Skin System
SharpEnviro shell replacement

written by
Martin Krämer
martin@sharpenviro.com

September 14, 2010

Contents

1 Introduction

This document will help you to understand how the SharpEnviro skin system works so that you can create your own skins. Creating skins isn't very difficult so we want to encourage you to try it out and create some new cool skins for SharpEnviro. If you have created a new skin you can share it with other users in our forum or you can contact the developers so that maybe your skin will be included in the next release of SharpEnviro or a possible update packages.

How does the skin system work?

The skin system developed for SharpEnviro is a XML based bitmap layer-based rendering system. A skin defines a rendering path from the top-most bitmap layer to the bottom layer and in the end all those layers will be merged together. The layers however aren't simple windows bitmaps, we are using 32bit bitmaps as layers which fully support transparency and alpha blending. Therefore, all images used in a skin must be saved in PNG format with transparency (32bit PNG). We also recommend you save the PNG files with the normal RGB color palette (not grayscale!).

A SharpEnviro skin is divided into several skin components which together form the complete skin. Those components are skinned independently from each other and a valid SharpEnviro skin must contain skins for all of the components.

- Buttons
- Labels/Captions
- Edit Boxes
- Menu background
- Menu items
- Notification Popups
- Progress bars
- SharpBar background
- SharpBar mini config buttons
- Taskbar items
- Taskbar preview windows
- Menu Item

How is the coloring of the skins working?

One of the more exciting features of the skinning system is the support for color schemes. SharpEnviro supports the use of color schemes which allow the colors of certain skin parts to be changed by the user. Which parts of the skin can be controlled and colored is up to the skin developer. Besides the colors itself the scheme system also allows for other properties like the alpha value of the skin to be changed. You can for example add a scheme property which would enable the dynamic changing of the background transparency for any skin components. Adding support for schemes is not a required part of a skin, but supporting scheme colors and values in your skin is a great feature and highly recommended. With schemes, you control the skin, but the end-user can define the colors and final look of it.

How to get started?

The best way to get started on creating a new skin for SharpEnviro is to modify an already existing skin. Chose the skin that matches your ideas and visions for a new skin best and start to modify this skin. By doing so you can very quickly see results even of small changes. However we still recommend to read this entire document for getting a better understanding about how the skin system and especially the advanced features work.

2 Skin System

2.1 File Structure

Each skin gets its own directory within the *SharpEnviro\Skins* folder
SharpEnviro\Skins\MySkin

Within the directory of the skin 3 xml files are to be created
SharpEnviro\Skins\MySkin\info.xml
SharpEnviro\Skins\MySkin\scheme.xml
SharpEnviro\Skins\MySkin\skin.xml

Info.xml - Skin Header and Information File

This file is a very simple xml file which contains nothing more than the name of the skin, the authors name, authors website, info description and the version of the skin. An example file would look like this:

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <SharpESkinInfo>
3    <header>
4      <name>Number 8</name>
5      <author>Martin Krämer</author>
6      <url>http://www.sharpenviro.com</url>
7      <info>Big skin similiar to Windows 7</info>
8      <version>0.8</version>
9    </header>
10 </SharpESkinInfo>
```

Code Example 2.1: Header file for the skin (*info.xml*).

Scheme.xml - Scheme Color Definitions

The scheme color xml file defines which scheme color and scheme values will be available for the skin. Those are not only the values the user can change by adjusting their color schemes, but those are also the values and tags which a skin developer can use while creating a skin. When you create a skin it's recommended to take a look at this file first because you can use all the scheme tags created here later when creating the skin. A more detailed description of the scheme color system can be found here.

Skin.xml - Skin Components

This file contains the actual skins for all the components which can be skinned. The basic structure of the file is very simple, under the root xml element each skin component gets it's own element under which the actual skin informations are stored.

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <SharpESkin>
3    <SharpBar>...</SharpBar>
4    <Menu>...</Menu>
5    <MenuItem>...</MenuItem>
6    <TaskItem>...</TaskItem>
7    <MiniThrobber>...</MiniThrobber> <!-- SharpBar mini config buttons -->
8    <Button>...</Button>
9    <Font>...</Font> <!-- Captions and Labels -->
10   <ProgressBar>...</Progressbar>
11   <Edit>...</Edit> <!-- Edit Box -->
12   <Notify>...</Notify>
13   <TaskPreview>...</TaskPreview>
14 </SharpESkin>
```

Code Example 2.2: Basic structure of the file *skin.xml*.

The skin components itself can have different sub parts which represent different states or parts of that component. A Button for example can have a part which defines how it looks when it's clicked and how it will look when it's not clicked. Which skin component parts exist depends on the skin component and will be discussed later for each component in detail. The following example shows how a button skin is structured.

```
1  [...]
2    <Button>
3      <Text>...</Text> <!-- Text Settings -->
4      <Icon>...</Icon> <!-- Icon Settings -->
5
6      <Normal>...</Normal> <!-- Actual Skin for a button in its normal state -->
7      <Hover>...</Hover> <!-- Skin for a button when the mouse is over the button -->
8      <Down>...</Down> <!-- Skin for when a button is clicked or pressed down -->
9    </Button>
10 [...]
```

Code Example 2.3: Structure of the skin for the button skin component. The skin consists of three different sates: *Normal*,*Hover*,*Down*. Which stated is used for drawing the button is based on the status of the mouse.