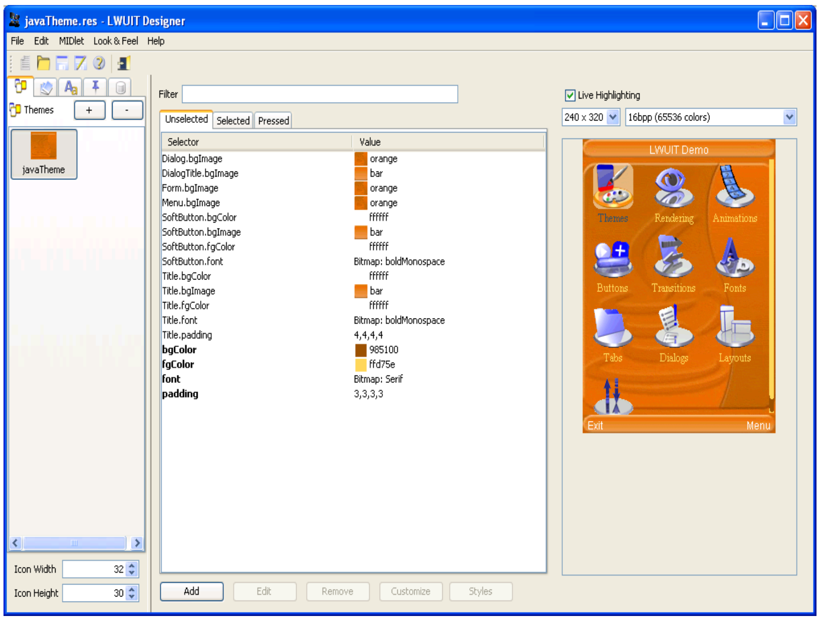
LWUIT Designer

The LWUIT Designer is a standalone GUI tool that allows UI experts, developers, and translators to open, create, and edit resource packages for LWUIT. The LWUIT Designer was designed for visual work and provides "live" preview of all UI changes, enabling rapid UI customization.

Currently the LWUIT Designer and the Ant tasks (discussed in the Developer's Guide) accomplish the same thing, with one limitation. In the LWUIT Designer all fonts used by the Theme must be defined within the theme itself. A theme cannot reference a font defined in a different resource file. This limitation does not apply to the Ant tasks.

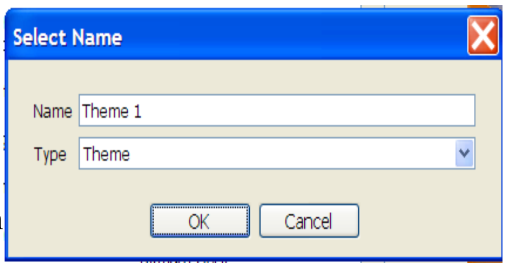


Selector	Value
Dialog.bgImage	🟧 orange
DialogTitle.bgImage	🟧 bar
Form.bgImage	🟧 orange
Menu.bgImage	🟧 orange
SoftButton.bgColor	ffffff
SoftButton.bgImage	🟧 bar
SoftButton.fgColor	ffffff
SoftButton.font	Bitmap: boldMonospace
Title.bgColor	ffffff
Title.bgImage	🟧 bar
Title.fgColor	ffffff
Title.font	Bitmap: boldMonospace
Title.padding	4,4,4,4
bgColor	🟫 985100
fgColor	🟨 ffd75e
font	Bitmap: Serif
padding	3,3,3,3

Illustration 1: Editing the Default LWUIT Look and Feel

The LWUIT Designer supports the six LWUIT resource types: Image, Animation, Theme, Font , Localization and Data. To use the tool:

- Use File > Open to load an existing resource (`.res`) file.
- To Add a resource click the + button in the tab representing the element type you wish to add & specify a name for the resource. The new resource is added under the appropriate tab.
- To create a new theme, select the Theme node, then click the + button. Note that a resource bundle can contain more than one theme.



Name: Theme 1
Type: Theme

Illustration 2: Add Resource Dialog

Note that the "live preview" is displayed for themes only and represents the behavior of the theme alone. It doesn't contain the other resources in the file that do not relate to the theme.

Images and Animations

Images and animations can be used either by a theme or by the LWUIT application. The LWUIT Designer supports images (JPEG, PNG) and animated GIFs. The images and animations can be replaced using the ... button.

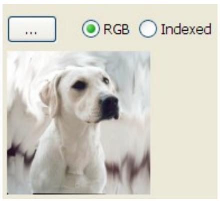


Illustration 3: Image View

Standard images can also be indexed. An indexed image takes less space in the final application and occupies less memory space, however, it takes longer to draw and supports up to 256 colors. When you toggle the indexed image radio button, the number of colors is verified. If more than 256 colors are present the application offers to try to reduce that number (with quality implications) it is recommended you use an image editor tool to index the images beforehand.

Notice that an Alpha channel (beyond full transparency) might be somewhat problematic with indexed images.

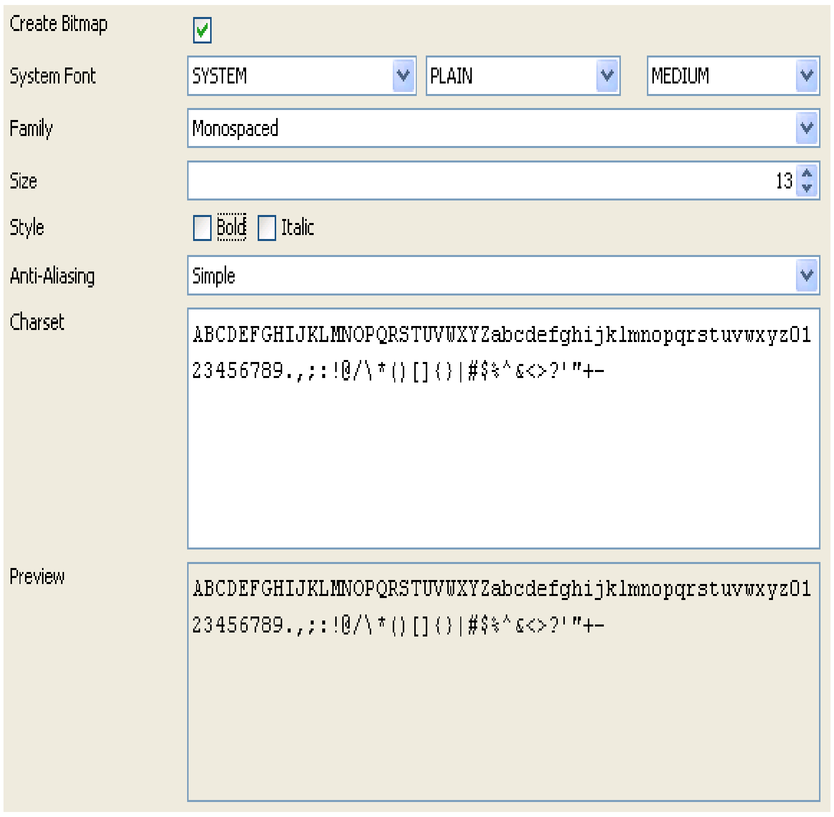
Fonts

The LWUIT Designer can create bitmap fonts for the device from any font installed in your operating system.

Note, using the LWUIT Designer does not grant you permission to use the fonts commercially in any way. Licensing the right to use a particular font within a mobile application is strictly your responsibility.

To create a new bitmap font, select a font size and style from the operating system and specify the characters you need from the font (defaults to upper and lower case English with numbers and symbols). Notice that the more characters you pick in the charset, the more RAM the font consumes.

Anti-aliasing is "built in" to the bitmap font. When running under Java 5 the LWUIT Designer has two anti-aliasing options: Off indicates no anti-aliasing in the bitmap font, and Simple indicates standard anti-aliasing.



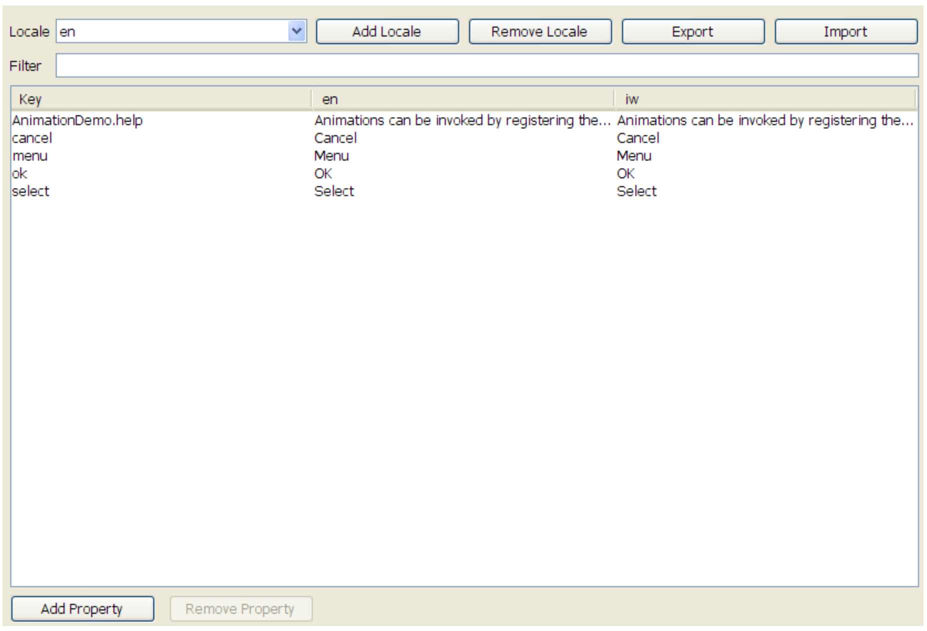
Create Bitmap	✓
System Font	SYSTEM, PLAIN, MEDIUM
Family	Monospaced
Size	13
Style	☐ Bold ☐ Italic
Anti-Aliasing	Simple
Charset	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz01 23456789.,;:!@/*()[]{}|#$%^&<>?'"+-
Preview	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz01 23456789.,;:!@/*()[]{}|#$%^&<>?'"+-

Illustration 4: Font Editing View

Localization

A localization resource can be edited by assigning key/value pairs to use within the application. A key can be mapped to a resource name in any locale.

The editor allows you to add or remove locales listed in the combo box above and appropriately edit the locale entries in the table below. To add or remove a locale property use the buttons on the bottom of the screen.



Key	en	iw
AnimationDemo.help	Animations can be invoked by registering the...	Animations can be invoked by registering the...
cancel	Cancel	Cancel
menu	Menu	Menu
ok	OK	OK
select	Select	Select

Illustration 5: Localization/Internationalization View

Themes

To modify a theme resource, set the selectors and the theme resources to appropriate values to produce an attractive UI. When creating a new theme you see a UI containing the table of selectors and resources (for more in depth details of themes for developers please see the Theme chapter in the *Developer's Guide*).

To modify the theme, choose a selector on the left side then click the Edit button. You can add new selectors using the Add button in the theme . To modify an existing selector, select it from the table and either double clicking or press the Edit button.

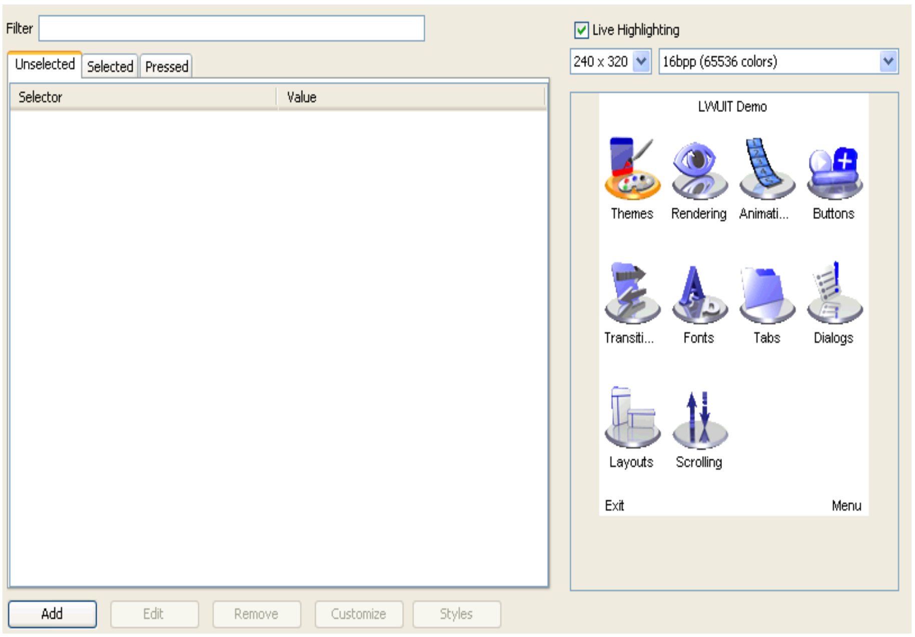


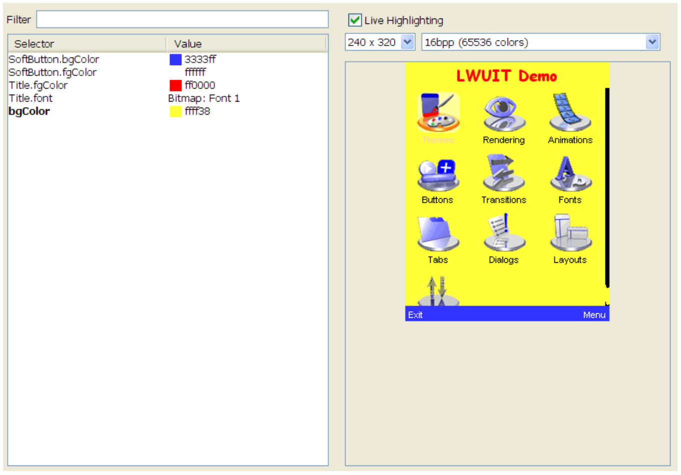
Illustration 6: Blank Theme View Without Any Styles

Example: Adding a New Theme

An example might be the best way to illustrate something like this:

1. Use the + button to add a new theme and select it in the tab.
2. Click the Add button within the theme area (Add Entry) and select bgColor for the attribute.
   - Pick yellow using the ... button next to color. Click OK.
     You have created a "default attribute" where the default background color for all components is yellow.
3. Click the Add button again and select SoftButton in the Components combo box.
   - Select bgColor in the attribute combo box.
   - Use the ... button next to color to pick blue. Click OK.

     You have overridden the default specifically for the SoftButton.
4. Because black over blue is unreadable, add another entry for SoftButton.
   - Pick the fgColor and make it white.



Selector	Value
SoftButton.bgColor	🟦 3333ff
SoftButton.fgColor	ffffff
Title.fgColor	🟥 ff0000
Title.font	Bitmap: Font 1
bgColor	🟨 ffff38

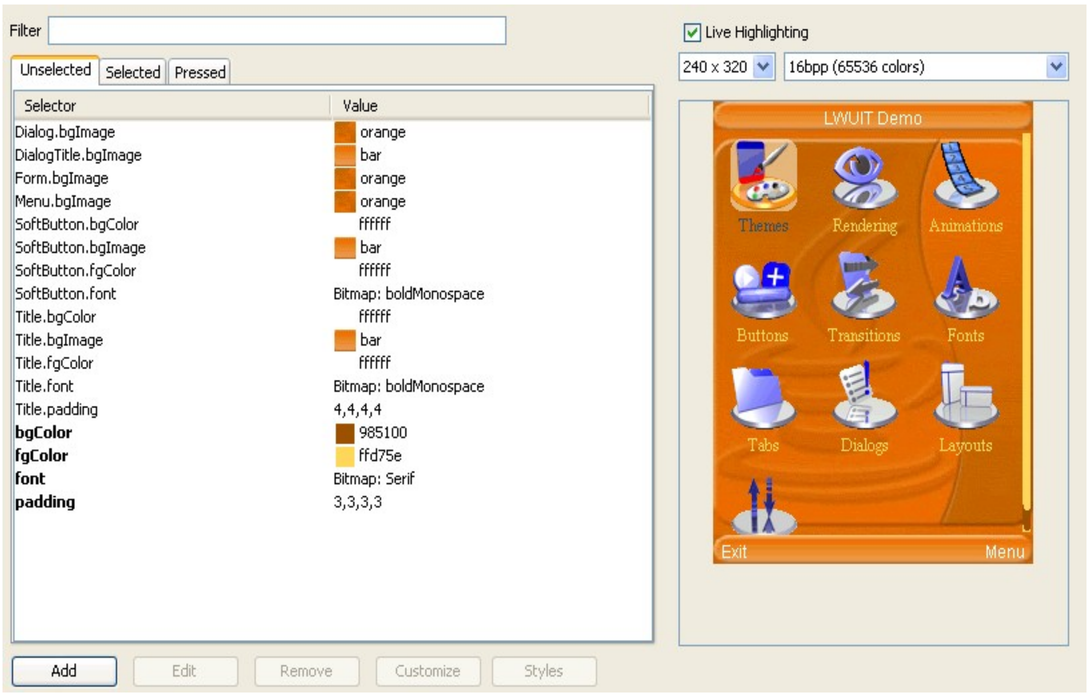
Illustration 7: Resulting Theme From Example

5. The title looks small and insignificant. You might add a Title fgColor and set it to red, but that's not quite enough.
   - Click on add and select Title/font, then select Bitmap from the font options.
   - Click ... next to the bitmap font combo to create a new font.
     Illustration 6 shows "Comic Sans MS" 18 Bold selected.

You can gain deeper understanding of the selector concept from the Style and Theme chapters in the

Developer's Guide.

Modifying Theme Properties


Filter

☑ Live Highlighting

240 x 320 | 16bpp (65536 colors)

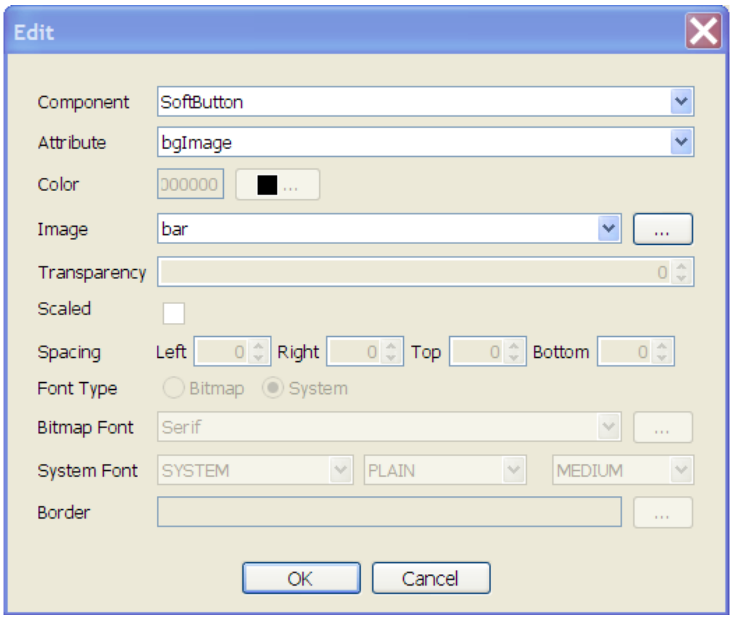
Unselected | Selected | Pressed

Selector	Value
Dialog.bgImage	🟧 orange
DialogTitle.bgImage	🟧 bar
Form.bgImage	🟧 orange
Menu.bgImage	🟧 orange
SoftButton.bgColor	ffffff
SoftButton.bgImage	🟧 bar
SoftButton.fgColor	ffffff
SoftButton.font	Bitmap: boldMonospace
Title.bgColor	ffffff
Title.bgImage	🟧 bar
Title.fgColor	ffffff
Title.font	Bitmap: boldMonospace
Title.padding	4,4,4,4
bgColor	🟫 985100
fgColor	🟨 ffd75e
font	Bitmap: Serif
padding	3,3,3,3

Illustration 8: Theme View When Editing the Default LWUIT Look and Feel

Another way to learn about themes is by experimentation. When you check the “Live Highlighting” box, as shown in Illustration 6, and select a table entry, the relevant property “blinks” on the screen. This allows you to investigate what theme aspects affect the application, otherwise it might be hard to notice its effect).

You can modify and add theme properties very easily using the “Edit” dialog. This dialog allows you to specify the component type (or no component for a “global/default” property) and the attribute that you wish to modify. As you make changes in this dialog the preview is updated . Click OK to make the changes to the preview permanent.


Edit

Component	SoftButton
Attribute	bgImage
Color	000000 ■ ...
Image	bar ...
Transparency	0
Scaled	☐
Spacing	Left 0 Right 0 Top 0 Bottom 0
Font Type	Bitmap [● System]
Bitmap Font	Serif ...
System Font	SYSTEM PLAIN MEDIUM
Border	...

Illustration 9: Theme View Editing Option

This dialog abstracts most of the complexity related to the different attribute types. For example, the font attribute only allows setting a bitmap or system font while a bgImage attribute only allows selecting or adding an image.

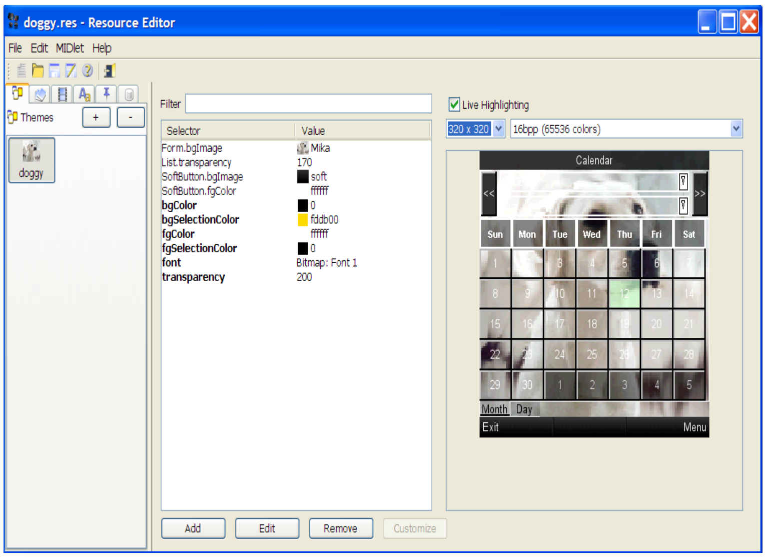
Data

Data is generally designed for developers and shouldn't be used by designers . An arbitrary file can be placed within this section and it can be accessed by developers in runtime. This section has no effect on the rest of the functionality even if the data file is an image or font.

Customizing The Preview

The preview showing the LWUIT Demo allows for easy customization of a MIDlet which is not necessarily limited to the LWUIT Demo. The LWUIT Designer allows plugging in your own MIDlet so you can test your theme on the fly.

To install your own MIDlet into the LWUIT Designer preview panel use the MIDlet > Pick MIDlet menu and select the JAR file for your own MIDlet.



Selector	Value
Form.bgImage	Mika
List.transparency	170
SoftButton.bgImage	■ soft
SoftButton.fgColor	ffffff
bgColor	■ 0
bgSelectionColor	🟨 fddb00
fgColor	ffffff
fgSelectionColor	■ 0
font	Bitmap: Font 1
transparency	200

Illustration 10: LWUIT Designer With A Different MIDlet

There are however several restrictions and limitations in this feature. Since the MIDlet will be executed in Java SE it can't leverage javax.microedition API's, while the API's are present they are implemented in stub form. E.g. if you use RMS, GCF etc. they will return null for all queries and do nothing in all operations.

Additionally invoking features such as themeing won't work for obvious reasons.

In case fo a failure in the MIDlet the LWUIT Designer will silently load the LWUIT Demo in the preview and use it instead. To debug this execute the LWUIT Designer from command line using “java -jar ResourceEditor.jar”, when entering the theme option you would see the stack trace of the exception that caused the failure.

Known Issues

There is currently a known issue in some operating systems which causes the LWUIT Designer to fail in some cases when using the Aero theme. This is an issue stemming from Java SE's look and feel implementation and the only workaround is to change the application look and feel using the “Look And Feel” menu option.