**How to Configure a GUI**

[GUI Construction 2](#_Toc427682274)

[ Window 2](#_Toc427682275)

[ Page 6](#_Toc427682276)

[ Component 8](#_Toc427682277)

[ Button 8](#_Toc427682278)

[ ComboBox 9](#_Toc427682279)

[ Label 10](#_Toc427682280)

[ TextField 10](#_Toc427682281)

[ PasswordField 11](#_Toc427682282)

[ Date 11](#_Toc427682283)

[ Table 12](#_Toc427682284)

[ AdvancedTable 14](#_Toc427682285)

[ Action configuration 16](#_Toc427682286)

[ customizedAction 17](#_Toc427682287)

[ jumpPanelAction 17](#_Toc427682288)

[ setFontAction 17](#_Toc427682289)

[ cleanAction 18](#_Toc427682290)

[ limitInputAction 18](#_Toc427682291)

[ useTipAction 18](#_Toc427682292)

[ showTableAction 19](#_Toc427682293)

[ showComboBoxAction 19](#_Toc427682294)

[ swipeCardAction 20](#_Toc427682295)

[ playMediaAction 20](#_Toc427682296)

[ rwFingerPrintAction 21](#_Toc427682297)

[ openBrowserAction 21](#_Toc427682298)

[ runCommandAction 21](#_Toc427682299)

[ virtualKeyboardAction 22](#_Toc427682300)

[ setPrinterAction 22](#_Toc427682301)

[ sendMsgAction 23](#_Toc427682302)

[ insertICardAction 25](#_Toc427682303)

[ writeICardAction 25](#_Toc427682304)

[ moveCursorAction 26](#_Toc427682305)

## GUI Construction

A general GUI includes three parts, top level frame, middle level page and inner component. That means a whole GUI needs a window, a load of panels and enormous non-container components.

If you wanna configure an application running on JVM, you need to declare a window firstly. And then you need to describe pages switching in this window. On a time point, window will only show a page. That means you need to define page relationship depending on business requirements. In each page, you can lay out different components, such as button, table etc. how to design a page, you need to consider page business purpose. Following paragraph will narrate detailed configuration of each plain-vanilla part.

### Window

A window is the outer frame or the top level container, it can be a Frame or Dialog. Currently, a window only contain panel type, meanwhile only one panel will be show on one time point.

A window can be represented as a xml file, as a default setting, system will access ‘./conf/xml\_window.xml’ in running folder as startup entry.

A typical and ordinary entry file ‘xml\_window.xml’ is as below

<?xml version="1.0" encoding="UTF-8"?>

<window

xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'

xsi:noNamespaceSchemaLocation='WindowXmlSchema.xsd'> <!--window schema and namespace declaration -->

<id>0</id> <!--window container id, an unique number. As top frame we suggest giving its zero value -->

<type>Frame</type> <!--window type, optional value is Frame or Dialog -->

<ui-style> <!--window UI style setting -->

<lookandfeel> <!--GUI look&feel name, this is optional. if no setting, default value will be used -->

com.jgoodies.looks.windows.WindowsLookAndFeel

</lookandfeel>

<font> <!--System default font setting, optional setting. -->

<name>Fangsong</name>

<size>21</size>

<style>BOLD</style>

</font>

<background>/image/5.jpg</background> <!--System background setting, optional setting -->

<progress>/image/loader\_orange\_256.gif</progress> <!--System progress animation setting, optional setting -->

<color>115,164,209</color> <!--System basic color setting, you can change base color via RGB value. optional setting -->

<textcolor>64,0,64</textcolor> <!--System text color setting, optional setting -->

</ui-style>

<driver> <!--System peripheral drivers setting, optional setting -->

<device>

<type>magnetCard</type> <!--peripheral device type -->

<port>com3</port> <!--serial port name-->

<dll>emptydll</dll> <!--driver dll name -->

</device>

<device>

<type>ICCard</type>

<port>com3</port>

<dll>emptydll</dll>

</device>

<device>

<type>sensorCard</type>

<port>com3</port>

<dll>emptydll</dll>

</device>

<device>

<type>printer</type>

<port>com3</port>

<dll>emptydll</dll>

</device>

<device>

<type>keyboard</type>

<port>com3</port>

<dll>emptydll</dll>

</device>

</driver>

<attribute> <!--Window attribute setting -->

<text>Application of Bank</text> <!-- title text -->

<icon>/image/bank-icon-256.png</icon> <!--window icon -->

</attribute>

<menubar> <!--Menu bar setting, optional setting. If you need no menu, remove it -->

<menu>

<id>-10</id>

<name>Operation of Conduction</name>

<item>

<id>-11</id>

<name>User Management</name>

<action>

<jumpPanelAction>

<nextPanel>200</nextPanel>

</jumpPanelAction>

</action>

</item>

<item>

<id>-12</id>

<name>Device Management</name>

<action>

<jumpPanelAction>

<nextPanel>300</nextPanel>

</jumpPanelAction>

</action>

</item>

</menu>

<menu>

<id>-20</id>

<name>About</name>

<enable>true</enable>

<item>

<id>-21</id>

<name>Version</name>

<action>

<customizedAction>

<className>king.flow.action.customization.AboutAction</className>

</customizedAction>

</action>

</item>

</menu>

</menubar>

<contents> <!--Page setting, this mandatory setting, you must assign page files here-->

<page>./conf/100/panel-100.xml</page> <!--page file link, if you wanna load panel in a window, you should add a link of panel file here -->

<page>./conf/200/panel-200.xml</page>

<page>./conf/300/panel-300.xml</page>

<page>./conf/300/panel-330.xml</page>

<page>./conf/300/panel-360.xml</page>

<page>./conf/900/panel-900.xml</page>

<page>./conf/900/panel-970.xml</page>

</contents>

</window>

That’s a standard window as picture below, it’s constructed wholly based on configuration above



### Page

Page is really container of component, which is loaded by window and contains all working components such as button, table , combobox and so on.

The system just support one type page, which is Panel, a middle-level container. A panel cannot show solely, only when a window has loaded it, it can show.

A typical and ordinary page configuration is a single xml file as below,

<?xml version="1.0" encoding="UTF-8"?>

<panel

xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'

xsi:noNamespaceSchemaLocation='../WindowXmlSchema.xsd'> <!-- Panel declaration -->

<id>100</id> <!--Panel id, an unique number -->

<type>Panel</type>

<active>true</active> <!--Panel initial show on/off setting, true means this panel will show firstly in a window, false means hide after loaded. Only one page should be true among lots of pages-->

<background>/image/5.jpg</background> <!--Panel background setting, optional setting -->

<component> <!-- Component setting in a panel container-->

<id>101</id> <!-- Component unique id -->

<type>Label</type> <!-- Component type, current supported type are Button, Table, AdvancedTable, ComboBox, Label, TextField, PasswordField, TextArea, Date -->

<attribute> <!-- Component attribute setting-->

<rect> <!--Component rectangle attribute -->

<x>600</x>

<y>0</y>

<width>600</width>

<heigh>150</heigh>

</rect>

<text>Welcome to use bank application</text> <!--Component text property, optional setting -->

<icon>/image/5.ico</icon> <!-- Component icon property, optional setting -->

<debug >false</debug> <!--debug attribute, true will show component border, false no border show. Optional setting -->

</attribute>

<action> <!-- Component action setting, optional setting -->

<fontSetAction>

<fontSize>45</fontSize>

</fontSetAction>

</action>

</component>

<component>

<id>104</id>

<type>ComboBox</type>

<attribute>

<rect>

<x>700</x>

<y>400</y>

<width>300</width>

<heigh>40</heigh>

</rect>

</attribute>

<action>

<comboShowAction>

<items>CNY/Yuan, USD/Dollar, JPY/Yen</items>

</comboShowAction>

<mediaPlayAction>

<media>./media/goodbye.wav</media>

</mediaPlayAction>

</action>

</component>

</panel>

### Component

Component is the plain-vanilla unit of GUI system. A page is composed of some components, and current supported types include Button, Table, AdvancedTable, ComboBox, Label, TextField, PasswordField, and Date.

#### Button

Sample configuration:

<component>

<id>102</id>

<type>Button</type>

<attribute>

<text>查询业务</text>

<rect>

<x>200</x>

<y>300</y>

<width>300</width>

<heigh>50</heigh>

</rect>

<debug>false</debug>

</attribute>

</component>



#### ComboBox

Sample configuration:

<component>

<id>104</id>

<type>ComboBox</type>

<attribute>

<rect>

<x>700</x>

<y>400</y>

<width>300</width>

<heigh>40</heigh>

</rect>

</attribute>

<action>

<showComboBoxAction>

<items>CNY/人民币, USD/美元, JPY/日元</items>

</showComboBoxAction>

</action>

</component>



#### Label

Sample configuration:

<component>

<id>101</id>

<type>Label</type>

<attribute>

<rect>

<x>600</x>

<y>0</y>

<width>600</width>

<heigh>150</heigh>

</rect>

<text>欢迎使用自助银行查询业务</text>

</attribute>

<action>

<fontSetAction>

<fontSize>45</fontSize>

</fontSetAction>

</action>

</component>



#### TextField

Sample configuration:

<component>

<id>107</id>

<type>TextField</type>

<attribute>

<rect>

<x>400</x>

<y>500</y>

<width>300</width>

<heigh>40</heigh>

</rect>

</attribute>

</component>



#### PasswordField

Sample configuration:

<component>

<id>105</id>

<type>PasswordField</type>

<attribute>

<rect>

<x>400</x>

<y>400</y>

<width>300</width>

<heigh>40</heigh>

</rect>

</attribute>

</component>



#### Date

Sample configuration:

<component>

<id>302</id>

<type>Date</type>

<attribute>

<rect>

<x>180</x>

<y>30</y>

<width>300</width>

<heigh>50</heigh>

</rect>

</attribute>

</component>



#### Table

Sample configuration:

<component>

<id>311</id>

<type>Table</type>

<attribute>

<rect>

<x>0</x>

<y>100</y>

<width>300</width>

<heigh>450</heigh>

</rect>

<debug>false</debug>

</attribute>

<action>

<showTableAction>

<columnNames>流水号, 名称, 开始时间, 结束时间, 费用</columnNames>

</showTableAction>

</action>

</component>



If you need a scroll bar aside a table, you need to put a table component in a decorator as below, and otherwise there is no scroll bar showing with this table.

Sample configuration:

**<decorator>**

**<id>310</id>**

**<type>ScrollPanel</type>**

**<attribute>**

**<rect>**

**<x>0</x>**

**<y>100</y>**

**<width>1600</width>**

**<heigh>600</heigh>**

**</rect>**

**<debug>false</debug>**

**</attribute>**

<component>

<id>311</id>

<type>Table</type>

<attribute>

<rect>

<x>0</x>

<y>100</y>

<width>300</width>

<heigh>450</heigh>

</rect>

<debug>false</debug>

</attribute>

<action>

<fontSetAction>

<fontSize>15</fontSize>

</fontSetAction>

</action>

<action>

<showTableAction>

<columnNames>流水号, 名称, 开始时间, 结束时间, 费用</columnNames>

</showTableAction>

</action>

</component>

**</decorator>**

****

#### AdvancedTable

Sample configuration:

<component>

<id>330337</id>

<type>AdvancedTable</type>

<attribute>

<rect>

<x>0</x>

<y>95</y>

<width>1275</width>

<heigh>600</heigh>

</rect>

</attribute>

<action>

<showTableAction>

<columnNames>流水号, 名称, 开始时间, 结束时间, 费用</columnNames>

</showTableAction>

</action>

<action>

<sendMsgAction>

<cmdCode>330</cmdCode>

<prsCode>queryAcount</prsCode>

<conditions>330332,330334</conditions>

<nextStep>

<nextPanel>330</nextPanel>

<display>330337</display>

</nextStep>

<exception>

<nextPanel>970</nextPanel>

<display>971</display>

</exception>

<checkRules>

<notEqual>

<more>330334</more>

<less>330332</less>

<errMsg>查询开始时间不能够大于结束时间</errMsg>

</notEqual>

</checkRules>

</sendMsgAction>

</action>

</component>



### Action configuration

After GUI is constructed, you need to configure component action to deal with business requirements. There are several action type here, each of them can do specific functionality.

The table below describes action configuration relationship with component, ‘√’means this action can be configured with this component.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Button | ComboBox | Label | TextField | PasswordField | Table | AdvancedTable |
| jumpPanelAction | √ |  |  |  |  |  |  |
| setFontAction | √ | √ | √ | √ | √ | √ | √ |
| cleanAction | √ |  |  |  |  |  |  |
| limitInputAction |  |  |  | √ | √ |  |  |
| useTipAction | √ | √ | √ | √ | √ |  |  |
| showTableAction |  |  |  |  |  | √ | √ |
| showComboBoxAction |  | √ |  |  |  |  |  |
| swipeCardAction |  | √ |  |  |  |  |  |
| playMediaAction | √ | √ |  | √ | √ |  |  |
| rwFingerPrintAction |  |  |  | √ | √ |  |  |
| openBroswerAction | √ |  |  |  |  |  |  |
| runCommandAction | √ |  |  |  |  |  |  |
| virtualKeyboardAction | √ |  |  | √ |  |  |  |
| setPrinterAction | √ |  |  |  |  |  |  |
| sendMsgAction | √ |  |  |  |  |  |  |
| insertICardAction | √ |  |  |  |  |  |  |
| writeICardAction | √ |  |  |  |  |  |  |
| moveCursorAction | √ | √ |  | √ | √ |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

#### customizedAction

This is an entry for outer developers, they can develop customized action plugin and put into current GUI system. Moreover, third party user can implement special functionality to meet ad-hoc requirement.

#### jumpPanelAction

This action is about page jumping, currently only button/menu can configure this action. That says if you have set this action with component, when you click button/menu, the next page you wanna go will come out.

Sample configuration:

<action>

<jumpPanelAction>

<nextPanel>200</nextPanel>

</jumpPanelAction>

</action>

#### setFontAction

This action is used to set component text font

Sample configuration:

<action>

<setFontAction>

<fontName>Fangsong</fontName> <!--optional setting -->

<fontSize>18</fontSize>

<fontStyle>BOLD</fontStyle> <!--optional setting, valid value is BOLD/PLAIN/ITALIC -->

</setFontAction>

</action>

#### cleanAction

When you leave some page, you hope to clean all data in that page, you can use this action.

Sample configuration:

<action>

<cleanAction>

<conditions>311,307, 337</conditions><!-- clean value of component 311,307 and 337 -->

</cleanAction>

</action>

#### limitInputAction

This action will limit input format in text field and password field component

Sample configuration:

<action>

<limitInputAction>

<length>6</length> <!-- limit 6 characters input in one text component -->

</limitInputAction>

</action>

#### useTipAction

Make component show tip when cursor moves to this component

Sample configuration:

<action>

<useTipAction>

<tip>&lt;html&gt;&lt;center&gt;&lt;h1&gt;Hello,&lt;/h1&gt;&lt;h2 style=&#39;color: red; font-style: italic;&#39;&gt;World&lt;/h2&gt;&lt;b&gt;Modified&lt;br/&gt;ToolTipText&lt;/b&gt;&lt;/center&gt;&lt;br/&gt;so simple.请输入账号密码</tip> <!-- tip content -->

</useTipAction>

</action>



#### showTableAction

This action allows to show a table in a page

Sample configuration:

<action>

<showTableAction>

<columnNames>流水号, 名称, 开始时间, 结束时间, 费用</columnNames> <!--table column names -->

</showTableAction >

</action>



#### showComboBoxAction

This action allows to show a combobox in a page

Sample configuration:

<action>

<showComboBoxAction>

<items>/, ACTION1/刷接触式卡, ACTION2/刷磁条卡, ACTION3/刷IC卡</items> <!—current system only support three types, ACTION1（SENSOR）,ACTION2（MAGNET）,ACTION3（IC）,ACTION4（2In1） -->

</showComboBoxAction>

</action>



#### swipeCardAction

This action allows to show a combobox in a page

Sample configuration:

<action>

<showComboBoxAction>

<items>/, ACTION2/刷磁条卡, ACTION3/插入IC卡</items>

</showComboBoxAction>

<swipeCardAction>

<nextCursor>333</nextCursor> <!-- next component cursor focuses on after swipe card operation -->

<editable>true</editable> <!—whether combobox is editable and allow to keyboard input -->

</swipeCardAction>

</action>

#### playMediaAction

Play a media when you move a cursor to this component

Sample configuration:

<action>

<playMediaAction>

<media>./media/goodbye.wav</media>

</playMediaAction>

</action>

#### rwFingerPrintAction

Read or write finger print when you put finger on peripheral device

Sample configuration:

<action>

<rwFingerPrintAction>

<nextCursor>104</nextCursor> <!-- next component cursor focuses on after swipe card operation -->

<writePrint>true</writePrint> <!—whether this is finger print registry operation -->

</rwFingerPrintAction>

</action>

#### openBrowserAction

Open a browser instance to access a web site

Sample configuration:

<action>

<openBrowserAction>

<url>http://www.baidu.com</url>

</openBrowserAction>

</action>

#### runCommandAction

Run local command offered by local operation system.

Sample configuration:

<action>

<runCommandAction>

<command>notepad</command> <!-- open notepad -->

</runCommandAction>

</action>



#### virtualKeyboardAction

Open Chinese character type tool.

Sample configuration:

<action>

<virtualKeyboardAction>

<start>AVF\_Display.exe</start>

<stop>AVF\_Hide.exe</stop>

</virtualKeyboardAction>

</action>

#### setPrinterAction

Setting receipt format, including receipt header and tail content.

Sample configuration:

<action>

<setPrinterAction>

<header>通商银行</header>

<tail>此凭条仅供参照自助银行,如有异议请联系客服中心:96669</tail>

</setPrinterAction>

</action>

#### sendMsgAction

This action is about communication task. When you configure this action with a component that usually is button, action will collect component values user designates and send to convergent server.

Sample configuration:

<action>

<sendMsgAction>

<prsCode>queryAcount</prsCode> <!-- operation PRS code -->

<conditions>302,304,307,333,334,335</conditions> <!—those are data need to send up to server, and we need to catch those values from these components -->

<nextStep>

<nextPanel>300</nextPanel> <!-- operation is successful, go to next page -->

<display>311</display> <!-- on successful page, show result in this component -->

</nextStep>

<exception>

<nextPanel>970</nextPanel> <!-- operation is failed, go to error page -->

<display>971</display> <!-- in error page, show result to user -->

</exception>

<checkRules> <!-- rule checking before sending data to server -->

<notNull>

<content>333</content> <!-- component input must not be empty -->

<errMsg>密码不能为空</errMsg>

</notNull>

<notNull>

<content>334</content>

<errMsg>确认密码不能为空</errMsg>

</notNull>

<notNull>

<content>335</content>

<errMsg>账户不能为空</errMsg>

</notNull>

<validateCJK>

<content>335</content> <!-- component input must be Chinese character -->

<errMsg>账户名称必须是中文</errMsg>

</validateCJK>

<validateCJK>

<content>336</content>

<errMsg>输入必须是中文</errMsg>

</validateCJK>

<template>

<content>307</content> <!-- component input must start with pattern value -->

<pattern>6222</pattern>

<errMsg>当前使用非本行发行卡片，请使用本行卡</errMsg>

</template>

<equal>

<conditions>333,334</conditions> <!-- components input must be equal -->

<errMsg>两次输入的密码不一致,请重新输入</errMsg>

</equal>

<notEqual> <!-- component input must not be equal -->

<more>304</more>

<less>302</less>

<errMsg>查询开始时间不能够大于结束时间</errMsg>

</notEqual>

</checkRules>

</sendMsgAction>

</action>

#### insertICardAction

When you hit a button and hope to enter IC operation page, I need to make you know inserting card firstly. So I will play tip animation and wait user to insert IC card. And then, I can read card information as well as open operation page.

Sample configuration:

<action>

<insertICardAction>

<suceessfulPanel>370</suceessfulPanel> <!-- insert a valid IC, jumping to next page -->

<failedPanel>200</failedPanel> <!-- no valid IC found, jumping to a page-->

</insertICardAction>

</action>

#### writeICardAction

Just like sendMsgAction, you can configure this action to write data to IC card after a transaction. The difference between writeICardAction and sendMsgAction is specific function, because there probably is error happening in writing an IC card, so we have to send strike-a-balance message to convergent server.

Sample configuration:

<action>

<writeICardAction>

<prsCode>buygas</prsCode>

<conditions>370004,370005,370006</conditions>

<checkRules>

<notNull>

<content>370004</content>

<errMsg>370004输入不能够为空</errMsg>

</notNull>

<notNull>

<content>370005</content>

<errMsg>370005输入不能够为空</errMsg>

</notNull>

<notNull>

<content>370006</content>

<errMsg>370006输入不能够为空</errMsg>

</notNull>

</checkRules>

<exception>

<display>9101</display>

<nextPanel>910</nextPanel>

</exception>

<nextStep>

<display>9101</display>

<nextPanel>910</nextPanel>

</nextStep>

</writeICardAction>

</action>

#### moveCursorAction

This action will move cursor when you hit up/down arrow key.

Sample configuration:

<action>

<moveCursorAction>

<upCursor>106</upCursor>

<downCursor>105</downCursor>

</moveCursorAction>

</action>