Contents

JADE Skins

Introduction ................................................................. 4
Selecting a Skin for the Development Environment .................. 4
JadeSkinMaintenance Form ................................................. 5
JadeSkinSelection Form ....................................................... 7
JadeSkinMaint Form .......................................................... 8
JadeSkinSelect Form .......................................................... 9
  Exercise 1 – Creating a Button Control Skin ......................... 9
  Exercise 2 – Previewing a Skin ................................... 10
  Exercise 3 – Creating a ComboBox Control Skin .................. 11
Customizing Forms with JADE Skins ..................................... 13
  Exercise 4 – Adding a Forms Skin to an Application Skin ........ 16
  Exercise 5 – Adding a Menus Skin to an Application Skin ....... 17
JadeSkinRoot Class ............................................................ 18
  Exercise 6 – Adding an Application Skin to an Application ... 19
JADE Skins

Introduction

JADE skins enable you to customize of JADE form Graphical User Interface (GUI) elements. By configuring and saving colors, fonts, shapes, and sizes of the various controls on JADE forms, you can enforce a consistent look-and-feel across multiple applications.

Selecting a Skin for the Development Environment

As the JADE development environment is implemented in JADE, you can use JADE skins to customize the look and feel of the development environment in the same way you can for any other JADE application.

To select a different skin for your JADE development environment, select one of the provided skins in the Select JADE Skin drop-down list box on the Window sheet of the Preferences dialog (accessed by selecting the Preferences command from the Options menu).
Note  JADE2018 Cashmere is the default skin for versioned schemas.

If you use this skin for your standard JADE development environment, you should change the default skin for versioned schemas (on the Miscellaneous sheet of the Preference dialog).

JadeSkinMaintenance Form

The JadeSkinMaintenance form (the Jade Skin Maintenance dialog) is used to maintain existing skins and define new skins.

This form enables you to re-skin any number of controls, forms, and menus. These re-skins are then combined into an application skin, which can be applied to any number of applications.

To open the Jade Skin Maintenance dialog, define and run the following JadeScript createJadeSkinMaintenance method.

```jadescript
createJadeSkinMaintenance();

vars
  form : JadeSkinMaintenance;
begin
  create form transient;
  form.showModal();
epilog
  delete form;
end;
```

The first sheet of the Jade Skin Maintenance dialog is used to combine individual control re-skins into an application skin.

An application skin requires at least one Form or Control skin to be applied to it before it can be created.
As you create Form and Control skins, they are added to the lists of Available Form Skins and Available Control Skins.
To create a **Control** skin, select the **Controls** sheet of the Jade Skin Maintenance dialog.

From this form, you can select a variety of options to be applied to all instances of a specific control. For example, you can select the color and font of the text of the button’s label, the background color of the button, and the style of the border for the **Button** control type button. Alternatively, you can set an image for each of the parts of the **Button** control.

**JadeSkinSelection Form**

The **JadeSkinSelection** form (the Skin Selection dialog) is used to select the skin to be applied to the current application. The form can be created directly from the end-user application; for example, from a menu or the **click** event of a button. It can also be called from a JadeScript method, to preview what a skin will look like. The form can be created with the following code.

```javascript
createJadeSkinSelection();

vars
 form : JadeSkinSelection;
begin
 create form transient;
 form.showModal();
 epilog
 delete form;
end;
```
When viewing the Skin Selection dialog, the dialog itself is displayed using the selected skin. This allows the user to preview what the skin will look like before selecting a skin for use at run time.

In this simple example, the Example application skin has customized buttons with yellow text on a black background. The Skin Selection dialog displays its buttons in this style while Example is selected as the skin. Click Apply, to apply the Example skin to all application forms for the remainder of the application’s lifetime.

**JadeSkinMaint Form**

The JadeSkinMaint form (the Skin Maintenance dialog) was the precursor to the JadeSkinMaintenance form and it is retained in JADE for backwards compatibility.
Exercise 1 – Creating a Button Control Skin

In this exercise, you will create a simple application skin consisting of a few control skins.

1. Add the following JadeScript `createJadeSkinMaintenance` method to `BankingViewSchema`.

```jade
createJadeSkinMaintenance();
vars
    form : JadeSkinMaintenance;
begin
    create form transient;
    form showModal();
epilog
    delete form;
end;
```

2. Run the method, to open the Jade Skin Maintenance dialog.

3. Select the Controls sheet and then select Button in the Control Type combo box.

   **Tip** As the options in the following steps are examples only, feel free to customize the elements to your requirements.

4. Set BorderStyle to 3 - 3D raised.

5. Uncheck the Default backColor check box and select Black as the color.
6. Uncheck the **Default foreColor** check box and select **Light Gray** as the color.

7. Set the font to **Terminal** and **Font Size** to **9**.

8. Call the skin **btnDOS**. The form should then look like the following.

9. Click **Update**, to save the **Button** skin.

10. Select the **Applications** sheet and double-click the **btnDOS** skin to add it to the current skin.

11. Enter **DOS** as the Application skin name and then click **Update**, to save it.

**Exercise 2 – Previewing a Skin**

In this exercise, you will use the **JadeSkinSelection** form to preview the changes to buttons made to your skin.

1. Add the following JadeScript **createJadeSkinSelection** method to **BankingViewSchema**.

```javascript
createJadeSkinSelection();

vars
    form : JadeSkinSelection;
begin
    create form transient;
    form.showModal();
epilog
    delete form;
end;
```
2. Run the method defined in the previous step and select **DOS** from the **Choose Skin** combo box.

You should see the buttons displayed using the skin you have designed in the first exercise in this module.

---

**Exercise 3 – Creating a ComboBox Control Skin**

In this exercise, you will add a control skin to combo boxes in the **DOS** application skin.

1. Open the Jade Skin Maintenance dialog by running the JadeScript `createJadeSkinMaintenance` method.

2. Select the **Controls** sheet and then set the control type to **ComboBox**.

3. Set the **BorderStyle** to **2 - 3D sunken**.

4. Uncheck the **Default backColor** check box and select **Black** as the color.

5. Uncheck the **Default foreColor** check box and select **Light Gray** as the color.

6. Set the font to **Terminal** and the **Font Size** to **9**.

7. Call the skin **cbDOS**.
8. Click **Update**, to save the **ComboBox** skin changes.

9. Select the **Applications** sheet, add **cbDOS** to the **DOS** application skin, and then click **Update**.

10. Preview your changes in the **JadeSkinSelection** form as you did in the previous exercise.

The form should now look like the following.
Customizing Forms with JADE Skins

The **Forms** sheet of the Jade Skin Maintenance dialog is used to specify a forms skin, which applies to the forms of an application themselves, rather than the controls on that form.

<table>
<thead>
<tr>
<th>Group Box</th>
<th>Element to be Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form Menu Line Options</td>
<td>The menu line at the top of forms. The drop-down menus also use this skin unless a menu skin is defined on the <strong>Menu</strong> sheet of the Jade Skin Maintenance dialog and applied to forms in the <strong>Popup Menu</strong> combo box.</td>
</tr>
<tr>
<td>Caption Options</td>
<td>The title caption of the forms.</td>
</tr>
<tr>
<td>Buttons</td>
<td>The minimize, maximize, and close buttons of the forms.</td>
</tr>
</tbody>
</table>

The **Forms** sheet also has the following options that are not in group boxes.

<table>
<thead>
<tr>
<th>Option</th>
<th>Element to be Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Image</td>
<td>The border elements of the forms. You can set an image for each one, to fully customize the look of forms.</td>
</tr>
</tbody>
</table>
### Option | Element to be Modified
---|---
Inactive Image | The border elements of the forms when they do not have focus. You can set an image for each one, to fully customize the look of forms.
Default backColor? | When unchecked, allows you to set a new default background color for the forms.
Inner Image | Displays the selected image as the background for forms.
Is brush | Displayed on the Forms sheet only when inner image is set. Checking this check box causes the inner image to be tiled on forms.
Popup Menu | The drop-down and popup menus. To use this option, you must first create a menu skin on the Menus sheet of the Jade Skin Maintenance dialog.
Image Mask | Allows for a non-rectangular region mask image to be applied to the forms’ skin.

### Customizing a Menu with JADE Skins

The **Menus** sheet of the Jade Skin Maintenance dialog is used to apply a skin to any drop-down and popup menus on forms. Menu skins do not apply to the menu line of forms, but the menu line’s skin is used as the default menu skin until it is set.
The **Menus** sheet has the following options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BorderStyle</td>
<td>Controls the borders between menu items. You can choose <strong>none</strong>, <strong>single</strong>,</td>
</tr>
<tr>
<td></td>
<td><strong>3D sunken</strong>, <strong>3D raised</strong>, or <strong>use border images</strong>.</td>
</tr>
<tr>
<td>Set All / Set / Clear</td>
<td>Allows for the setting of image files as the borders between menu items</td>
</tr>
<tr>
<td></td>
<td>when <strong>BorderStyle</strong> is set to <strong>4 - use border images</strong>.</td>
</tr>
<tr>
<td>Font</td>
<td>Sets the font (typeface) of the text in menu items.</td>
</tr>
<tr>
<td>Default backColor</td>
<td>Sets the default background color for menu items.</td>
</tr>
<tr>
<td>Default backColor selected</td>
<td>Sets the default background color for how menu items are displayed when</td>
</tr>
<tr>
<td></td>
<td>they are selected.</td>
</tr>
<tr>
<td>Default foreColor</td>
<td>Sets the default text color for menu items.</td>
</tr>
<tr>
<td>Default disabled foreColor</td>
<td>Sets the default text color for disabled menu items.</td>
</tr>
<tr>
<td>Default selected foreColor</td>
<td>Sets the default text color for how menu items are displayed when they are</td>
</tr>
<tr>
<td></td>
<td>selected.</td>
</tr>
<tr>
<td>Draw flat selection</td>
<td>When checked, selected menu items are drawn flat regardless of the</td>
</tr>
<tr>
<td></td>
<td>border style that is selected. In addition, if the border style is set to **</td>
</tr>
<tr>
<td></td>
<td>none and multiple menu items are selected, no line is drawn between them.</td>
</tr>
<tr>
<td>Check mark image</td>
<td>Sets the image to be used for check marks in menu items.</td>
</tr>
<tr>
<td>Sub-menu arrow</td>
<td>Sets the image to be used to show that a menu item contains a submenu.</td>
</tr>
<tr>
<td>Separator image</td>
<td>Sets the image to be used for any separators between menu items.</td>
</tr>
</tbody>
</table>

The Pixel Spacings group box on **Menus** sheet has the following text boxes, which enable you to specify the number of pixels used for spacing.

<table>
<thead>
<tr>
<th>Check Box</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before checkmark</td>
<td><img src="image1.png" alt="Example" /></td>
</tr>
<tr>
<td>After checkmark</td>
<td><img src="image2.png" alt="Example" /></td>
</tr>
<tr>
<td>After picture</td>
<td><img src="image3.png" alt="Example" /></td>
</tr>
<tr>
<td>Before accelerator</td>
<td><img src="image4.png" alt="Example" /></td>
</tr>
<tr>
<td>Before right Arrow</td>
<td><img src="image5.png" alt="Example" /></td>
</tr>
<tr>
<td>Menu item height</td>
<td><img src="image6.png" alt="Example" /></td>
</tr>
</tbody>
</table>
Exercise 4 – Adding a Forms Skin to an Application Skin

In this exercise, you will add a forms skin to your DOS application skin.

1. Run the JadeScript createJadeSkinMaintenance method to open the Jade Skin Maintenance dialog.
2. Select the Forms sheet on the Jade Skin Maintenance dialog.
3. Set the Default backColor to Light Gray.
4. In the Form Menu Line Options group box, set the following values.
   a. Font to Terminal, size 9
   b. Default backColor to Black
   c. Default backColor selected to Dark Gray
   d. Default foreColor to Light Gray
   e. Default selected foreColor to White
   f. Check the Draw flat selection check box
5. In the Caption Options group box, set the following values.
   a. Top margin to 5
   b. Check the Center check box
   c. Font to Terminal, size 9, bold
6. Save the forms skin as frmDOS.
7. Preview the skin using the JadeScript `JadeSkinSelection` method. It should look like the following.

![Select or Cancel a Skin](image)

**Exercise 5 – Adding a Menus Skin to an Application Skin**

In this exercise, you will customize the menus skin for the DOS application skin.

1. Run the JadeScript `createJadeSkinMaintenance` method to open the Jade Skin Maintenance dialog.
2. Select the Menus sheet on the Jade Skin Maintenance dialog.
3. Set the BorderStyle to 0 - none.
4. In the Pixel Spacings group box, set the values shown in the following image.

![Pixel Spacings](image)

5. Set the Font to Terminal, size 9.
6. Set the Default backColor to Light Gray.
7. Set the Default backColor selected to Dark Gray.
8. Set the Default foreColor to Black.
9. Set the Default selected foreColor to White.
10. Check the Draw flat selection check box.
11. Save the menus skin as mnuDOS.
12. On the Forms sheet of the Jade Skin Maintenance dialog, set the Popup Menu option to mnuDOS.
13. Preview the skin using the JadeScript JadeSkinSelection method. It should look like the following.

![Select or Cancel a Skin](image)

### JadeSkinRoot Class

JadeSkinRoot is the root object that contains collections of all skins (including Application skins as well as the Control and Forms skins that make up an application skin).

As JadeSkinRoot is a root object that uses the singleton pattern, it can be accessed safely with JadeSkinRoot.firstInstance.

---

**Note** The singleton pattern that is used for root objects means that that “There can be only one” is enforced. For more details, see Module 7 - Root Object of the JADE Developer’s course.
The **JadeSkinRoot** class can be used to find specific skins with a dictionary lookup by name, as shown in the following example.

```pascal
initialize() updating;

vars
    jsRoot     : JadeSkinRoot;
    exampleSkin : JadeSkinApplication;

begin
    jsRoot := JadeSkinRoot.firstInstance;
    exampleSkin := jsRoot.allApplicationSkins["Example"];  
    app.setApplicationSkin(exampleSkin);
end;
```

This method sets the application's skin to the skin called **Example** when the application is initialized.

**Exercise 6 – Adding an Application Skin to an Application**

In this exercise, you will apply the **DOS** application skin to the **Banking** application.

1. Open **BankingModelSchema** in the Class Browser.
2. Navigate to the **BankingModelSchema** application subclass.
3. Add a method called **applySkin** to the **BankingModelSchema** application class and code it as follows.

```pascal
applySkin();

vars
    jsRoot : JadeSkinRoot;
    skin   : JadeSkinApplication;

begin
    jsRoot := JadeSkinRoot.firstInstance;
    skin  := jsRoot.allApplicationSkins["DOS"];  
    app.setApplicationSkin(skin);
end;
```
4. Modify the initialize method in the BankingModelSchema application class to call the applySkin method, as follows.

```javascript
initialize() updating:
begin
  self.applySkin();
  on Exception do self.genericExceptionHandler(exception) global;
  self.myBank := Bank.firstInstance();
  if self.myBank = null then
    beginTransaction;
    create myBank persistent;
    commitTransaction;
  endif;
end;
```

5. Run the Banking application and open the Add Customer dialog. The Banking system should have the DOS application skin applied to it.