1 public static class AlignRenderer extends DefaultTableCellRenderer {
2
3     @Override
4     public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) {
5         // align center
6         setHorizontalAlignment(SwingConstants.CENTER);
7         // align left
8         //setHorizontalAlignment(SwingConstants.LEFT);
9         // align right
10        //setHorizontalAlignment(SwingConstants.RIGHT);
11        super.getTableCellRendererComponent(table, value, isSelected,
12                hasFocus, row, column);
13        return this;
14    }
15}
```

Chèn ảnh vào ô của bảng

Lợi dụng khả năng biểu diễn hình ảnh của JLabel thông qua phương thức setIcon. Trong demo, mình sẽ chèn 2 ảnh khác nhau vào cell của JTable dựa vào giá trị được đưa vào là `true` hay `false`.

```
1 public static class ImageRenderer extends DefaultTableCellRenderer {
2
3     @Override
4     public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) {
5         boolean bool = Boolean.parseBoolean(value.toString());
6         Image img = null;
7         if(bool) {
8             img = getToolkit().getImage(getClass().getResource("/images/male.png"));
9         } else {
10            img = getToolkit().getImage(getClass().getResource("/images/female.png"));
11        }
12        setSize(16, 16);
13        setHorizontalAlignment(SwingConstants.CENTER);
14        setIcon(new ImageIcon(img));
15        super.getTableCellRendererComponent(table, "", isSelected,
16                hasFocus, row, column);
17        return this;
18    }
19}
```

Tương tự từ demo này, bạn cũng có thể tạo ra renderer biểu diễn cả text và hình ảnh trên JTable.

Định dạng tiền tệ – Currency

```
1 public static class CurrencyRenderer extends DefaultTableCellRenderer {
2
3     @Override
4     public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) {
5         if ((value != null) && (value instanceof Number)) {
```

```
            Number numberValue = (Number) value;
            NumberFormat formater = NumberFormat.getCurrencyInstance();
            value = formater.format(numberValue.doubleValue());
        }
        setHorizontalAlignment(javax.<a title="swing" href="http://codeimba.com/tag/swing/">swing</a>.SwingConstants.RIGHT);
        super.getTableCellRendererComponent(table, value, isSelected,
                hasFocus, row, column);
        return this;
    }
}
```

Hoặc bạn làm như sau:

```
public static class CurrencyRenderer extends DefaultTableCellRenderer {

    public CurrencyRenderer() {
        super();
        setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
    }

    @Override
    public void setValue(Object value) {
        if ((value != null) && (value instanceof Number)) {
            Number numberValue = (Number) value;
            NumberFormat formater = NumberFormat.getCurrencyInstance();
            value = formater.format(numberValue.doubleValue());
        }
        super.setValue(value);
    }
}
```

Hai cách viết này tương đương nhau. Bản chất của phương thức setValue của DefaultTableCellRenderer là gọi đến phương thức setText được kế thừa từ JLabel.

Output một giá trị khác với giá trị đầu vào

Các bạn tránh nhầm lẫn nhé. Giá trị mà renderer vẽ ra hoàn toàn không có ảnh hưởng gì đến giá trị đầu vào. Nó chỉ có hiệu quả thị giác mà thôi. Bạn có thể hình dung rằng renderer chỉ vẽ ra một tấm mặt nạ để che đậy khuôn mặt bên dưới vậy. Một cell có giá trị đầu vào là "CodeBlue" nhưng được render ra hiển thị là "Mr.CodeBlue". Khi bạn lấy giá trị của cell này (thông qua getValueAt của JTable chẳng hạn), thì giá trị bạn lấy được vẫn là "CodeBlue".

```
public static class OutputRenderer extends DefaultTableCellRenderer {

    @Override
    public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) {
        value = "Mr." + value;
        super.getTableCellRendererComponent(table, value, isSelected,
                hasFocus, row, column);
        return this;
    }
}
```

Đến đây, bạn có thể thấy rằng tất cả các renderer mà chúng ta vừa tạo đều được kế thừa từ DefaultTableCellRenderer. Và trên thực tế, đây cũng là cách phổ biến nhất để tạo ra renderer cho JTable. Như các bạn biết, DefaultTableCellRenderer được kế thừa từ JLabel. Chúng ta có thể rút ra một kinh nghiệm từ đây:

Khi bạn muốn JTable được hiển thị theo một cách nào đó, bạn hãy nghĩ xem cách đó có thể được biểu diễn bởi JLabel hay không? Nếu có thể, bạn chỉ cần viết các phương thức tương ứng của JLabel trong khi override lại getTableCellRendererComponent của DefaultTableCellRenderer. Nó sẽ trả về tham chiếu đến JLabel mà bạn đã sửa đổi để đạt được sự hiển thị như mong muốn.

Vậy trường hợp nếu bạn muốn một cột trong JTable được hiển thị theo cách mà không thể được biểu diễn bằng JLabel thì sao? Chúng ta cũng biết JLabel có những giới hạn. Chẳng hạn nó không thể tạo ra màu background được. Trong trường hợp này, bạn có thể trả về cho phương thức getTableCellRendererComponent một component khác mà có khả năng biểu diễn được yêu cầu của bạn, hơn là trả về một JLabel như bình thường.

Tạo màu nền cho các ô trong bảng

Ví dụ mình tạo ra một renderer để render ra các ô có màu nền màu đỏ, text màu xanh, căn lề giữa:

```
public static class BgRenderer extends DefaultTableCellRenderer {

    @Override
    public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) {
        JPanel pn = new JPanel(new BorderLayout());
        JLabel lb = new JLabel();
        pn.add(lb, BorderLayout.CENTER);
        lb.setHorizontalAlignment(SwingConstants.CENTER);
        lb.setText(value.toString());
        lb.setForeground(Color.BLUE);
        pn.setBackground(Color.RED);

```

```
        super.getTableCellRendererComponent(table, value, isSelected,
                hasFocus, row, column);
        return pn;
    }
}
```

Sử dụng các renderer đã được tạo

Chúng ta đã có renderer, vậy làm thế nào để sử dụng chúng? Trong JTable, bạn có 2 cách thường dùng sau:

Sử dụng renderer cho từng cột

Bạn xác định renderer cho từng cột bằng cách lấy về cột theo chỉ số của nó:

```
myTable.getColumnModel().getColumn(0).setCellRenderer(new MyTableRenderer.OutputRenderer());
myTable.getColumnModel().getColumn(1).setCellRenderer(new MyTableRenderer.BgRenderer());
```

Sử dụng renderer cho từng kiểu dữ liệu

Áp dụng renderer cho tất cả các cột có cùng kiểu dữ liệu:

```
myTable.setDefaultRenderer(Number.class, new MyTableRenderer.CurrencyRenderer());
myTable.setDefaultRenderer(Boolean.class, new MyTableRenderer.ImageRenderer());
```