SDSJavaFXControls

Version 1.0.0 4th July 2012

Introduction

One of the big issues I faced when writing a Swing Application a few years back was getting a simple but flexible set of input controls that restricted users ability to enter data that was out of context. My need went beyond the usual suspects such as date fields – my application was financial and so need to allow user to enter valid numbers or account codes formatted as per the application's configuration screens. This led to the design of a set of controls for Swing applications.

Earlier in 2012 year I decided to explore Java FX and recreate my Swing Application as an FX app. The port went very well until I needed set of input fields. I found the platform lacking and have, therefore, ported the controls created for Swing into Java FX. This may not be a *perfect* piece of FX code; at the time of writing this set of notes it can be regarded as Beta quality and my first major work in FX.

The components have been developed and tested against Java FX 2.1 as bundled with JDK 1.7.04.

Getting Started

The distribution includes a sample application (which also forms part of the unit test suite) com.stddata.fx.uitest.FXUiControlDemo.java. This application and it's associated stage contain examples of various field configurations.

Masks

The controls are built around standard FX Textfield components. In essence input is control by edit masks that restrict input.

- Fields are always returned with guaranteed length of data – for example a char 10 field will return an empty 10 character string.
- Each position in the string can be formatted to a particular input. This can be numeric only, any character, forced uppercase or forced lowercase.
- Fields can contain literals – fixed (e.g. the – between month and year for a date) or optional literals that only appear when text adjoins the literal.
- Dedicated numeric input formatting (ideal for financial applications). Signs for negative numbers are optional, but where required can be set to a leading or trailing '-' or '()' characters. Groups of numbers can be separated by a character of the developers choosing (most commonly a ',' in the UK). Edit masks can be configured just before rendering at runtime so allowing dynamic support for international standards.
- Date input options include a choice between short numeric or short alpha numeric inputs. Setting a java Locale object will allow international formats to be supported. The software

will work most Locales, however, Japanese and Chinese Locale have not been tested and are unlikely to work. When leaving the field, the date switches from numeric input to an alpha numeric display format. Date fields also benefit from a simple pop up calendar chooser activated by pressing the <F4> key or button next to the date field. Only valid dates are allowed on input.

- Time fields are can also be created and validated.
- All fields (apart from those containing dates) can be configured to call a user defined pop up window by pressing the <F4> key or clicking the button that appears next to the field.

Specification

Edit masks are created from a Specification object. Once created a specification can be used by multiple fields that require the same input controls. For example if a form has several date fields, then the date specification only need be created once.

```
EditMaskSpecification moneys = new EditMaskSpecification
(18,2,".",",",EditMaskFactory.NEG_FMT_BRACKETS);
```

To be valid for use, the Specification object must have it's mask created by the EditMaskFactory.

```
EditMaskFactory.getMask(moneys);
```

The specification object has various constructors to allow various types of masks to be created. You can use the EditMaskSpecification (int maskLength, char maskChar) constructor to create simple masks of a particular length. The MaskChar must be one of the static values defined in the EditMaskFactory class.

Sophisticated custom masks can be created via the EditMaskSpecification (String editMask) construct. This examples creates an edit mask suitable for Intrastat codes.

```
StringBuilder mask = new StringBuilder ();

mask.append("DDDD").append(EditMaskFactory.ESCAPE).append("
").append("DD").append(EditMaskFactory.ESCAPE).append(" ").append("DD");

try

{

        EditMaskSpecification intrastats = new EditMaskSpecification (editMask);

}

catch (Exception E)

{

}
```

TextMaskField

The TextMaskField is a wrapper for a TextField and optional look up button. The TextMasks with a Date based specification automatically receive a look up button which when invoked calls the supplied pop up calendar.

If you have your own look up window (typically associated with picking codes etc from database lookup tables) you need to create an action listener for the text field. The maskLookup field in the FXUiControlDemo application shows an example.

Collecting Input Values

When the user indicates that have completed work on the form, you can obtain the input value from the getValue () method of TextMaskField. The software attempts to return an object that matches the input value (if any) set when the user gained focus on the field. A default value can be explicitly set via the setValue() method.

Unit Tests

The software is supplied with a set of Unit Tests. These are based on JUnit and for the GUI Components Jemmy and JemmyFX. I have changed the behaviour of JemmyFX to work FX2.1. You should therefore use the supplied JemmyFX jar to test the project. The distribution also includes the source code for the alterations to JemmyFX.

-End-