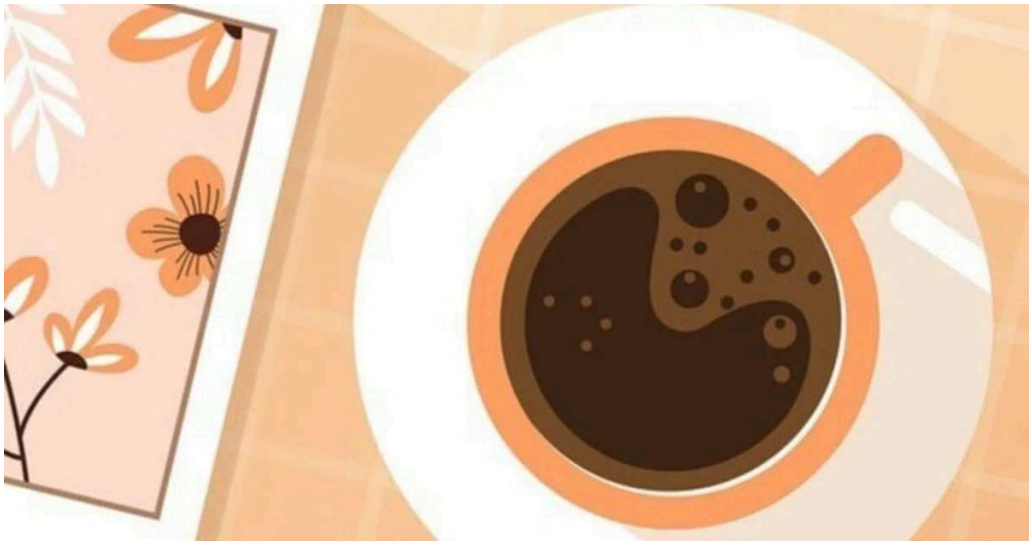

Swingによるデスクトップアプリケーション開発（カレンダー）-JavaSE1.8

office・M
2024年9月1日 20:01

Java8のSwing環境でデスクトップアプリケーションの開発方法を学ぶ講座をシリーズで提供しています。今回はWindowBuilder（Swingデザイナー）を使ってカレンダーアプリケーションを作成します。カレンダーの計算ロジックはWebアプリケーション開発で利用した独自クラスを再利用します。

2024年9月よりECLIPSEのバージョンを最新版（Version: 2024-06 (4.32.0)）に変更しました。

目次

外部設計
GUI設計のポイント
内部設計
処理ロジック
提供クラス
実装準備
プロジェクトの作成
外部classの利用準備

ビルド・パスに追加
実装
ひな形の作成
GUI実装
イベント実装
実装変更
単独起動
実行可能JARファイル
最後に

外部設計

WindowBuilderのSwingデザイナーで図1のようなGUIを作成します。

図1. CalendarアプリケーションのGUI画面

GUI設計のポイント

カレンダーなので土日、祝祭日は青や赤に日付の色を変更する必要がありますが、JTextAreaはPLAINテキストでしか文字を表示できません。
つまり黒しか表示できません。

テキストの部分的な文字の色等を変更したい場合は**JEditorPane**を使います。プロパティの**contentTypeを text/html に設定**することでhtmlによる文字飾りが利用できるようになります。

今回はJEditorPaneを用いて<font>タグのcolor属性で文字色を変更します。

内部設計

カレンダーの日付計算と表示を行う外部クラス（CalendarHtmlBean.class）を『MVCモデルによるWebアプリケーション開発（No４.カレンダー）-EE８』の記事より取得してください。Eclipseのビルドパスに追加することで、当該クラスを利用できるようになります。

処理ロジック

作成ボタンをクリックすることでアクションイベントが発生し、外部クラス（CalendarHtmlBean.class）が実体化されることで、任意の年月のカレンダーが生成されるよう設計します。

提供クラス

提供されるカレンダー作成ロジック（CalendarHtmlBean.class）の使い方とクラス図は以下のようになります。

（使い方）
・CalendarHtmlBeanクラスを**年**と**月**の２つの引数をもつコンストラクタで実体化します。
・createメソッドを実行することでdayフィールドに対象年月のカレンダーがhtml文字列で保持されます。
・利用側のクラスはgetDay()メソッドでカレンダーの内容を取得します。

（クラス図）

package:jp.ict.aso.model CalendarHtmlBean	
- year : int	//作成年
- month : int	//作成月
- day : String	//対象年月カレンダー（html形式）
+ CalendarHtmlBean()	//コンストラクタ
+ CalendarHtmlBean(year: int, month: int)	//コンストラクタ
+ create() : void	//カレンダー作成
+ getDay() : String	//カレンダー取得
+ getYear() : int	//作成年取得
+ getMonth() : int	//作成月取得
+ week_of_day(y: int, m:int, d:int) : int	//曜日計算
+ month_last_day(y: int, m: int) : int	//月末日計算
+ is_leap_year(y: int) : boolean	//うるう年判定
+ is_holiday(m: int, d: int) : boolean	//休日判定

-:private　+:public　#:protected

図2．CalendarHtmlBeanクラス図

実装準備

プロジェクトの作成

Eclipseのメニューバーより
ファイル→新規→Javaプロジェクト→「SwingCalendar」プロジェクトを作成する → 図3の内容で設定する

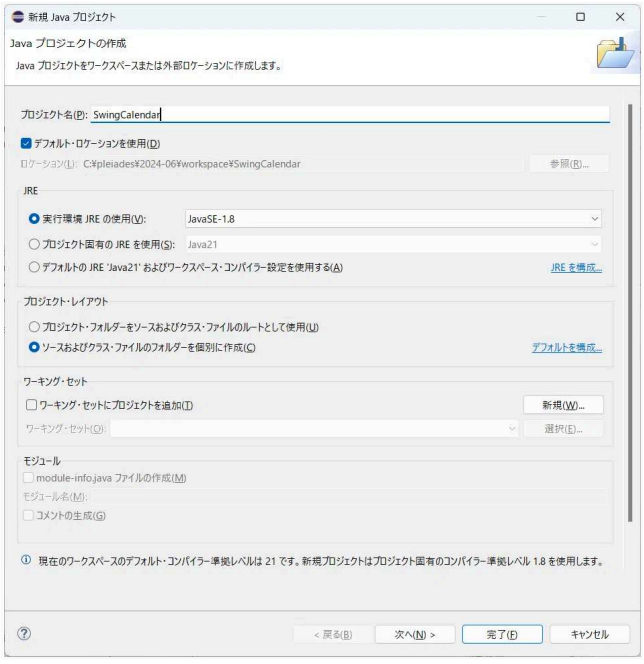

図3．SwingCalendarプロジェクト設定

※作成済みであればこの処理は必要ありません。

以下画面のスクリーンショットはライトテーマで取得します。
（ライトテーマの設定方法）
Eclipseのメニューバーより
　ウィンドウ → 設定 → 一般 → 外観 → ルック＆フィール → ライト
　→ 適用して閉じる → Eclipseの再起動がかかります

外部classの利用準備

以下Windows環境を想定しています。
事前に「提供Beanフォルダ」を作成しておきます（例　c:￥提供Bean）。
提供Beanフォルダ内に**CalendarHtmlBean.class**を保存しておきます。
その際**パッケージの階層**に従ってください。
（例　c:￥提供Bean￥jp￥ict￥aso￥model￥CalendarHtmlBean.class）

ビルド・パスに追加

CalendarHtmlBean.classをEclipseのビルド・パスに追加します。

Eclipseパッケージ・エクスプローラより
プロジェクトを右クリック→プロパティ→「Javaのビルド・パス」を選択
→「ライブラリ」タブ→「外部クラス・フォルダーの追加」ボタンクリック
→「提供Bean」を選択→最後に「適用して閉じる」をクリック

※図4のように一度設定されていれば再度設定する必要はありません

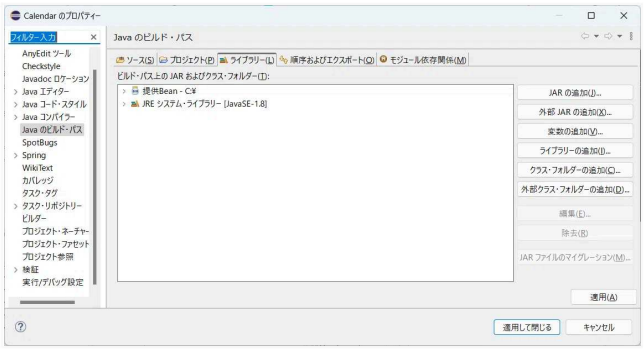

図4．外部クラス・フォルダの追加

実装

ひな形の作成

WindowBuilderを用いてSwingアプリケーションのスケルトン（骨格）を自動生成させます。

Eclipseパッケージ・エクスプローラより
SwingCalendarプロジェクトを右クリック→新規→その他 → WindowBuilder
→ Swingデザイナー → JFrameを選択 → 次へ

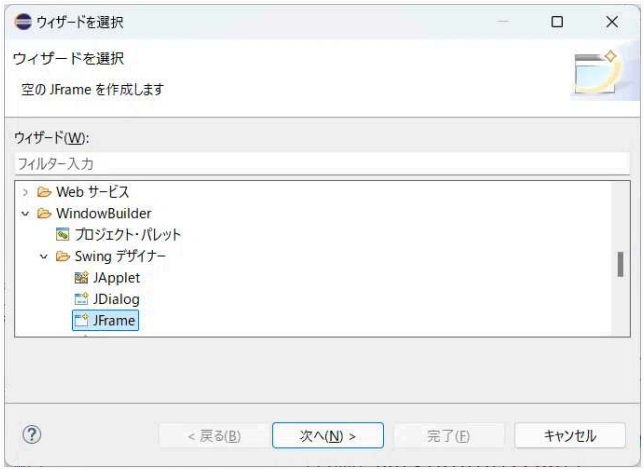

図5．JFrameウィザードの選択

以下の内容で作成
　パッケージ：jp.ict.aso.swing
　名前：Calendar

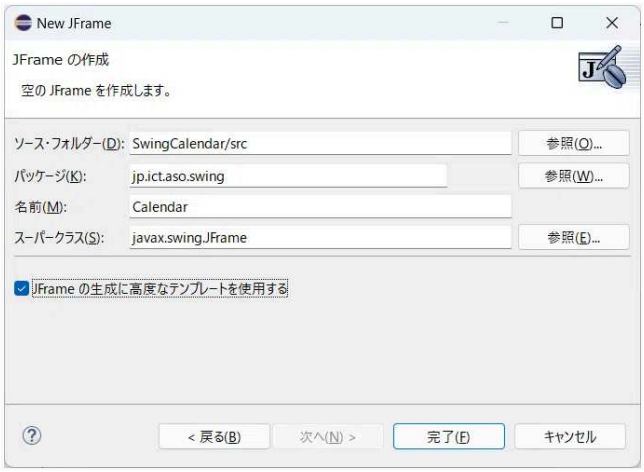

図6．Swingアプリケーションのスケルトン生成

GUI実装

自動生成されたプログラム（スケルトン）からGUIのデザインを実装します。

パレットと構造（コンポーネント、プロパティ）のWindowを利用するのがコツです。デザインイメージは設定反映の参考としてとらえた方が良いでしょう。

①画面中央下部にあるデザインタブでソースコード編集画面からSwingデザイナーに切り替えます。

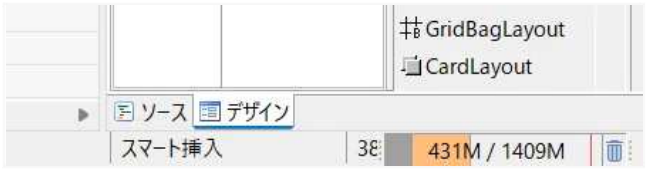

図7．Swingデザイナーに切り替え

②contentPaneのLayoutプロパティをBorderLayoutに設定します。

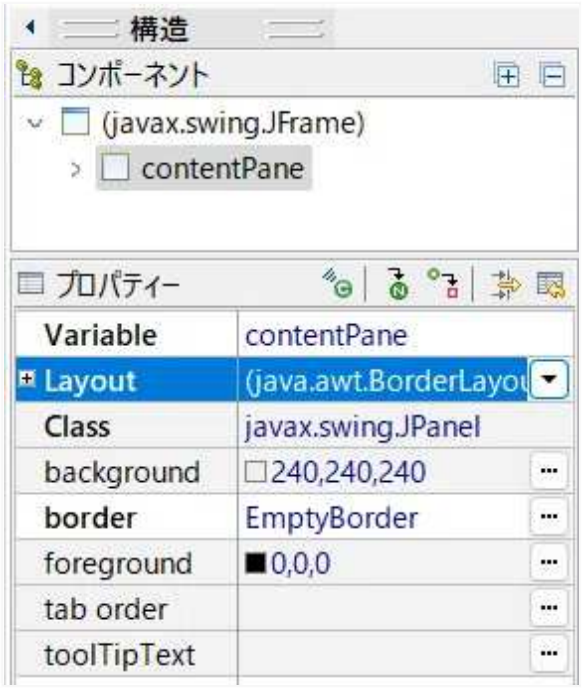

図8．コンテンツペインの設定

③contentPane内にGUI部品のJPanelを3つ、パレットから配置します。JPanelのConstraintsプロパティ（配置する位置）はそれぞれ「North」「Center」「South」とします。図9、図10、図11のように、それぞれのpanelのプロパティの値を設定してください。

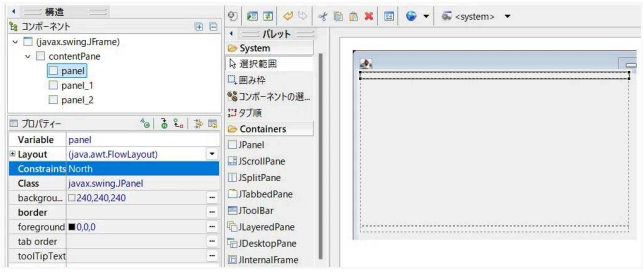

図9．一つ目のJPanelのプロパティ値

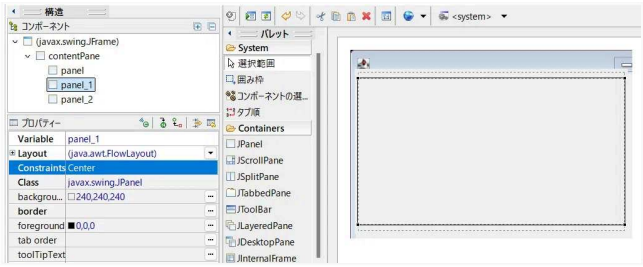

図10．二つ目のJPanelのプロパティ値

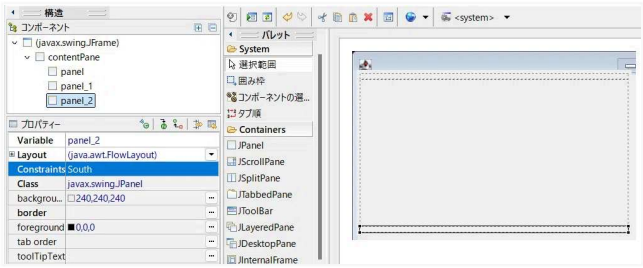

図11．三つ目のJPanelのプロパティ値

④「North」位置のJPanelにGUI部品のJTextField、JLabel、JComboBoxをパレットから配置します。

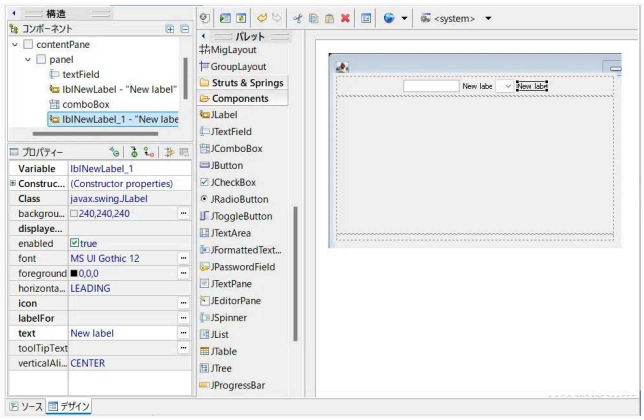

図12．North位置のJPanelの配置

⑤各部品のプロパティを設定します。以下にJTextFieldとJComboboxの設定値を示します。

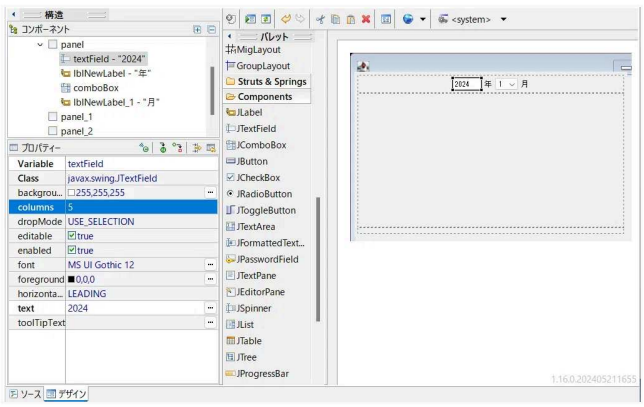

図13．JTexFieldのプロパティ値

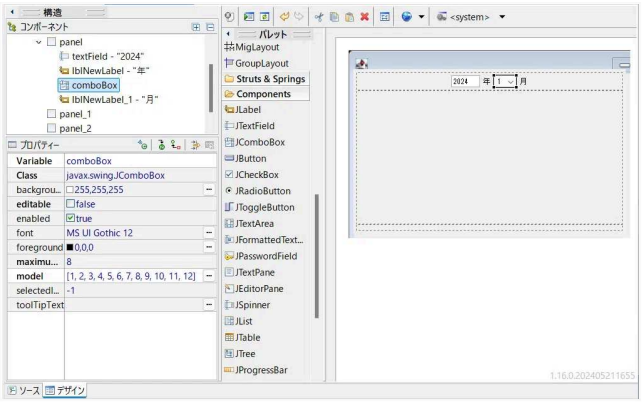

図14．JComboBoxのプロパティ値

⑥「Center」位置のJPanelのLayoutプロパティーをBoxLayoutに変更し、Constructor propertiesのaxisをY_AXISに変更します。

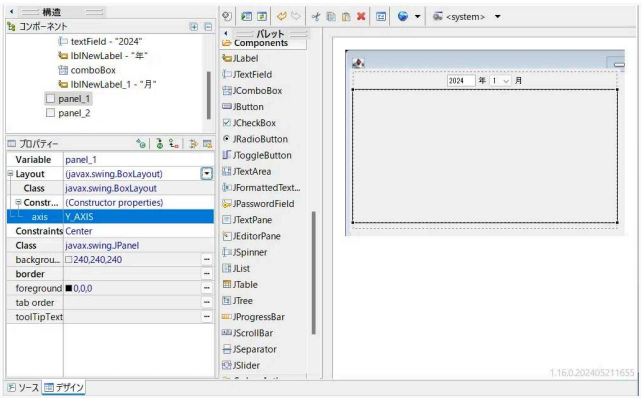

図15．Center位置のJPanelのプロパティ値

⑦「Center」位置のJPanelにGUI部品のJEditorPaneを２つ、パレットから配置します。

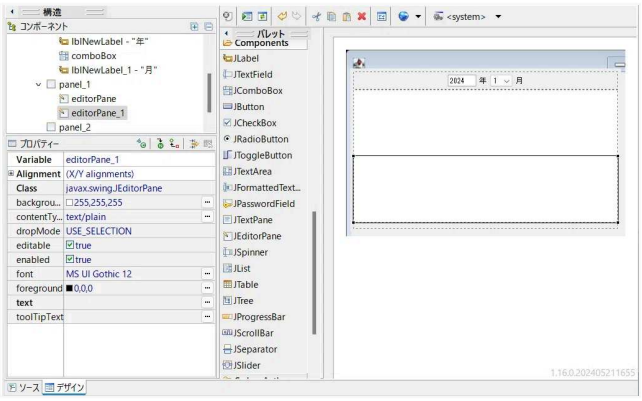

図16．Center位置のJPanelの配置

⑧htmlを使った文字装飾が可能なように、それぞれ（EditorPaneとEditorPane_1）のJEditorPaneのcontentTypeプロパティをtext/htmlに変更します。

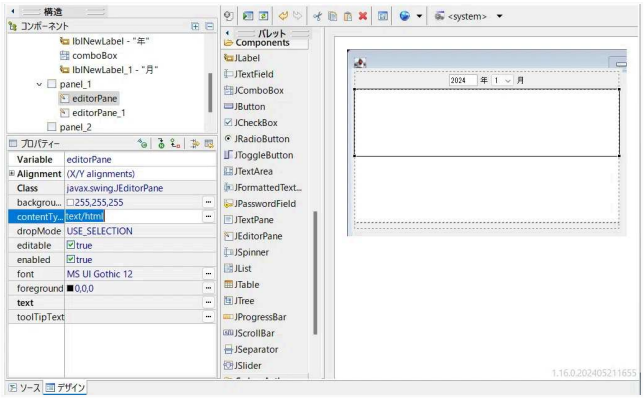

図17．JEditorPaneのプロパティ値

⑨「South」位置のJPanelにGUI部品のJButtonを２つ、パレットから配置します。あわせて、表示テキストを以下のように変更します。

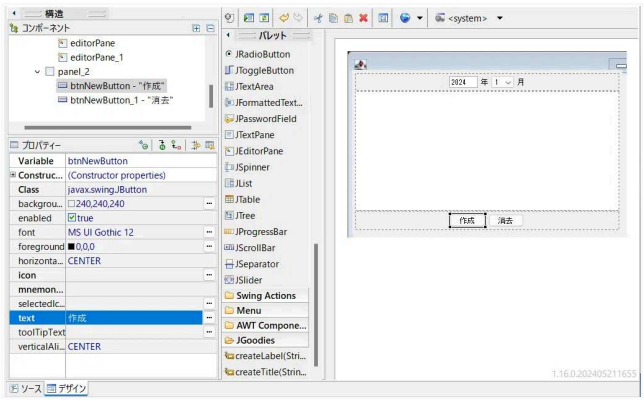

図18．South位置のJPanelの配置

⑩ボタンにイベントリスナーを対応付けます。パレットのSwingActions内にある「新規」のリスナーを選択してそれぞれのボタンをクリックすることで対応付けられます。

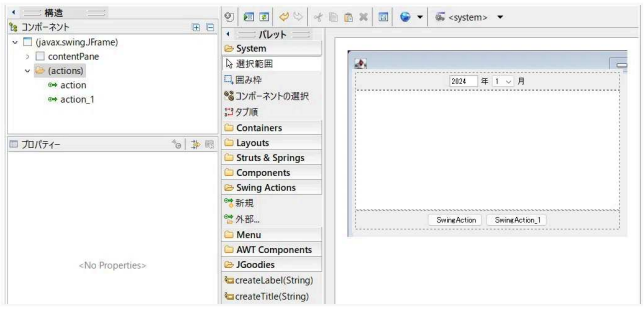

図19．イベントリスナーの対応付け

イベント実装

ソースタブに変更します。

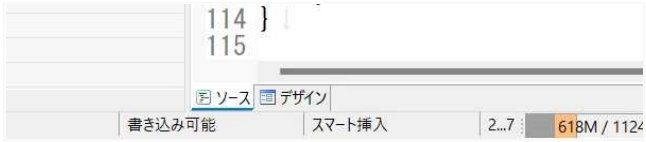

図20．ソースタブに変更

①「作成ボタン」のイベントのソース部分を変更します。

```
private class SwingAction extends AbstractAction {
    public SwingAction() {
        putValue(NAME, "作　成");
        putValue(SHORT_DESCRIPTION, "任意の年月のカレンダーを作成する");
    }
    public void actionPerformed(ActionEvent e) {
        //year,monthは整数に変換
        int year=Integer.parseInt(textField.getText());
        int month=comboBox.getSelectedIndex()+1;

        //一旦表示領域クリア
        head="";
        day="";
        editorPane.setText(head);
        editorPane_1.setText(day);

        //ヘッダーの作成
        head=head+"<br><center>"+textField.getText()+"年"+comboBox.getSelectedItem()+"月";
        editorPane.setText(head);

        // カレンダーの作成
        CalendarHtmlBean chb = new CalendarHtmlBean(year,month);
        chb.create();
        day="<center>"+chb.getDay()+"</center>";
        editorPane_1.setText(day);

    }
}
```

図21．作成ボタンのイベント内容

②「消去ボタン」のイベントのソース部分を変更します。

```
private class SwingAction_1 extends AbstractAction {
    public SwingAction_1() {
        putValue(NAME, "消　去");
        putValue(SHORT_DESCRIPTION, "表示領域をクリアします");
    }
    public void actionPerformed(ActionEvent e) {
        //表示領域クリア
        head="";
        day="";
        editorPane.setText(head);
        editorPane_1.setText(day);
    }
}
```

図22．消去ボタンのイベント内容

③フィールド変数を変更します。//kokoの部分を追加します。

```
public class Calendar extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField textField;
    private final Action action = new SwingAction();
    private final Action action_1 = new SwingAction_1();

    JComboBox comboBox = new JComboBox(); //koko
    JEditorPane editorPane;     //koko
    JEditorPane editorPane_1;   //koko

    String head="";             //koko
    String day="";              //koko

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
```

図23．フィールド変数の変更

④ローカル変数の宣言になっている部分を変更します。//kokoの部分を変更します。３か所あります。

```
comboBox = new JComboBox();                    //koko
comboBox.setModel(new DefaultComboBoxModel(new String[] {"1", "2", "3",
panel.add(comboBox);

JLabel lblNewLabel_1 = new JLabel("月");
panel.add(lblNewLabel_1);

JPanel panel_1 = new JPanel();
contentPane.add(panel_1, BorderLayout.CENTER);
panel_1.setLayout(new BoxLayout(panel_1, BoxLayout.Y_AXIS));

editorPane = new JEditorPane();                //koko
editorPane.setContentType("text/html");
panel_1.add(editorPane);

editorPane_1 = new JEditorPane();              //koko
editorPane_1.setContentType("text/html");
panel_1.add(editorPane_1);

JPanel panel_2 = new JPanel();
contentPane.add(panel_2, BorderLayout.SOUTH);
```

図24．ローカル変数の宣言変更

⑤実行確認します。エディタの画面内で右クリック → 実行 → Javaアプリケーションで実行されます。

図25．実行方法

⑥作成ボタンクリックでカレンダーが表示されるか確認します。

図26．初期カレンダーの表示

ここまでのカレンダーの実装ではいろいろと不具合があるようです。
とりあえず「タイトルがない」「年月が一部消えている」「起動時にカレンダーが表示されていない（作成ボタンをクリックしないと表示されない）」「画面の大きさが任意に変えられてしまう」の４点を修正します。

実装変更

①フレームにタイトルを追加し画面の大きさを調整します。//kokoの部分を追加します。

```
/**
 * Create the frame.
 */
public Calendar() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setTitle("カレンダー");                                    //koko
    setBounds(100, 100, 300, 400);                          //koko
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

    setContentPane(contentPane);
    contentPane.setLayout(new BorderLayout(0, 0));
```

図27．タイトル追加と画面サイズの変更

②画面（フレーム）の大きさを固定します。//kokoの部分を追加します。

```
/**
 * Launch the application.
 */
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                Calendar frame = new Calendar();
                frame.setVisible(true);
                //枠固定
                frame.setResizable(false);        //koko
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}
```

図28．画面サイズの固定

③起動時にデフォルトのカレンダーを表示します。コンストラクタの最後に「//起動時にカレンダーを表示」以下を追加します。

```
        JButton btnNewButton_1 = new JButton("消去");
        btnNewButton_1.setAction(action_1);
        panel_2.add(btnNewButton_1);

//起動時にカレンダーを表示
        //year,monthは整数に変換
        int year=Integer.parseInt(textField.getText());
        int month=comboBox.getSelectedIndex()+1;
        //ヘッダーの作成
        head=head+"<br><center>"+textField.getText()+"年"+comboBox.getSelectedItem()+"月";
        editorPane.setText(head);
        // カレンダーの作成
        CalendarHtmlBean chb = new CalendarHtmlBean(year,month);
        chb.create();
        day="<center>"+chb.getDay()+"</center>";
        editorPane_1.setText(day);
//////////////////////////
    }
```

図29．起動時のカレンダー表示

④実行して動きを確認します。これで完成しました。

図30．カレンダー作成アプリケーション完成

単独起動

実行可能JARファイル

①せっかくですので、単独で起動できるアプリケーションにエクスポートしましょう。Java1.8以上のJREの環境がWindowsのPCにインストールされていればダブルクリックで起動できます。

Eclipseパッケージ・エクスプローラより
Calendarプロジェクトを右クリック → エクスポート
→ Java → 実行可能JARファイル → 次へ
→ 以下のように設定する → 完了

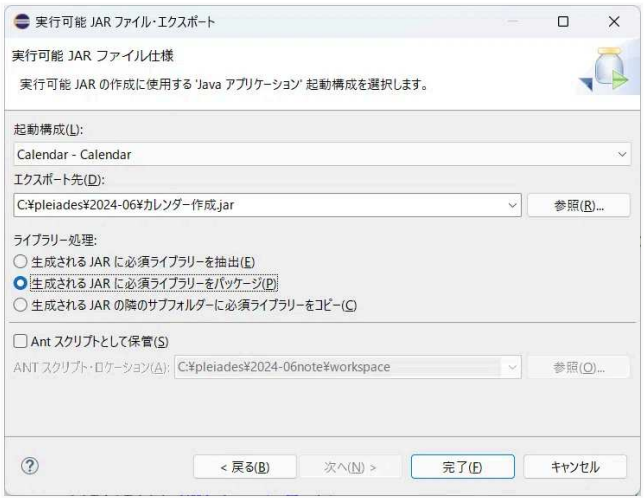

図31．実行可能JARファイルの設定

以下のような警告が出る場合がありますが、気にしません。

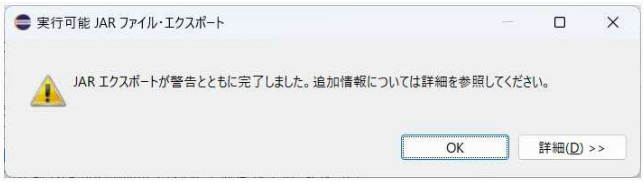

図32．警告ダイアログ

作成されたjarファイルをダブルクリックするとカレンダーが起動します。

図33．実行可能JARファイル

最後に

以上でSwingデザイナーを使って任意の月のカレンダーを表示するデスクトップアプリケーションを作成できました。

ただし、このアプリを実装するにはCalendarHtmlBean.classが必要です。このクラスは『MVCモデルによるWebアプリケーション開発（No４.カレンダー）-EE8』の記事から取得可能です。

ps.祝祭日が変わる場合がありますのでご了承ください。