

# 15

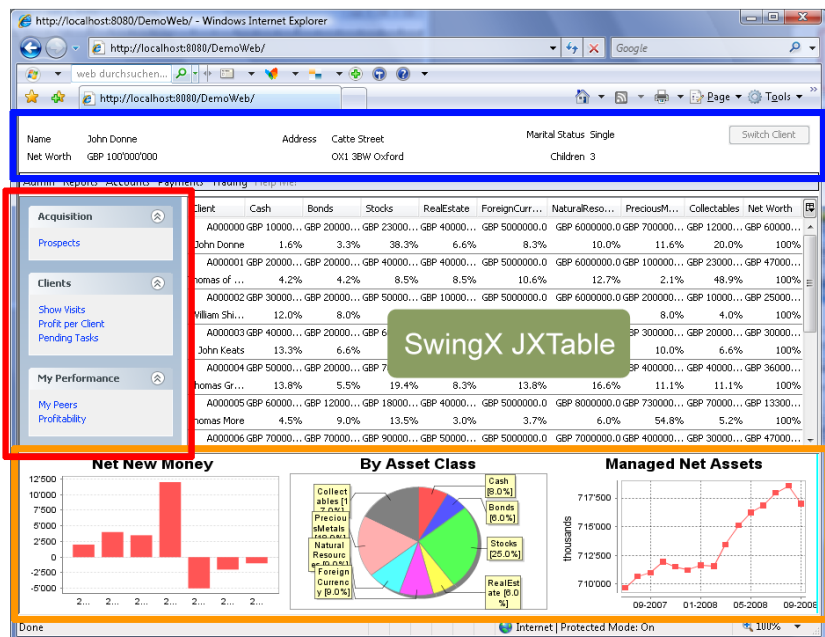
## Frontend Demo

In the previous chapter we have used server side technologies to simplify the backend. It has illustrated that the technologies used are not limited by Riatrix4Applet: We can use just any libraries and frameworks to simplify the server side programming. This chapter illustrates that this is the case for the GUI as well.

L&F customization

SwingX taskpane

Interactive  
JFreeChart  
controls



Most chapters' examples have trivial front ends. They illustrate a single, specific aspect of the possibilities you have with Riatrix4Applet. However, Riatrix4Applet is intended for really rich front ends - much richer than what you can get by using pre-fabricated AJAX controls for instance. In this chapter, too, we try and keep the example simple - so that it can be used as a

pattern of what can be done and is still small enough not to be puzzling. From a conceptual point of view, it is not much different from the initial examples in the first chapters, so we will just focus on the differences.

The DemoWeb project is a GUI as it could be used by wealth management professionals to manage their client relations. It consists of a header where basic information on the currently active client is displayed and a menu bar that allows the selection of different screens. We have implemented just two screens for the sake of illustration: one to enter payments (shortcut F3) and an overview screen that displays information on all clients managed by the client advisor shown above (shortcut F2).

The screenshot illustrates the following points:

## 15.1 L&F Customization

The background for the header is white: This is achieved by customizing the look and feel via `UIDefaults`:

```
class LookAndFeelHelper {
    public static void adjustLookAndFeel() {
        UIDefaults def = UIManager.getLookAndFeelDefaults();
        def.put("Panel.background", new ColorUIResource(Color.white));
    }
}
```

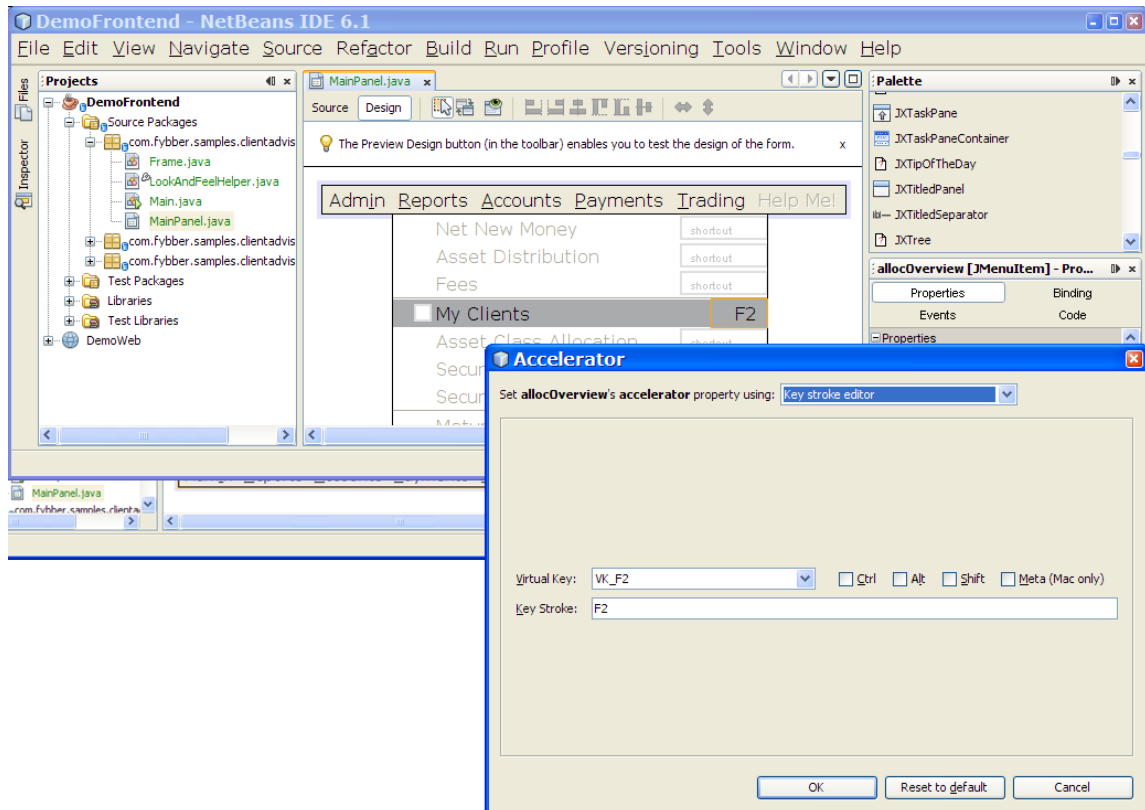
The relevant method that has to be overridden in `RiatrixApplet` or `RiatrixFrame` is called `setLookAndFeel`:

```
public class Frame extends RiatrixFrame{
    @Override
    protected void setLookAndFeel() {
        super.setLookAndFeel();
        LookAndFeelHelper.adjustLookAndFeel();
    }
}
```



## 15.2 Standard Menu Bar

The toolbar is set up using the standard Netbeans GUI builder. Accelerators can be entered using the respective dialog window. There is no need to programmatically handle accelerator keys.

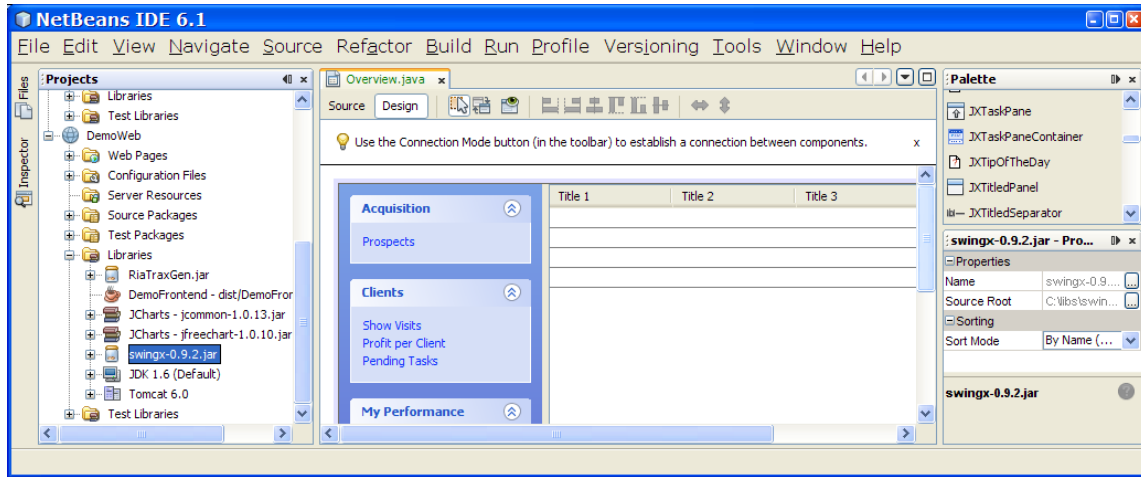


Accelerator keys provide an additional value to a rich interface and allows power users to quickly navigate through the application, saving time and costs. Moreover, many users are already used to accelerator keys from other desktop applications, e.g. when saving a Word document using Ctrl + S.



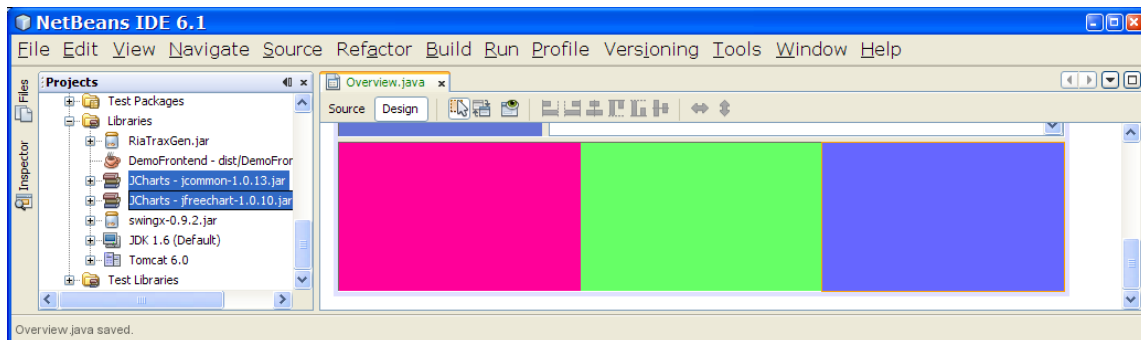
## 15.3 SwingX and JFreeChart Controls

SwingX is a library with Swing extension controls. They can be used as-is to for the design of the front end -Ū as long as the respective libraries are added to the server classpath:



SwingX controls can be added to the GUI builder palette as shown above.

For the JFreeChart also visible on the overview page, we have just added JPanel containers because we are doing some customization on the controls inside the code:



## 15.4 Payments Screen

The payments screen shown below illustrates some of the possibilities that come with Java SE and associated libraries to build form oriented screens.

The screen illustrates the following points:

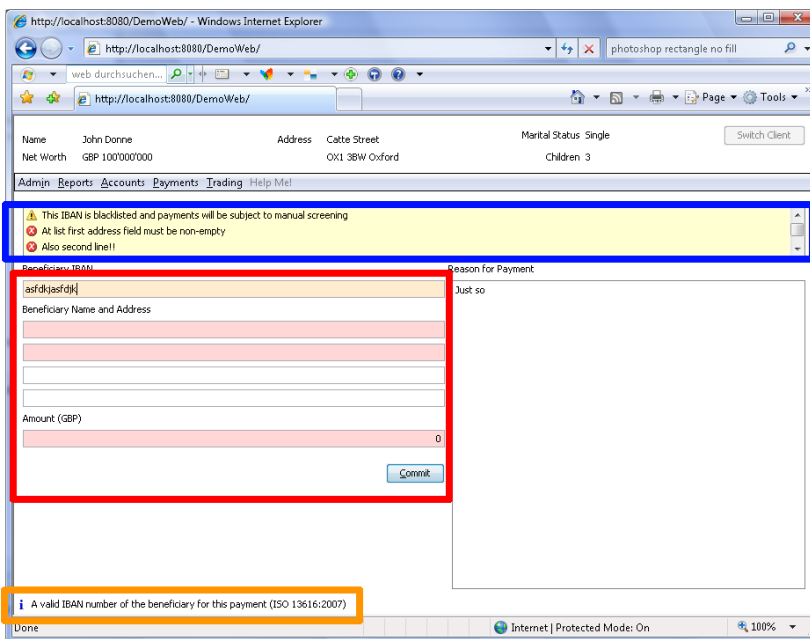
- At the top of the screen is a validation report listŪ as provided by the JGoodies Validation Library. The icons and the handling as well as coloring come with the library and can be customized. The library provides classes that help keep track of the fields that still need valid user entry (these are the rose fields) or have warnings associated.
- Validation is done with a short delay (600ms in the example) and includes a backend call that checks if the IBAN number is blacklisted -Ū the result of that call is cached.



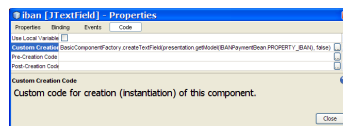
**JGoodies Validation**

**JGoodies Binding**

**JGoodies Validation**



- The data binding between the domain object (a payments request) and the fields on the front end is done by JGoodies Binding. We are not using beans bindings (JSR 295), because it is less mature and does not work as reliably with all security managers. However, JGoodies Binding requires the use of factory methods for the creation of elements:



- At the bottom of the screen is an information banner that displays additional information for the fields. This is managed using JGoodies Validation functionality, too.
- The amount field is a JFormattedTextField for long numbers. The example uses the standard JFormattedTextField out-of-the box. This, of course, may or may not be appropriate depending on corporate style guides and policies and is easy to adjust.
- Pressing enter will revalidate the screen and either submit it or place the cursor in the first whose input is invalid. This allows the user to navigate through the required fields using the enter key.
- Submitting the screen will replace the current input form with a successor screen that depends on the output of the backend service (either displaying `Success` or the error message).



## 15.5 Conclusion

By just adding 3rd party libraries to the server classpath, we can leverage the extensive Swing GUI controls and helper libraries. They add out-of-the-box interactivity to the frontend that is difficult to achieve with other technologies.

This is illustrated in this chapter using three libraries that are freely available: SwingX, JFreeChart and JGoodies Validation and Binding. There is nothing that can stop you from using more complex components like the Eclipse BIRT engine to directly produce reports on the client side therefore taking a significant burden of the server infrastructure.

