# Oracle® Fusion Middleware

Skin Editor User's Guide for Oracle Application Development Framework

11*g* Release 2 (11.1.2.1.0)

**E17456-04**

November 2011

Documentation for Oracle Application Development Framework (Oracle ADF) developers and user interface designers that describes how to create and apply skins to an application using the ADF Skin Editor.

ORACLE

Oracle Fusion Middleware Skin Editor User's Guide for Oracle Application Development Framework 11*g* Release 2 (11.1.2.1.0)

E17456-04



Primary Author: Walter Egan

Contributing Author: Robin Merrin, Odile Sullivan-Tarazi, and Robin Whitmore

Contributor: Laura Akel

# Contents

## 4 Creating the Source Files for an ADF Skin



## 5 Working with Component-Specific Selectors

## 10 Applying the Finished ADF Skin to Your Web Application



## 11 Advanced Topics

# Preface

Welcome to the *Skin Editor User's Guide for Oracle Application Development Framework*.

## Audience

This document is intended for application developers and user interface designers who want to change the look and feel of their application by skinning ADF Faces Rich Client components.

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

### Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

## Related Documents

For more information, see the following documents for the release that pertains to the application that you are skinning:

- *Oracle Fusion Middleware Installation Guide for Oracle Application Development Framework Skin Editor*
- *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework*
- *Oracle Fusion Middleware Tag Reference for Oracle ADF Faces*
- *Oracle Fusion Middleware Tag Reference for Oracle ADF Faces Skin Selectors*
- *Oracle Fusion Middleware Data Visualization Tools Tag Reference for Oracle ADF Faces*
- *Oracle Fusion Middleware Data Visualization Tools Tag Reference for Oracle ADF Skin Selectors*

# Conventions

The following text conventions are used in this document:

| Convention | Meaning |
|---|---|
| **boldface** | Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary. |
| *italic* | Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values. |
| `monospace` | Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter. |

# What's New in This Guide for Release 11.1.2.1.0

For 11*g* Release 2 (11.1.2.1.0), this guide has been updated in a number of ways. The following table lists the sections that have been added or changed.

For changes made to Oracle JDeveloper and Oracle Application Development Framework (Oracle ADF) for this release, see the What's New page on the Oracle Technology Network at `http://www.oracle.com/technetwork/developer-tools/jdev/documentation/index.html`.

| Sections | Changes Description |
|---|---|
| **Chapter 1, "About Skinning a Web Application"** | |
| Chapter 1.2, "Using the ADF Skin Editor for Different Releases of Oracle ADF." | Added section to elaborate on how you use the ADF Skin Editor to create ADF skins for different releases of Oracle ADF by choosing a value from the Target Application Release dropdown list. |
| **Chapter 8, "Working With Global Selector Aliases"** | |
| Section 8.4.3, "What You May Need to Know About Applying a Global Selector Alias." | Added section to elaborate on how the ADF skinning framework determines what style property to apply when it overrides the value of a global selector alias from an ADF skin that is extended or within a local ADF skin. |
| **Chapter 11, "Advanced Topics"** | |
| Section 11.3, "Configuration Files for an ADF Skin." | Revised section to describe a context initialization parameter (`MAX_SKINS_CACHED`) that specifies the maximum number of unique ADF skins (for example, `fusion` or `fusionFx-simple`) for which you store information in memory about the generated CSS files. |

# 1 About Skinning a Web Application

This chapter introduces you to skinning, the ADF skinning framework, and the ADF skins that Oracle ADF provides to apply to your Fusion web application or to help you get started creating your own ADF skin. It also describes how you use the ADF Skin Editor to create ADF skins for applications developed using various releases of Oracle ADF.

This chapter includes the following sections:



## 1.1 Introduction to Skinning a Web Application

*Skinning* refers to the task of developing an ADF skin to apply to a web application that uses ADF Faces and ADF Data Visualization components in the user interface. An ADF skin is a special type of cascading style sheet (CSS) that allows you to customize the appearance of these components. Instead of providing a CSS file for each component, or inserting a style sheet on each page of the application, you create one ADF skin for the web application. Every component that renders in the user interface automatically uses the styles defined by the ADF skin. This means you do not have to make design-time changes to individual pages to change their appearance when you use an ADF skin.

Using an ADF skin also makes it easy for you to maintain a consistent appearance for all the pages that the application renders. Changes to the appearance of your application can easily be made should you decide to do so. You might decide, for example, to change colors to make your application adhere to your company's corporate brand. Additionally, you may want to define a style property for a component to make your application more usable. For example, Figure 1–1 shows an ADF Faces `inputText` component.

***Figure 1–1 Writable inputText Component***

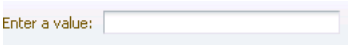
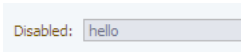


Figure 1–2 shows another ADF Faces `inputText` component where the background color is grayed out by the ADF skin to indicate to the end user that the `inputText` component is read only.

***Figure 1–2 Read-Only inputText Component with Grayed-Out Background Color***




Other benefits of skinning include the ability to easily change the default text labels that ADF Faces components render at runtime. For example, the default text for the `dialog` component's labels are **OK** and **Cancel** if you set the component's `type` property to `okCancel`. You cannot modify the values of these labels by specifying properties for the `dialog` component. Instead, if you want to change **OK** to **Submit**, for example, you make changes in the ADF skin that references a resource bundle with the string value, `Submit`. For more information, see Chapter 7, "Working With Text in an ADF Skin."

The previous examples illustrate some of the use cases for ADF skins plus the benefits of creating an ADF skin. Note that you do not have to define all the changes that you want for your application in one ADF skin. You can create different ADF skins to serve different purposes. For example, you might create ADF skins with different color schemes to adhere to the corporate brand of different companies. In addition, you can configure an application so that end users can dynamically change the ADF skin at runtime.

Note that this guide makes the following assumptions:

- You are familiar with the ADF Faces and ADF Data Visualization components that you can skin. The usage and functionality of these components is beyond the scope of this guide. For more information about these components, see the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* (for the release that pertains to the application you are skinning).
- You are familiar with CSS. It is beyond the scope of this guide to explain CSS. For extensive information about CSS, including the official specification, visit the World Wide Web Consortium (W3C) web site at:

  `http://www.w3.org/`

## 1.2 Using the ADF Skin Editor for Different Releases of Oracle ADF

You can use the ADF Skin Editor to create ADF skins for Fusion web applications developed using various releases of Oracle ADF. You choose the release of Oracle ADF for which you want to create an ADF skin by selecting a supported release from the Target Application Release dropdown list. You do this when you create an ADF skin project, as described in Section 4.2, "Creating ADF Skin Applications and ADF Skin Projects." The ADF Skin Editor configures the ADF skin project when you choose a release from the Target Application Release dropdown list so that only appropriate features to the Oracle ADF release you choose appear in the ADF skin project. Examples of configuration changes include the following:

- Filters the list of ADF skins that you can extend from to display only those ADF skins supported by the release you choose. For example, if you choose to create an ADF skin for 11*g* Release 1 (11.1.1.5.0), the ADF Skin Editor does not display the `fusionFx-v2` ADF skin because this ADF skin was introduced in 11*g* Release 2 (11.1.2).

- Selectors that a particular release of Oracle ADF does not support do not appear in the Selector Tree. For example, if you target a release earlier than 11*g* Release 2 (11.1.2), the ADF Skin Editor will not display selectors for the ADF Data Visualization Thematic Map component because this component was first introduced in 11*g* Release 2 (11.1.2).

  **Tip:** Press the F1 key on a selector in the Selector Tree to display the tag reference information for the release that you target.

As of 11*g* Release 2 (11.1.2), Oracle JDeveloper also provides a visual editor with the capability to create ADF skins for Oracle ADF applications created using that release. You can create an ADF skin within the project for a Fusion web application, as described in Section 4.4.2, "How to Create an ADF Skin in JDeveloper." The visual editors in JDeveloper and in the ADF Skin Editor offer the same functionality so the procedures described in this guide can be used to carry out tasks in both visual editors.

**Note:** If you want to create an ADF skin for an application developed using a release of Oracle ADF earlier than 11*g* Release 2 (11.1.2), use the ADF Skin Editor.

## 1.3 Overview of Developing an ADF Skin

Developing an ADF skin is an iterative process. Before you proceed, familiarize yourself with the concepts of CSS plus the ADF Faces and ADF Data Visualization components. The high level steps to develop an ADF skin are:

1. Create a source file for the ADF skin.

   You create a source file where you write the declarations for the selectors. When creating a source file using the visual editor in JDeveloper or the ADF Skin Editor, you must choose an existing ADF skin to extend from. If this ADF skin is the first ADF skin that you create, you choose from one of the ADF skins that Oracle ADF provides. For more information, see Section 11.4, "ADF Skins Provided by Oracle ADF." For information about the inheritance relationship between these ADF skins, see Section 1.5, "Inheritance Relationship of the ADF Skins Provided by Oracle ADF." If you create subsequent ADF skins, you can choose to extend from an ADF skin that you created previously.

   For more information about creating an ADF skin, see Section 4.4, "Creating an ADF Skin File."

2. Define values for the ADF skin properties and CSS properties. You can write values for these properties using the selectors exposed by the ADF Faces and ADF Data Visualization components through the ADF skinning framework.

   For a description of the different categories of selectors, rules, and pseudo-elements, see Chapter 2, "Working with ADF Skin Selectors."

   For information about how to define values for the selectors exposed by the ADF Faces and ADF Data Visualization components, see Section 5, "Working with Component-Specific Selectors."

3. If applicable, import images that you want your ADF skin to reference at runtime in the Fusion web application. For more information, see Chapter 6, "Working with Images in Your ADF Skin."

**Tip:** If you choose to extend an ADF skin from the Fusion Simple family of ADF skins, described in Section 11.4, "ADF Skins Provided by Oracle ADF," you can use the Images window to import and edit images in the project for your ADF skin. For more information about the Images window, see Section 6.3, "Working with the Images Window."

4. If applicable, override the default text labels defined for the ADF Faces and ADF Data Visualization components by entering new values in a resource bundle. For more information, see Chapter 7, "Working With Text in an ADF Skin."
5. If applicable, edit or create themes in your ADF skin. Themes are a way of implementing a look and feel at a component level. For more information, see Section 5.6, "Applying Themes to ADF Faces Components."
6. Preview and test the changes that you made to the ADF skin to verify that the results are what you want. Modify the ADF skin as necessary. For more information about previewing and testing an ADF skin, see Section 10.2, "Testing Changes in Your ADF Skin."
7. Once you complete development of the ADF skin, you may want to package it for distribution. For more information, see Section 10.3, "Packaging an ADF Skin into an ADF Library JAR."
8. Having completed the ADF skin and distributed it, you configure your Fusion web application so that it uses it. For more information, see Section 10.4, "Applying an ADF Skin to Your Web Application."

**Tip:** Consider completing the *Changing an Application's Look and Feel by Using Skins* tutorial that is available on the Oracle Technology Network (OTN). This tutorial provides a step-by-step guide to developing an ADF skin using the visual editor for ADF skins in JDeveloper. The tutorial also provides an accompanying sample application for you to use. For more information, navigate to: `http://www.oracle.com/technetwork/developer-tools/adf/overview/index.html`.

## 1.4 Taking a Look at an ADF Skin

An ADF skin is a type of cascading style sheet. It differs from a cascading style sheet in a number of ways. One way it differs is that you can specify properties for the selectors that the ADF skinning framework exposes in the source file for the ADF skin. A selector exposed by the ADF skinning framework is similar to a CSS selector in that it identifies the ADF Faces and ADF Data Visualization components for which you want to change the appearance and allows you to specify one or more style properties for the component.

A selector exposed by the ADF skinning framework differs from a CSS selector in that it allows you to set values both for CSS properties and ADF skin properties exposed by the ADF skinning framework. CSS properties are interpreted directly by the end user's browser. ADF skin properties are prefaced by the characters `-tr-`. Some of these ADF skin properties are read and interpreted by the Fusion web application. These properties are also known as *server-side properties*. A component that renders in the user interface may read these properties before it decides what to render. Other types of ADF skin properties, for example `-tr-rule-ref` or `-tr-property-ref`, enhance the capabilities of the ADF skinning framework, as described in Section 2.3, "Properties in the ADF Skinning Framework."

Example 1–1 shows the selector for the `gauge` component that sets values for the ADF skin properties `-tr-graphic-antialiasing` and `-tr-animation-indicators`, plus the CSS properties `background-color` and `font-family`.

***Example 1–1 Gauge Component's Selector with ADF Skin and CSS Properties***

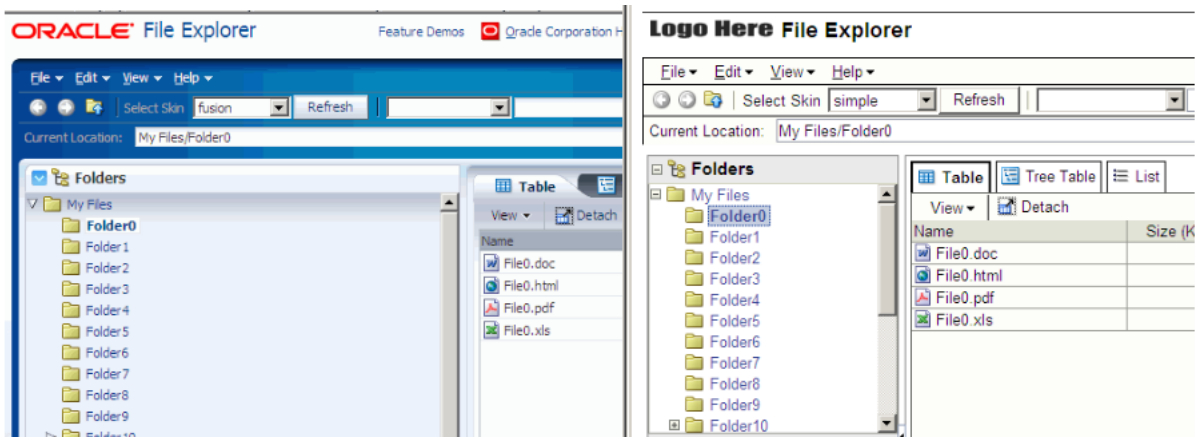
```
af|dvt-gauge
{
  /** ADF skin properties */
  -tr-graphic-antialiasing: false;
  -tr-animation-indicators: none;
  /** CSS properties */
  font-family: Geneva, Arial, Helvetica, sans-serif;
  background-color: rgb(243,255,185);
}
```

As Example 1–1 demonstrates, you can set values for CSS properties and ADF skin properties within the declaration of a selector exposed by the ADF skinning framework. The ADF skinning framework exposes the ADF skin properties that you can define. In addition to ADF skin properties, the ADF skinning framework defines a number of pseudo classes and rules that you can specify in an ADF skin. Examples of supported rules and pseudo classes include `@platform`, `@agent`, `@accessibility-profile`, `:rtl`, and `@locale`. For more information, see Chapter 2, "Working with ADF Skin Selectors."

At runtime, the Fusion web application reads the source file for the ADF skin, converts the entries in the ADF skin to style classes that it dynamically adds to the generated HTML output for an ADF Faces component.

Figure 1–3 demonstrates the impact that an ADF skin can have on the appearance of an application's page. The page on the left renders using the `fusion` ADF skin. The page on the right renders using the `simple` ADF skin. Each ADF skin defines values for colors and fonts. The `fusion` ADF skin uses many more colors, in addition to referencing an image for the Oracle logo.

***Figure 1–3 File Explorer Application Using the Fusion ADF Skin and the Simple ADF Skin***

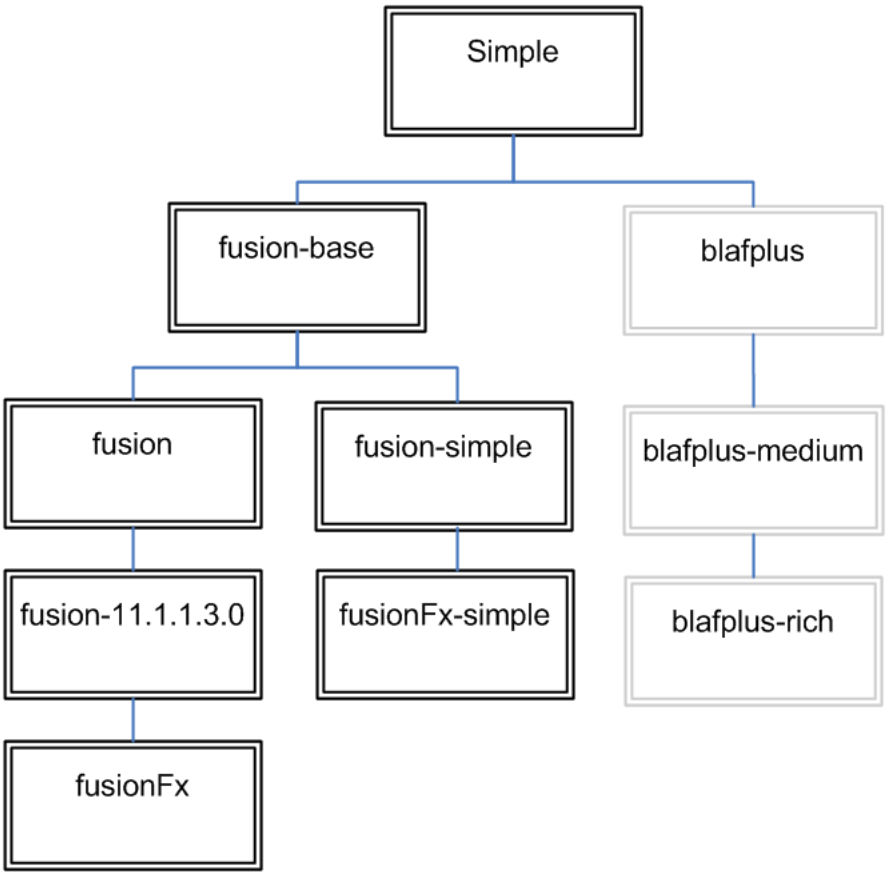



**Note:** An ADF skin can affect the time it takes a client to render the user interface. The more styles that an ADF skin uses, the more the client has to load. This can affect performance in low bandwidth or high latency environments.

## 1.5 Inheritance Relationship of the ADF Skins Provided by Oracle ADF

Oracle ADF provides a number of ADF skin families that you can use in your application or extend when you create an ADF skin. The ADF skins provided by Oracle ADF offer increasing levels of customization for the appearance rendered by ADF Faces and ADF Data Visualization components at runtime. Figure 1–4 illustrates the inheritance relationship between the different ADF skin families where, for example, the `fusion-base` ADF skin inherits the style properties defined in the `simple` ADF skin in addition to defining more style properties. All ADF Faces components use, at a minimum, styles defined in the `simple` ADF skin as this is the skin from which the others extend from. The `simple` ADF skin defines the minimum style properties that ADF Faces components require to render in an ADF application. If you want to create an ADF skin with a minimal amount of customization, you create an ADF skin that extends from the `simple` ADF skin. If you want to create an ADF skin that resembles the Fusion family of ADF skins but is easier to modify (because it uses fewer colors and fonts), consider creating an ADF skin that extends from the Fusion Simple family of ADF skins.

***Figure 1–4 Inheritance Relationship of ADF Skin Families Provided by Oracle ADF***




**Note:** The Blafplus family of ADF skins shown in Figure 1–4 is deprecated.

You can apply any of the ADF skins in Figure 1–4 or an ADF skin that you create yourself to an application. For more information, see Section 10.4, "Applying an ADF Skin to Your Web Application."

For a more detailed description of the ADF skins that Oracle ADF provides, see Section 11.4, "ADF Skins Provided by Oracle ADF."

# 2

# Working with ADF Skin Selectors

This chapter describes the ADF skin selectors. These selectors along with pseudo-elements, pseudo-classes, ADF skin properties and ADF skinning framework rules allow you to customize the appearance of ADF Faces and ADF Data Visualization components.

This chapter includes the following sections:

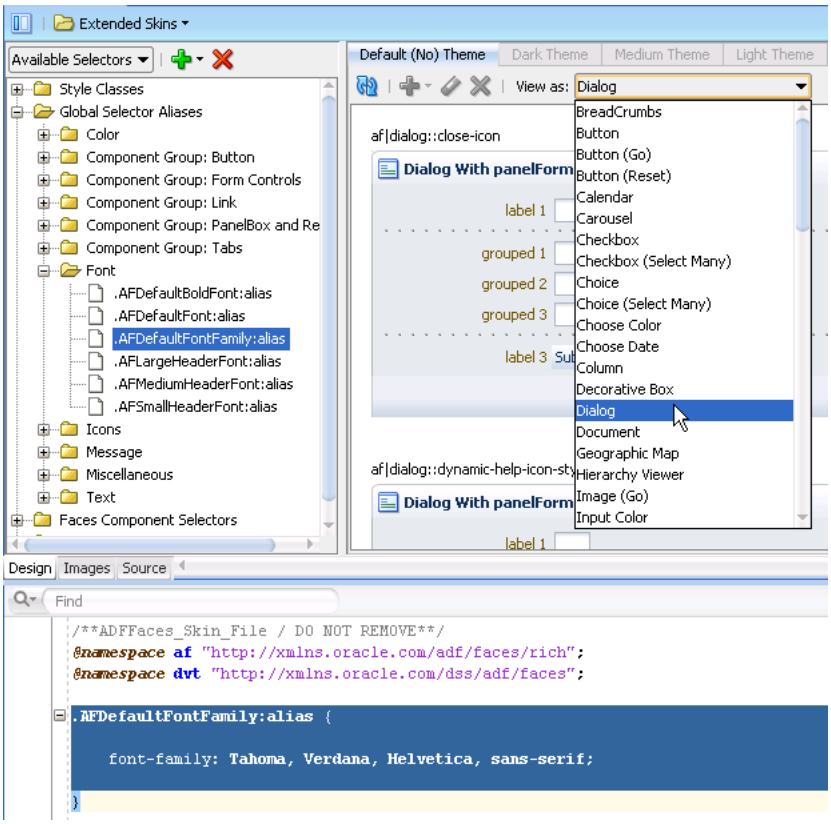


## 2.1 About ADF Skin Selectors

The ADF skinning framework provides a range of selectors that you can specify in an ADF skin to customize the appearance of ADF Faces and ADF Data Visualization components. There are two types of selectors: global selectors and component-specific selectors. A global selector defines style properties that you apply to one or more selectors. A component-specific selector defines style properties that apply to one selector.

The ADF skins provided by Oracle ADF define many global selectors (**Global Selector Aliases** in the user interface of the visual editor) that many ADF Faces components inherit. These ADF skins do not define global selectors inherited by the ADF Data Visualization components. For example, many ADF Faces components use the `.AFDefaultFontFamily:alias` selector to specify the font family. If you create an ADF skin that overrides this selector by specifying a different font family, that change affects all the components that have included the `.AFDefaultFontFamily:alias` selector in their selector definition. Figure 2–1 shows the `.AFDefaultFontFamily:alias` selector in the source view and the design view. The **View as** list displays the current list of ADF Faces components that use the value defined in the `.AFDefaultFontFamily:alias` global selector alias to determine what font family to use.

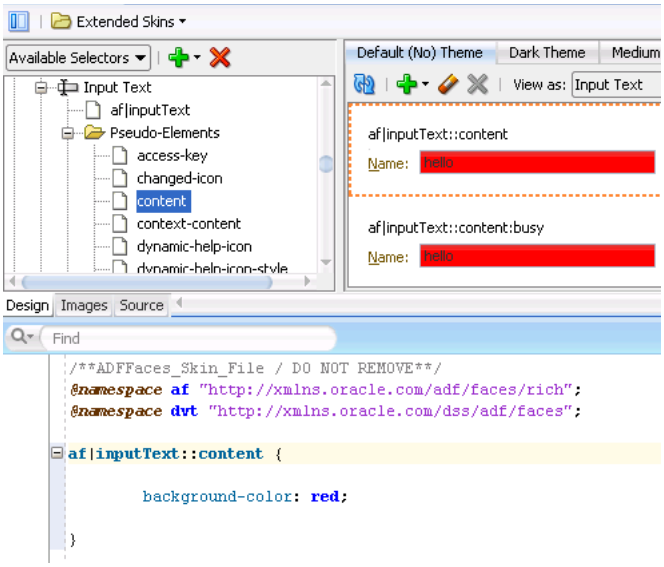
***Figure 2–1 The .AFDefaultFontFamily:alias Global Selector Alias***




An ADF skin that you create inherits the global selector aliases defined in the ADF skin that it extends from. You can also create new global selector aliases in your ADF skin files. For more information, see Chapter 8, "Working With Global Selector Aliases."

Component-specific selectors are selectors that the ADF skinning framework exposes that allow you to identify the corresponding ADF Faces and ADF Data Visualization components for which you can define style properties. For example, Figure 2–2 shows the selector for the ADF Faces `inputText` component in the source editor and visual editor. A value of `red` has been set for the CSS `background-color` property in the `content` pseudo-element exposed by this component's selector (`af|inputText`).

***Figure 2–2 inputText Component's Skin Selector***

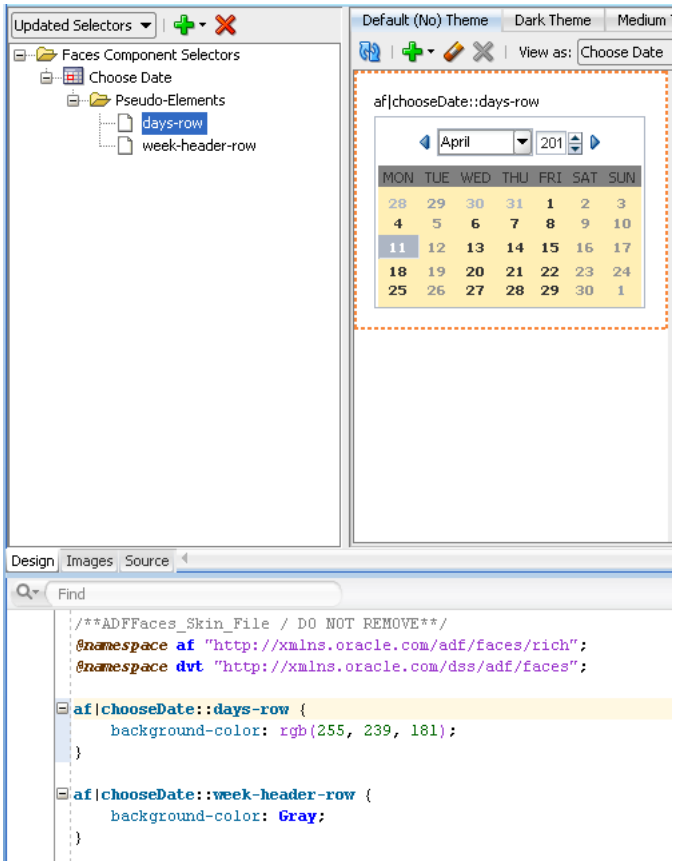



For more information about the component-specific selectors, see Chapter 5, "Working with Component-Specific Selectors." For more information about global selector aliases, see Chapter 8, "Working With Global Selector Aliases."

### 2.1.1 ADF Skin Selectors and Pseudo-Elements

Many ADF skin selectors expose pseudo-elements. A pseudo-element denotes a specific area of a component for which you can define style properties. Pseudo-elements are denoted by a double colon followed by the portion of the component the selector represents. For example, Figure 2–3 shows how the `days-row` pseudo-element exposed by the `af|chooseDate` selector allows you to configure style properties for the appearance of the calendar grid.

*Figure 2–3 Pseudo-Elements for the Choose Date Component*

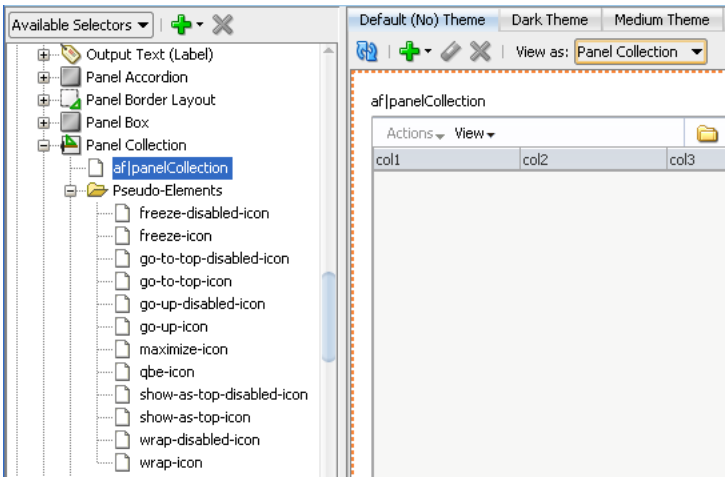



## 2.1.2 ADF Skin Selectors and Icon Images

ADF Faces components that render icons do so using a set of base icon images. No CSS code entries appear in the source file of the ADF skin for these icon images in contrast to, for example, the CSS code entries that appear in a source file when you specify an image as a value for the CSS `background-image` property. Instead, the ADF skinning framework registers the icon image for use with the renderer. For more information about the renderer and supported render kits, see Section 11.2, "ADF Skinning Framework and Supported Render Kits."

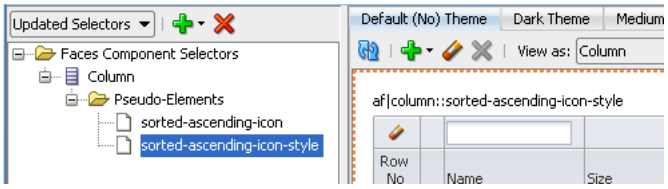
ADF skin selectors use two naming conventions for pseudo-elements that identify icon images that render in a component. The names of these pseudo-elements end in `-icon` or in `icon-style`. For example, the Panel Collection selector's pseudo-elements end in `-icon`, as shown in Figure 2–4.

***Figure 2–4 Panel Collection Pseudo-Elements for Icon Images***




In contrast, the Column selector (`af|column`) defines pseudo-elements that end in both `-icon` and `-icon-style`, as shown in Figure 2–5.

***Figure 2–5 Column Pseudo-Elements for Icons***




In Figure 2–5, the `sort-ascending-icon-style` pseudo-element styles the icon used for the sort ascending indicator in the column selector. This pseudo-element specifies the icon as a background image. Use the `:hover` and `:active` pseudo-classes to customize the appearance. For example, write the following syntax to make the background red if the end user hovers the mouse over the sort ascending indicator:

```
af|column::sort-ascending-icon-style:hover
{
   background-color: Red;
}
```

> **Tip:** Many browsers do not render background images when in accessibility mode. The following example demonstrates how you can work around this behavior if you want your application to display an image when in accessibility mode.

If you want to use your own image rather than the icon specified as a background image, you need to first prevent the background image from rendering. You do this by specifying the `-tr-inhibit` ADF skin property for the `sort-ascending-icon-style` pseudo-element as follows:

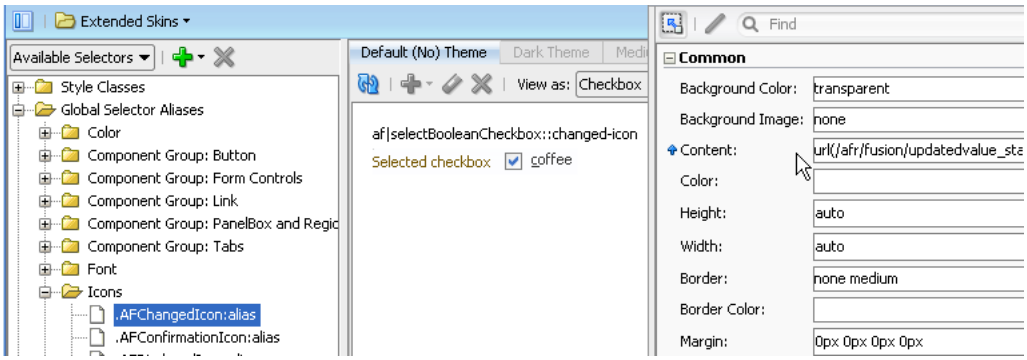
```
af|column::sort-ascending-icon-style
{
  -tr-inhibit: background-image;
}
```

Next you specify the text or image that you want to render as a value for the `content` property of the `sort-ascending-icon` selector. For example, write syntax similar to the following to specify an alternative image:

```
af|column::sort-ascending-icon
{
    content:url("images/arrow-up.jpg");
    width: 10px;
    height: 10px;
}
```

The ADF skinning framework also defines a number of global selector aliases that specific icon images to use in different scenarios. These global selector aliases appear under the **Icons** node in the Selector Tree, as shown in Figure 2–6. The `.AFChangedIcon:alias` shown in Figure 2–6 enables you to globally set the changed icon for all components using that icon.

***Figure 2–6 Global Selector Aliases for Icons***



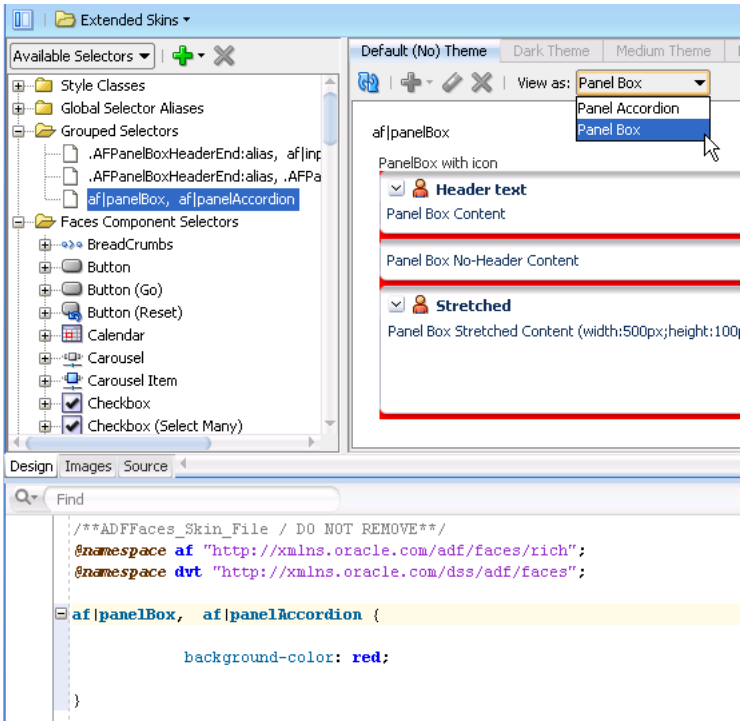

For more information, see Chapter 6, "Working with Images in Your ADF Skin."

## 2.1.3 Grouped ADF Skin Selectors

You can group ADF skin selectors and global selector aliases to minimize the number of entries in the source file of the ADF skin. The selectors that you group render under the **Grouped Selectors** node in the Selector Tree, as shown in Figure 2–7. The **View as** list in the Preview Pane displays all the selectors that you grouped.

As the visual editor does not provide a way to specify grouped selectors, you use the source editor to specify the selectors and/or global selector aliases that you want to put in a grouped selector. Separate each selector by a comma ( , ) to include in the grouped selector.

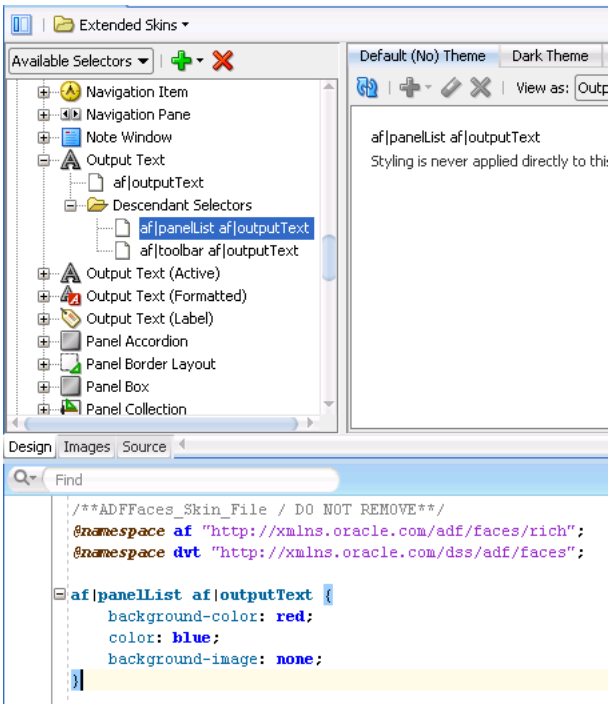
*Figure 2–7 Grouped Selectors in the Selector Tree*




## 2.1.4 Descendant ADF Skin Selectors

A descendant selector is made up of two or more selectors separated by white space. You can configure descendant selectors for ADF skin selectors. These allows you to configure style properties for specific selectors when they render within another selector. When you configure a descendant selector, the visual editor displays a **Descendant Selectors** node under the selector included in the descendant selector, as shown in Figure 2–8.

***Figure 2–8 Descendant Selectors in the Selector Tree***




As the visual editor does not provide a way to specify descendant selectors, you use the source editor to specify the selectors and/or global selector aliases that you want to specify in a descendant selector. Separate each selector by a white space.

## 2.2 Pseudo-Classes in the ADF Skinning Framework

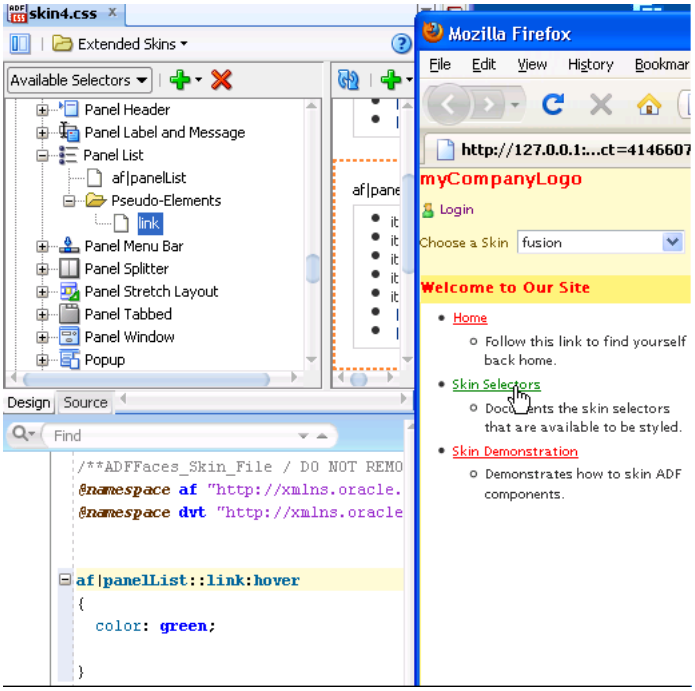
The CSS specification defines pseudo-classes, such as `:hover` and `:active`, which are used to define style properties for when a selector is in a particular state. You can apply these pseudo-classes to almost every ADF Faces component. In addition, the ADF skinning framework provides additional pseudo-classes for specialized functions. Examples include pseudo-classes to apply when a browser's locale is a right-left language (`:rtl`) or for drag and drop operations (`:drag-target` and `:drag-source`). The syntax that appears in the source file of an ADF skin to denote a pseudo-class uses the following format(s):

```
adfskinselector:pseudo-class
```

```
adfskinselector::pseudo-element:pseudo-class
```

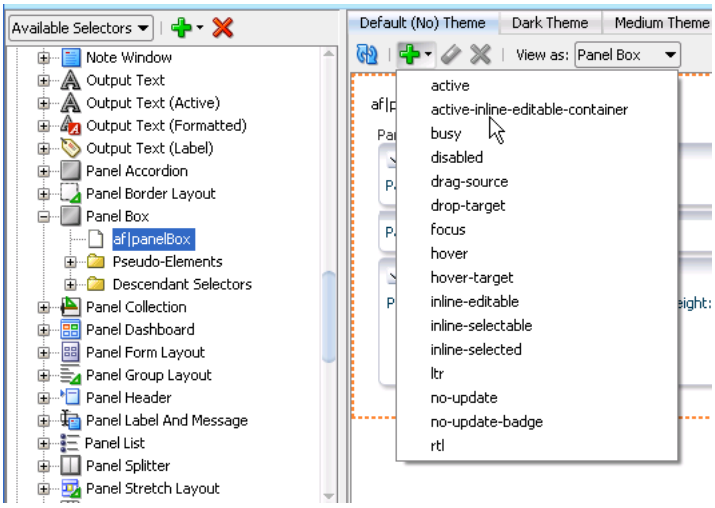
Figure 2–9 shows the syntax that you write in the source file of an ADF skin for the `:hover` pseudo-class so that a `panelList` component's link renders green when an end user hovers a mouse over the link in Figure 2–9.

***Figure 2–9 Pseudo-Class Syntax and Runtime Behavior for a Panel List Link***




Some components make more use of pseudo-classes than other components. The `panelBox` component's selector, for example, makes extensive use of pseudo-classes to define its appearance when it is in various states (for example, active, disabled, rendering right to left). Figure 2–10 shows the list of available pseudo-classes that renders when you select the `panelBox` component's selector in the Selector Tree and click the **Add Pseudo-Class** icon to display the list of available pseudo-classes.

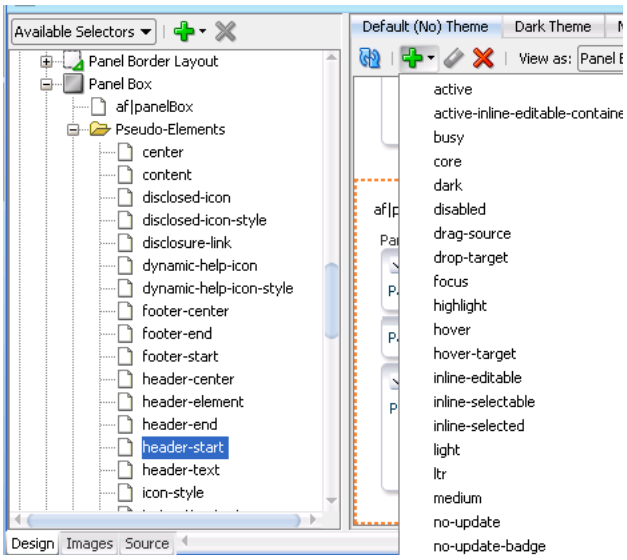
***Figure 2–10 Pseudo-classes for the panelBox Component's Selector***




Pseudo-classes can also be applied to pseudo-elements that selectors expose. The `panelBox` component selector's pseudo elements are a good example. Figure 2–11 shows the list of pseudo-classes that the `header-start` pseudo-element exposed by the `panelBox` component selector accepts. Many of these pseudo-classes allow you to define the appearance for the `panelBox` component depending on the value that the application developer sets for its attributes. For example, the `core` and `highlight`

pseudo-classes define the corresponding appearance when the application developer sets the `panelBox` component's `ramp` attribute to `core` or `highlight`.

***Figure 2–11 Pseudo-classes for the header-start pseudo-element***




The following are common pseudo-classes used by ADF Faces selectors.

- **Drag and drop**: The two pseudo-classes available are `:drag-source` applied to the component initiating the drag and removed once the drag is over, and `:drop-target` applied to a component willing to accept the drop of the current drag.
- **Standard**: In CSS, pseudo-classes like `:hover`, `:active`, and `:focus` are considered states of the component. This same concept is used in applying skins to components. Components can have states like `read-only` or `disabled`. When states are combined in the same selector, the selector applies only when all states are satisfied.
- **Right-to-left**: Use this pseudo-class to set a style or icon definition when the browser is in a right-to-left language. Another typical use case is asymmetrical images. You will want the image to be flipped when setting skin selectors that use the image in a right-to-left reading direction. Be sure to append the `:rtl` pseudo-class to the very end of the selector and point it to a flipped image file. For example, the end image of the `panelBox` component will be the `panelBoxStart.png` file when the browser is set to right-to-left. The `panelBox` end image in right-to-left is the same as the flipped left-to-right `panelBox` start image.

  ```
  af|panelBox::medium af|panelBox::top-end:rtl {
            background-image: url(/skins/purple/images/panelBoxStart.png);
            width:8px;
            height:8px
  }
  ```

  You can also use `:rtl` to apply to skin icons. For more information, see Chapter 6, "Working with Images in Your ADF Skin."
- **Inline editing**: This pseudo-class is applied when the application activates a component subtree for editing in the browser. For example, `:inline-selected`

is a pseudo-class applied to currently selected components in the active inline-editable subtree.

- **Message**: This pseudo-class is used to set component-level message styles using CSS pseudo-classes of `:fatal`, `:error`, `:warning`, `:confirmation`, and `:info`. For more information, see Section 5.5, "Configuring ADF Skin Properties to Apply to Messages."

> **Note:** The global selector aliases that appear in the Selector Tree are a special type of pseudo-class (`:alias`). For more information, see Chapter 8, "Working With Global Selector Aliases."

## 2.3 Properties in the ADF Skinning Framework

The ADF skinning framework defines a number of ADF skin properties. The Fusion web application, rather than the user's browser, interprets ADF skin properties. When configured, ADF skin properties enable you to do the following:

- Suppress styles defined in an ADF skin with the `-tr-inhibit` skin property.

  Suppress or reset CSS properties inherited from a base skin with the `-tr-inhibit` skin property. For example, the `-tr-inhibit:padding` property will remove any inherited padding. Remove (clear) all inherited properties with the `-tr-inhibit:all` property. The suppressed property name must be matched exactly with the property name in the base skin

- Reference styles from other selectors with the `-tr-rule-ref` property.

  Create your own global selector alias and combine it with other selectors using the `-tr-rule-ref` property. For more information, see Section 8.2, "Creating a Global Selector Alias."

- Reference the value of a property defined in another selector using the `-tr-property-ref` property.

  For more information, see Section 8.5, "Referencing a Property Value from Another Selector."

- Configure the themes for child components with the `-tr-children-theme` property.

  For more information, see Section 5.6, "Applying Themes to ADF Faces Components."

- ADF skin selectors with style properties.

  Skin style properties allow you to customize the rendering of a component throughout the application. A CSS property is stored with a value in the `Skin` object and is available when the component is being rendered. For example, in `af|breadCrumbs{-tr-show-last-item: false}`, the skin property `-tr-show-last-item` is set to hide the last item in the `breadCrumbs` component's navigation path.

The ADF skinning framework also provides the + and - operators that allow you to set a selector's color or font properties relative to the value that you specify for the color or font properties of another selector. This is useful if you want to apply a range of colors to selectors or maintain a relative size between fonts.

Example 2–1 demonstrates the syntax that you write to make use of this feature for a color property. A global selector alias defines the background color that two additional

global selector aliases (`.fooColorTestPlus` and `.fooColorTestMinus`) apply using the + and – operators.

***Example 2–1 Using the + and - Operators to Apply Color***

```
.BaseBackgroundColor:alias { background-color: #0099ff; }
.fooColorTestPlus {
    -tr-rule-ref: selector(".BaseBackgroundColor:alias");
    background-color: +#333333;
    }
.fooColorTestMinus {
    -tr-rule-ref: selector(".BaseBackgroundColor:alias");
    background-color: -#333333;
}
```

Example 2–2 demonstrates the syntax that you write to make use of this feature for a font property. A global selector alias defines the font size and an additional global selector alias (`.fooFontTest`) increases this font size by `1pt` using the + operator.

***Example 2–2 Using the + Operator to Increase Font Size***

```
.FontSizeTest:alias {font-size: 12pt;}
.fooFontTest {
    -tr-rule-ref: selector(".FontSizeTest:alias");
    font-size:+1pt;
}
```

## 2.4 Rules in the ADF Skinning Framework

The ADF skinning framework defines a number CSS at-rules that allow you to define properties for selectors that you do not want to apply to all browsers, platforms, locales, or reading directions.

For example, you may need to add some padding in Internet Explorer that you do not need on any other browser or maybe you want a font style to be different on Windows than it is on other platforms. To style a selector for a particular user environment, put that skinning information inside an ADF skinning framework rule. The ADF skinning framework picks the styles based on the `HTTP` request information, such as agent and platform, and merges them with the styles without rules. Those CSS properties that match the rules get merged with those outside of any rules. The most specific rules that match a user's environment take precedence.

> **Note:** The visual editor does not currently support the creation of the following rules in an ADF skin. Use the source editor to create and modify the supported rules.

The ADF skinning framework currently supports these rules:

- Define platform styles using `@platform`

  The supported values to set a platform-specific style are `windows`, `macos`, `linux`, `solaris`, and `ppc`.

- Define browser styles using `@agent`

  The supported values to set a browser agent-specific style are `ie`, `mozilla`, `gecko`, `webkit` (maps to safari), `ice`, and `email`.

- Specify styles for any version of Internet Explorer:

  ```
  @agent ie
  ```

- Optionally, specify a specific version of the agent using the `and` keyword. For example, to specify version 7 of Internet Explorer:

  ```
  @agent ie and (version: 7)
  ```

- Use comma-separated rules to specify styles for a number of agents. For example, use the following rule to specify styles for Internet Explorer 6.x, Internet Explorer 7.x, or Gecko 1.9:

  ```
  @agent ie and (version: 6), ie and (version: 7), gecko and
  (version: 1.9)
  ```

- Note that the following two syntax examples specify the same rule:

  ```
  @agent ie and (version: 7.*)
  ```

  ```
  @agent ie and (version: 7)
  ```

  To specify a rule for styles to apply only to Internet Explorer 7.0.x, write the following:

  ```
  @agent ie and (version: 7.0.x)
  ```

- Use the `max-version` and `min-version` keywords to specify a range of versions. For example, you can rewrite the following rule:

  ```
  @agent ie and (version: 6), ie and (version: 7)
  ```

  as:

  ```
  @agent ie and (min-version: 6) and (max-version: 7)
  ```

  to apply styles that you define to all versions of Internet Explorer 6 and 7.

The following example demonstrates how you set the content area of the `inputText` component to the color pink for versions 7 and 8 of Internet Explorer, and version 1.9 of `gecko` on the Windows and Linux platforms.

```
@platform window, linux {
  @agent ie and (version: 7) and (version: 8), gecko and (version: 1.9) {
   af|inputText::content {background-color:pink
   }
  }
}
```

- `@import`

  Specify another ADF skin that defines styles which you can import into your ADF skin. The following example demonstrates valid syntax to import an ADF skin (`skinA`) into the current ADF skin:

  ```
  @import "skinA.css";
  @import url("skinA.css");
  ```

  The `@import` statement(s) must be the first entry in your ADF skin file, as shown in the following example that imports two ADF skins into the current ADF skin:

  ```
  @import url("skinA.css");
  @import url("skinB.css");

  /**ADFFaces_Skin_File / DO NOT REMOVE**/
  @namespace af "http://xmlns.oracle.com/adf/faces/rich";
  @namespace dvt "http://xmlns.oracle.com/dss/adf/faces";
  ```

```
af|inputText{
    background-color: Green;
}
...
```

- @accessibility-profile

  Define `@accessibility-profile`, which defines styles for *high-contrast* and *large-fonts* accessibility profile settings when enabled in the `trinidad-config.xml` file.

  The *high-contrast* value is for cases where background and foreground colors need to contrast highly with each other. The *large-fonts* value is for cases where the user must be allowed to increase or decrease the text scaling setting in the web browser. Defining *large-fonts* does not mean that the fonts are large, but rather that they are scalable fonts or dimensions instead of fixed pixel sizes.

  ```
  <!-- Enable both high-contrast and large-fonts content -->
    <accessibility-profile>high-contrast large-fonts</accessibility-profile>
  ```

  For more information about the `@accessibility-profile` rule, see Section 5.7, "Configuring an ADF Skin for Accessibility."

- `@locale`

  A certain locale must be specified, either only the language or both the language and the country. This defines styles only for a particular language and country. Example 2–3 demonstrates how you can set the color of text rendered by the `af:commandMenuItem` component on a menu bar when the locale is German (`de`).

  **Note:** The ADF skinning framework does not support the `:lang` pseudo class.

Example 2–3 shows several selectors in the CSS file that will be merged together to provide the final style.

***Example 2–3 Merging of Style Selectors***

```
/** For IE and Gecko on Windows, Linux and Solaris, make the color pink. **/
@platform windows, linux, solaris
        {
          @agent ie, gecko
          {
            af|inputText::content {background-color:pink}
          }
        }

af|someComponent {color: red; width: 10px; padding: 4px}

/* For IE, we need to increase the width, so we override the width.
 We still want the color and padding; this gets merged in. We want to add
 height in IE.  */

@agent ie
        {
          af|someComponent {width: 25px; height: 10px}
        }
```

```
/* For IE 7 and 8, we also need some margins.*/
@agent ie (version: 7) and (version: 8)
        {
          af|someComponent {margin: 5px;}
        }

/* For Firefox 3 (Gecko 1.9) use a smaller margin.*/
@agent gecko (version: 1.9)\
        {
          af|someComponent {margin: 4px;}
        }

/* The following selectors are for all platforms and all browsers. */
/* rounded corners on the top-start and top-end */
/* shows how to use :rtl mode pseudo-class. The start image in ltr mode is the */
 /* same as the end image in the right-to-left mode. */
af|panelBox::medium af|panelBox::top-start,
af|panelBox::medium af|panelBox::top-end:rtl {
          background-image: url(/skins/purple/images/panelBoxStart.png);
          width:8px;
          height:8px
        }
/* The following example sets the text color to red when the locale is German
(de)*/
@locale de {
    af|commandMenuItem::bar-item-text
      {
        color: Red;
    }
}
af|panelBox::medium af|panelBox::top-end,
af|panelBox::medium af|panelBox::top-start:rtl {
          background-image: url(/skins/purple/images/panelBoxEnd.png);
          height: 8px;
          width:  8px;
        }
```

# 3

# Working with the Oracle ADF Skin Editor

This chapter describes the visual editor for creating ADF skins. Key features of this editor such as the Selector Tree that you use to browse the items that you can configure in an ADF skin, the Property Inspector that you use to set properties, and how you navigate to an ADF skin that you extend are also described.

This chapter includes the following sections:



## 3.1 Overview of the ADF Skin Editor

The ADF Skin Editor provides a range of features that facilitate the creation and modification of ADF skins. The following list, for which each item has a corresponding label number in Figure 3–2, describes the individual features that the editor exposes when you create an ADF skin, as described Section 4.4, "Creating an ADF Skin File":

1. The Projects node in the Application Navigator contains a node named **skins** where the source files for the ADF skins that you create are stored. The **skins** node is not created until after you create the first ADF skin, as described in Chapter 4, "Creating the Source Files for an ADF Skin."
2. The Structure window lists the selectors, global selector aliases, and style classes that you add to the ADF skin file.

   For more information, see Chapter 2, "Working with ADF Skin Selectors."
3. Click the **Hide/Show Divider** icon to hide or show the Selector Tree.
4. Filter the selectors that appear in the Selector Tree to display all selectors (**Available Selectors**) or only those that you modified in the ADF skin (**Updated Selectors**).

5. The **Extended Skins** list displays the list of ADF skins from which the current ADF skin extends.

   For more information, see Section 3.4, "Navigating to the ADF Skins That Your ADF Skin Extends."

6. Use the **Add** icon to create a new style class or alias selector.

   For more information about creating a new style class, see Chapter 9, "Working with Style Classes." For information about creating an alias selector, see Chapter 8, "Working With Global Selector Aliases."

7. Use the **Delete** icon to remove a selector that you added to the ADF skin.

8. Click the **Refresh** icon to update the Preview Pane after you make changes to the properties of a selector in the Property Inspector.

9. Click the **Add Pseudo-Class** icon to apply a pseudo-class to the item that you selected in the Selector Tree.

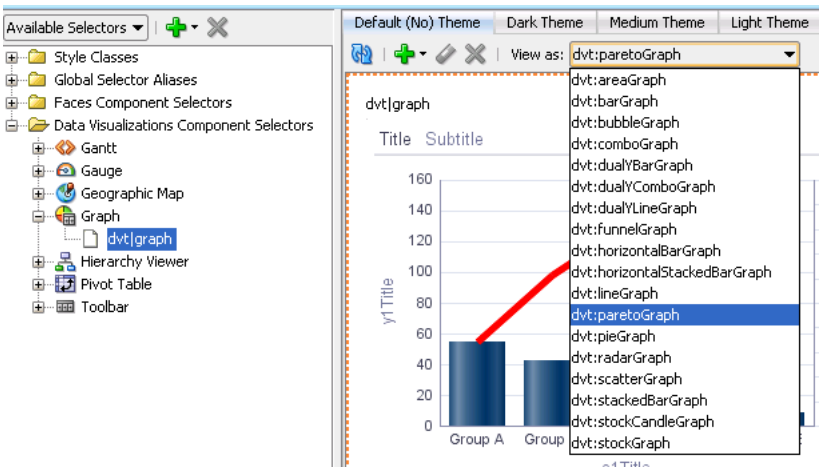
   For more information about pseudo-classes, see Section 2.2, "Pseudo-Classes in the ADF Skinning Framework."

10. Click the **Clear Property Settings** icon to undo any change that you made to the item selected in the Selector Tree.

11. Click the **Delete Pseudo-Class from Skin File** icon to delete any pseudo-classes that you specified in the ADF Skin.

12. The **View as** list allows you to preview how changes you make to a global selector alias in the Selector Tree affect the components that reference the global selector alias. The **View as** list displays all components that reference the global selector alias. The **View as** list also allows you to preview how changes you make to the properties of one component-specific selector impact all sub-types of that component. For example, Figure 3–1 shows the ADF Data Visualization component selector for the `graph` component (`af|dvt-graph`) that exposes a single set of component-specific selectors that apply changes to all graph types. Use the **View as** list to preview a change that you make to a selector in one of the other types of graph (for example, Bar, Funnel, Pareto, and so on).

***Figure 3–1 View as List for a Component***




For more information about global selector aliases, Chapter 8, "Working With Global Selector Aliases."

13. The Selector Tree displays the list of selectors, global selector aliases, and style classes that you can configure values for in an ADF skin.

    For more information, see Section 3.2, "Working with the Selector Tree.".

14. The Preview Pane renders a preview of the changes that you make to a selector in an ADF skin after you click the **Refresh** icon (**8**).

15. You can also view the source of an ADF skin file.

    **Tip:** Select **Split Document** from a context menu that you can invoke from the Preview Pane to render the source and design views of an ADF skin side by side.

16. The Property Inspector identifies properties that you can configure for the ADF skin.

    For more information, see Section 3.3, "Working with the Property Inspector."

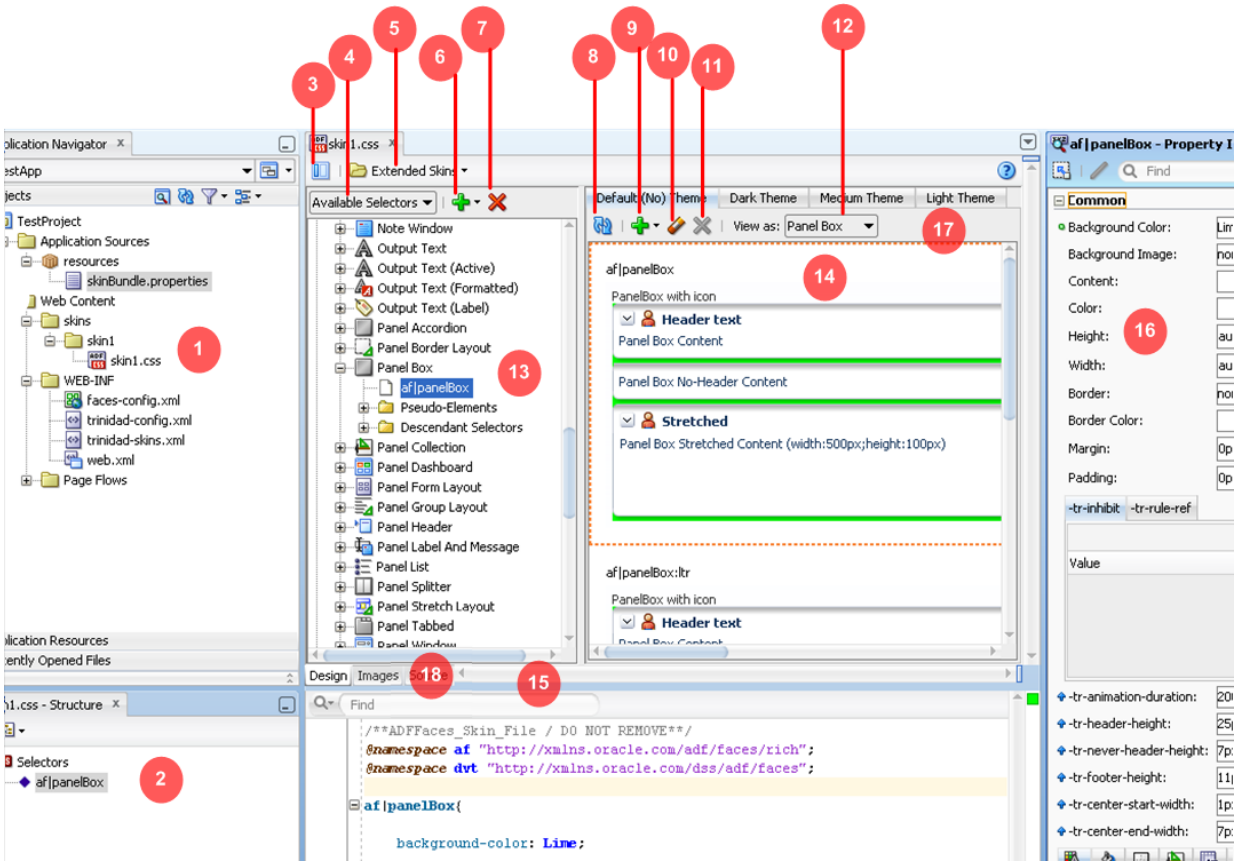
17. The tabs for themes allow you to preview changes that you make for supported themes.

    For more information, see Section 5.6, "Applying Themes to ADF Faces Components."

18. The Images window helps you manage the images that you want to use with an ADF skin.

    For more information, see Section 6.3, "Working with the Images Window."

***Figure 3–2 Overview of the ADF Skin Editor***



## 3.2 Working with the Selector Tree

The Selector Tree displays a list of the style classes, global selector aliases, and selectors for which you can configure properties to change the appearance of ADF Faces and ADF Data Visualization components.

Figure 3–3 shows the nodes that the Selector Tree exposes:

- Style Classes

  A style class defines one or more style properties that you can apply to specific instances of a component. The Selector Tree categorizes the inherited style classes into style classes defined for general usage, note windows, and popups. For more information, see Chapter 9, "Working with Style Classes."

- Global Selector Aliases

  A global selector alias defines style properties that you apply to one or more selectors. The Selector Tree categorizes the inherited global selector aliases into selector aliases defined for general usage, icons, and messages. For more information, see Chapter 8, "Working With Global Selector Aliases."

- Grouped Selectors

  Identifies style properties grouped into one declaration to apply to more than one selector. For example, Figure 3–3 shows a grouped selector for the `commandImageLink` and `goImageLink` component's selectors.

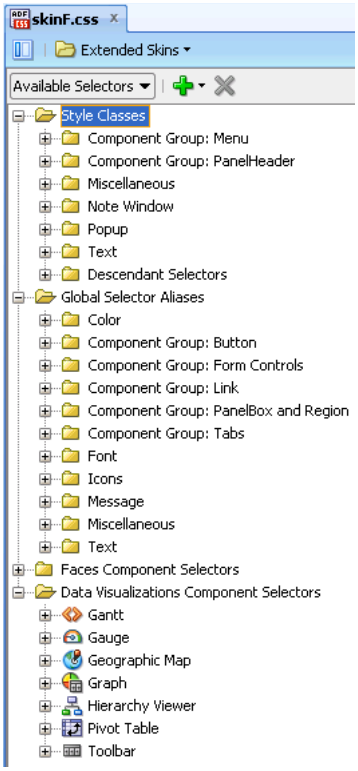
- Faces Component Selector

  Selectors identify the ADF Faces components for which you can configure properties. The Selector Tree displays subcategories for pseudo-elements, component selector aliases, and descendant selectors. For brevity, the ADF Faces components node is not expanded. For more information, see Chapter 5, "Working with Component-Specific Selectors."

- Data Visualizations Component Selectors

  Selectors identify the ADF Data Visualization components for which you can configure properties. The Selector Tree displays subcategories for pseudo-elements, component selector aliases, and descendant selectors. For more information, see Chapter 5, "Working with Component-Specific Selectors."

***Figure 3–3 Selector Tree***




## 3.3 Working with the Property Inspector

The Property Inspector serves a number of functions apart from its primary role of allowing you to set values for CSS properties and ADF skin properties for the selectors that the ADF skinning framework exposes. These functions are the ability to:

- Copy an image into the project where you develop the ADF skin.

  For more information, see Chapter 6, "Working with Images in Your ADF Skin."

- Identify the properties that inherit their values from an extended ADF skin and identify the properties that you configured in the ADF skin, as shown in Figure 3–4.

- Present ADF skin properties that you can configure for a selector.

  For more information, see Section 2.3, "Properties in the ADF Skinning Framework."

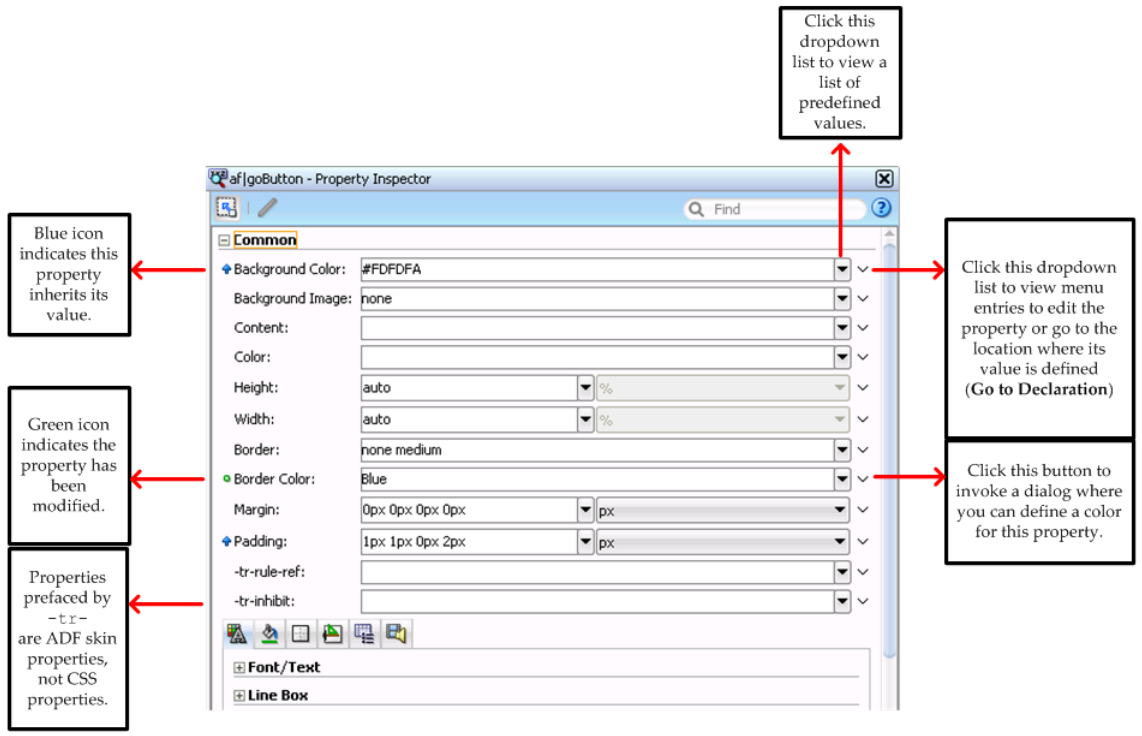
- Navigate to the selector in an extended ADF skin that defines an inherited property in your ADF skin (**Go to Declaration**).

  For more information, see Section 3.4, "Navigating to the ADF Skins That Your ADF Skin Extends."

- Invoke a dialog where you can define the colors for properties that support color value.

Figure 3–4 presents an overview of the various controls that the Property Inspector exposes when you edit an ADF skin.

***Figure 3–4 Property Inspector Controls for ADF Skins***



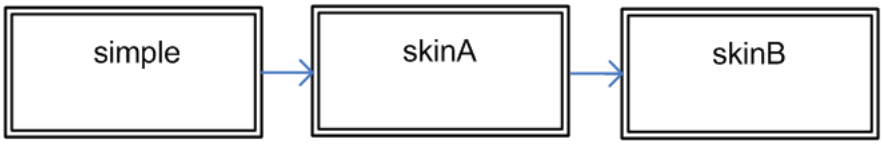

## 3.4 Navigating to the ADF Skins That Your ADF Skin Extends

When you create an ADF skin, as described in Section 4.4, "Creating an ADF Skin File," you choose an ADF skin from which to extend. The ADF skin that you choose to extend from defines properties that your newly created ADF skin inherits. When you create your first ADF skin, you must choose one of the ADF skins that Oracle ADF provides.

Subsequent ADF skins that you create can extend an ADF skin that you created or one of the ADF skins provided by Oracle ADF. For example, you create your first ADF skin named `skinA` that extends the `simple` ADF skin provided by Oracle ADF. You then create a second ADF skin named `skinB`. When creating `skinB`, you have the choice of extending from `skinA` or from any of the ADF skins provided by Oracle ADF. If you choose to extend `skinB` from `skinA`, the inheritance relationship between the ADF skins is as illustrated in Figure 3–5.

For more information about the ADF skins that Oracle ADF provides, see Section 1.5, "Inheritance Relationship of the ADF Skins Provided by Oracle ADF," and Section 11.4, "ADF Skins Provided by Oracle ADF."

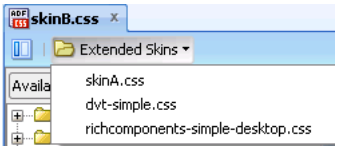
***Figure 3–5 Example Inheritance Relationship Between ADF Skins***




The **Extended Skins** list in the visual editor displays the list of ADF skins that the current ADF skin extends. Figure 3–6 shows the list of ADF skins that appears if you

implement the inheritance relationship described in Figure 3–5. You open an extended ADF skin that you want to view by clicking it in the **Extended Skins** list.

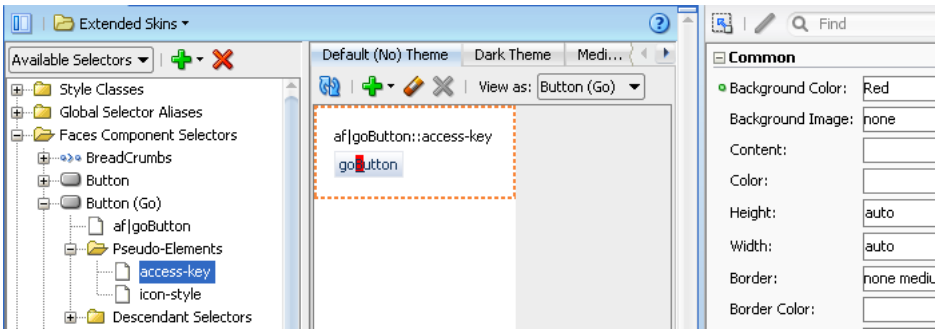
***Figure 3–6 Extended Skins List***




> **Note:** You cannot edit the properties of the selectors in the ADF skins provided by Oracle ADF. You can only edit the properties of selectors in extended ADF skins that you created.

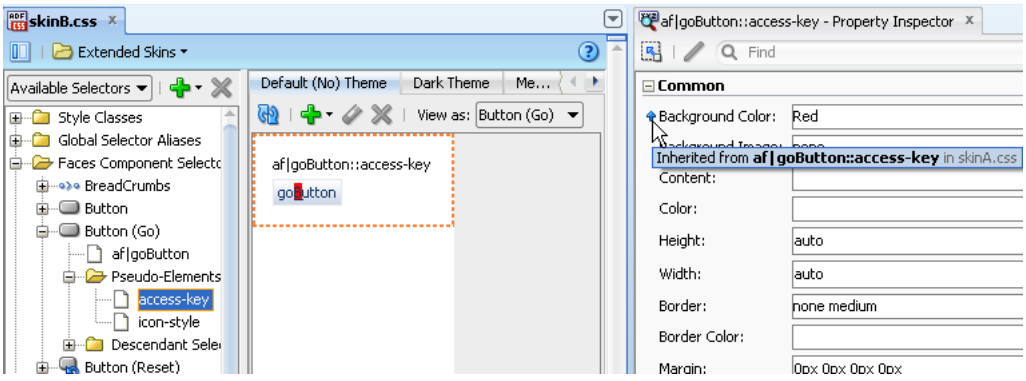
Using the **Go to Declaration** menu that the Property Inspector exposes, you can navigate to the location in an extended ADF skin where the extended ADF skin declares style properties inherited by other ADF skins. For example, assume that the `skinA` ADF skin defines a background color of `Red` for the `af|goButton` selector's `access-key` pseudo-element, as shown in Figure 3.4.

***Figure 3–7 Declaration of a Property Value***




The `skinB` ADF skin that extends from `skinA` ADF skin inherits the property values, as shown in Figure 3.4.

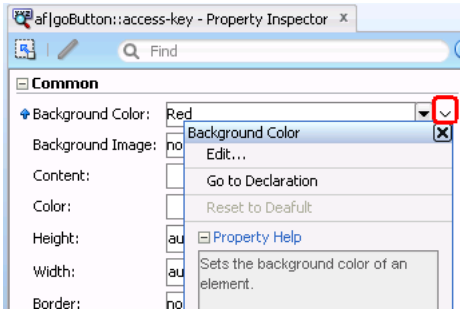
***Figure 3–8 Inheriting a Property Value from an Extended Skin***



**To go to the declaration of a property:**

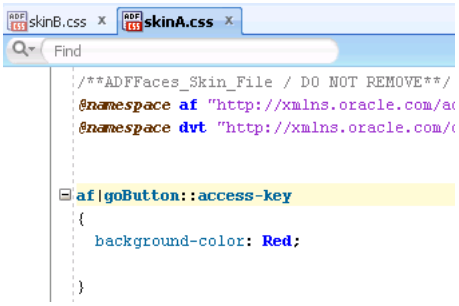
1. Identify a property in your ADF skin that inherits its values from an extended ADF skin. A blue upward-pointing arrow, as shown in Figure 3–8, identifies these properties.
2. Click the list beside this property to invoke a context menu where you select **Go to Declaration**, as shown in Figure 3–9.

***Figure 3–9 Go to Declaration Context Menu***




The extended ADF skin opens in the source view, as shown in Figure 3–10. If the extended ADF skin is one that you created, you can modify the property values defined in it. The ADF skins provided by Oracle ADF, described in Section 11.4, "ADF Skins Provided by Oracle ADF," are read-only.

***Figure 3–10 Property Value Defined in Extended ADF Skin***




## 3.5 Customizing the ADF Skin Editor

You can alter the appearance and functionality of a variety of ADF Skin Editor features.

### 3.5.1 How to Change the Look and Feel of the ADF Skin Editor

You can alter the appearance of the ADF Skin Editor using pre-defined settings.

**To change the look and feel of the ADF Skin Editor:**

1. From the main menu, choose **Tools** > **Preferences**. For more information at any time, press F1 or click **Help** from within the Preferences dialog.
2. In the Preferences dialog, select the **Environment** node if it is not already selected.
3. On the Environment page, select a different look and feel from the **Look and Feel** list.

4. Click **OK**.
5. Restart the ADF Skin Editor.

> **Note:** The key bindings in Motif are different from key bindings in Windows. Under Motif, the arrow keys do not change the selection. Instead they change the lead focus cell. You must press Ctrl + Space to select an item. This is expected behavior.

## 3.5.2 How to Customize the General Environment for the ADF Skin Editor

You can customize the default display options (such as whether dockable windows are always on top), as well as other general behavior, such as whether the ADF Skin Editor will automatically reload externally modified files and whether output to the Log window is automatically saved to a file.

**To change the general environment settings for the ADF Skin Editor:**

1. From the main menu, choose **Tools** > **Preferences**. For more information at any time, press F1 or click **Help** from within the Preferences dialog.
2. In the Preferences dialog, select the **Environment** node if it is not already selected.
3. On the Environment page, select the options and set the fields as appropriate.
4. Click **OK**.
5. Restart the ADF Skin Editor.

   For information about how to start the ADF Skin Editor, see the *Oracle Fusion Middleware Installation Guide for Oracle Application Development Framework Skin Editor*.

## 3.5.3 How to Customize Dockable Windows in the ADF Skin Editor

You can customize the layout for dockable windows in their docked position. You can also set dockable windows to remain on top of other GUI elements, or not, when those windows are moved.

**To change the shape of one or more of the four docking areas:**

1. From the main menu, choose **Tools** > **Preferences**. For more information at any time, press F1 or click **Help** from within the Preferences dialog.
2. In the Preferences dialog, select the **Environment** node select **Dockable Windows**.
3. On the Dockable Windows page, click the corner arrows to lengthen or shorten each docking area's shape.
4. Click **OK**.

**To change whether dockable windows remain on top or not when moved:**

1. From the main menu, choose **Tools** > **Preferences**. For more information at any time, press F1 or click **Help** from within the Preferences dialog.
2. In the Preferences dialog, select the **Environment** node select **Dockable Windows**.
3. On the Dockable Windows page, select or deselect **Dockable Windows Always on Top** as appropriate.
4. Click **OK**.

# 3.6 Searching the Source Files of ADF Skins

The ADF Skin Editor provides a source editor where you can view, edit, and search the syntax that the visual editor generates for an ADF skin.

## 3.6.1 How to Search the Source Files of ADF Skins

You can search the source files of an ADF skin in a number of ways.

**To search a source file currently open in the source editor, with the option to replace text:**

1. With the file open in the source editor, make sure that the editor has focus.
2. Optionally, if an instance of the text you want to search for is easily found, you can highlight it now.
3. From the main menu, choose **Search > Find**. Alternatively, press Ctrl+F.
4. In the Find Text Dialog, enter or select the text to locate.

   Text previously searched for in this session of the ADF Skin Editor appears in the **Text to Search For** list.
5. Select other search parameters accordingly.

   For more information, press F1 or click **Help** from within the dialog.
6. Click **OK**.

**To do a simple search in the open source file for a single text string:**

1. With the file open in the editor, ensure that the editor has focus.
2. Place the cursor in the file at the point you wish to search from.
3. From the main menu, choose **Search > Incremental Find Forward** or **Search > Incremental Find Backwards**.
4. In the dialog, enter the search text.

   As you type, the cursor jumps to the next instance of the group of letters displayed.

Alternatively, enter the text string in the search box. As you type, the cursor jumps to the next instance of the group of letters displayed. Use the **Previous** or **Next** buttons to search up and down the file. Click in the search box to set **Match Case**, **Whole Word**, or **Highlight Occurrences**.

# 3.7 Working with Extensions

Extensions are components that are loaded and integrated with the ADF Skin Editor after it is started. Extensions can access the editor and perform many useful tasks. You can add existing extensions into the ADF Skin Editor, or create your own.

This section contains information on finding and installing extensions. The simplest way to find and download extensions is through the Check for Updates wizard.

If you need additional capabilities (such as integration with a version control system or a special editor), you can add external tools to the ADF Skin Editor. See Section 3.9, "Adding External Tools to the ADF Skin Editor" for more information.

In addition, you can obtain additional extension development tools and functionality in the Extension Software Development Kit (SDK). You can download the Extension SDK via the Check for Updates wizard.

You can also download the Extension SDK from the Oracle Technology Network Web page.

> **Note:** Any time an extension is added or upgraded, the migration dialog appears at startup in case you need to migrate any previous settings related to that extension.

### 3.7.1 How to Install Extensions with Check for Updates

The easiest way to find and install extensions is to use the Check for Updates wizard.

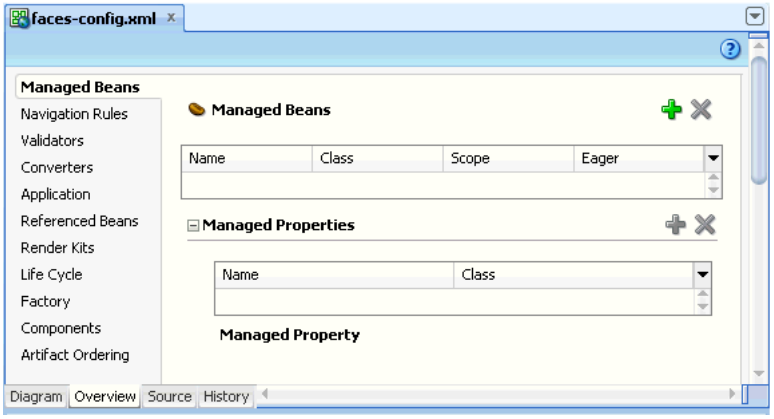
**To install extensions using the Check for Updates wizard:**

1. From the **Help** menu, select **Check for Updates**.
2. Follow the steps in the wizard to browse, download, and install patches and extensions.

## 3.8 Working with the Overview Editor in the ADF Skin Editor

Although the ADF Skin Editor creates the `faces-config.xml` file by default when you create a new ADF skin project, this file is not required for ADF skin projects. In the unlikely event that you need to edit the `faces-config.xml` file, you can use the overview editor for JSF configuration files. Figure 3–11 displays the overview editor.

***Figure 3–11 Overview Editor for JSF Configuration File***




When you open `faces-config.xml` its contents are displayed in an editor group. When you select the **Overview** tab at the bottom of this group, the overview editor appears.

When the overview editor is open, the Property Inspector displays the metadata child elements for the currently selected element. Use the Property Inspector to manage these. For instance, you use the Property Inspector to set the `<description>` and `<display-name>` child elements.

The overview editor has three sections:

- The left-hand column displays the main JSF configuration elements.

- The top area of the main panel shows child elements for the element selected in the element list on the left.
- The bottom area of the main panel shows child elements for the element selected at the top area.

### 3.8.1 How to Use the Overview Editor for JSF Configuration Files in the ADF Skin Editor

You can add, delete, or edit your JSF element and child elements using the Overview Editor.

**To work with a main JSF configuration element and its immediate child elements:**

1. In Application Navigator, open the workspace that contains your JSF application.
2. In the workspace, open the project that contains your JSF pages.
3. In the project, open the WEB-INF node.
4. Under the WEB-INF node, double-click the `faces-config.xml` file to open.
5. At the bottom of the editor, select the Overview tab.
6. Select an element from the element list on the left. The main panel displays corresponding configurable child elements in a table at the top of the main panel.

**To add, delete, or edit JSF configuration elements:**

- **To add a new child element**. Click **New.** A dialog box opens to create the element. If no new button displays, the child element must be an existing class. You can select the class by clicking **Browse...** . If no browse button appears, or if the entry is not a class name, you can enter a value directly.
- **To delete an existing child element**. Select the element from the table and click **Delete**. The element is removed from the table. If no delete button displays, the entry can be deleted manually.
- **To edit an existing child element**. Select the element from the table and click **Edit**. The Properties panel for the element opens to change the value.

**To view, add, delete, or edit child configuration element child element:**

- **To view child elements.** Select an element from the element list on the left. The main panel displays. Select an existing child element from a table at the top of the main panel. Allowed child elements display in a table at the bottom of the main panel. If a child element allows child elements, but no children are currently defined, the list area for those children might be hidden. To display the list area and add children, click the show arrow to the left of the area title. To hide the list area, click the hide arrow.
- **To add a new child element.** Click **New.** If no new button displays and the child element must be an existing class, you can select the class by clicking **Browse...** to open the **Class Editor** dialog box. If no browse button appears, or if the entry is not a class name, you can enter a value directly.
- **To edit an existing child element.** Select it from the table and click **Edit.** The Properties panel for the element opens to change the value. If no edit button displays, you can either select a new class (if applicable), or edit the entry To delete an existing child element, select it from the table and click **Delete**.

- T**o delete an existing child element**. Select it from the table and click **Delete**. The element is removed from the table. If no delete button displays, you can delete the entry manually.

## 3.9 Adding External Tools to the ADF Skin Editor

External tools are custom ADF Skin Editor menu items and toolbar buttons that launch applications installed on your system, applications that are not packaged as part of the ADF Skin Editor.

### 3.9.1 How to Add External Tools to the ADF Skin Editor

You find and add available external tools to the ADF Skin Editor using the **External Tools** menu.

**To find all external programs that the ADF Skin Editor is preconfigured to support:**

1. From the main menu, choose **Tools > External Tools**.
2. In the External Tools dialog, click **Find Tools**.

**To add access to an external program from the ADF Skin Editor:**

1. From the main menu, choose **Tools > External Tools**.
2. In the External Tools dialog, click **New**. Follow the instructions in the wizard.

**To change how an external program appears, or remove access to an external program from the ADF Skin Editor:**

1. From the main menu, choose **Tools > External Tools**.
2. In the External Tools dialog, click **Edit** or **Delete**. If you are editing the options, display, integration or availability of an external tool from the ADF Skin Editor, select the corresponding tab and change the values. Click **Help** for help choosing valid values.
3. Click **OK**. Your changes are reflected immediately.

## 3.10 Navigating the ADF Skin Editor

You can accomplish any task in the ADF Skin Editor using the keyboard as you use the mouse.

### 3.10.1 How to Work With Shortcut Keys In the ADF Skin Editor

The ADF Skin Editor comes with several predefined keyboard schemes. You can choose to use one of these, or customize an existing set to suit your own coding style by changing which keyboard shortcuts map to which actions.

**To load preset keyboard schemes:**

1. From the main menu, choose **Tools** > **Preferences**.
2. In the Preferences dialog, select the **Shortcut Keys** node. For more information at any time, press F1 or click **Help** from within the Preferences dialog.

3. On the shortcut keys page, click **More Actions** and then select **Load Keyboard Scheme**. The Load Keyboard Scheme dialog appears, with the currently loaded keyboard scheme highlighted.
4. In the Load Keyboard Scheme dialog, select the scheme you wish to load and click **Ok**.
5. On the Shortcut Keys page, if you have finished, click **Ok**.

**To view the ADF Skin Editor commands and their associated keyboard shortcuts (if assigned):**

1. From the main menu, choose **Tools** > **Preferences**.
2. In the Preferences dialog, select the **Shortcut Keys** node.
3. On the Shortcut Keys page, under **Available Commands**, you can view the complete set of the ADF Skin Editor commands, and what keyboards shortcuts (if any) are assigned to each. If you are looking for a particular command or shortcut, or want to look at shortcuts for a particular category of commands only, enter a filtering expression in the **Search** field.
4. You can also define new shortcuts, or change existing ones.

**To define a new keyboard shortcut for a command within a given keyboard scheme:**

1. From the main menu, choose **Tools** > **Preferences**.
2. In the Preferences dialog, select the **Shortcut Keys** node. For more information at any time, press F1 or click **Help** from within the preferences dialog.
3. On the Shortcut Keys page, under **Available Commands**, select the command that you wish to define a new shortcut for.
4. To define a new shortcut for this action, place focus on the **New Shortcut** field, and then press the key combination on the keyboard.

   If this proposed shortcut already has a command associated with it, that command will now appear in the **Conflicts** field. Any conflicting shortcuts are overwritten when a new shortcut is assigned.
5. To assign this shortcut to the action selected, click **Assign**. If you want to delete an already-assigned shortcut, click the **Delete** button in the toolbar.

   If you want to assign more than one shortcut to a command, select the command and click the **Duplicate** button. Then, type the shortcut key in the **New Shortcut** field and click **Assign**.
6. When you are finished, click **Ok**.

**To import or export keyboard schemes:**

1. From the main menu, select **Tools > Preferences** to open the Preferences dialog.
2. Click **More Actions > Export** or **Import**. Keyboard schemes are stored as XML files.

## 3.10.2 Keyboard Navigation In the ADF Skin Editor

For any action that can be accomplished with a mouse, including selection, there is a way to accomplish the action solely from the keyboard. You can accomplish any task in the ADF Skin Editor using the keyboard as you can using the mouse.

The shortcut keys defined in the Java Look and Feel guidelines provide the base set for the ADF Skin Editor. The various predefined keyboard schemes available in the ADF Skin Editor are then overlaid upon this base set. If the same shortcut key exists in both the look and feel guidelines and the ADF Skin Editor keyboard scheme, the ADF Skin Editor scheme prevails. If a shortcut key defined by the look and feel guidelines does not appear in the ADF Skin Editor scheme, then it is the original look and feel definition that remains in effect when the scheme in question is enabled.

At any given time, then, the shortcut keys enabled in the ADF Skin Editor depend upon the interaction of the currently enabled scheme with the Java look and feel guidelines. When you first open the ADF Skin Editor, the default scheme is enabled. You can change this scheme whenever you wish, and within each scheme, you can customize any of the shortcut key assignments that you would like. Note that any customized shortcuts you create in a scheme are not retained when another predefined keyboard scheme is activated (or even if the same scheme is reloaded).

To load predefined keyboard schemes, view current shortcut assignments within a scheme, and customize those assignments, you will need to open the preferences dialog. To open the dialog, choose **Tools** > **Preferences** (or on the keyboard, press Alt+T+P) from the main menu and then, using the arrow keys in the left-hand pane, navigate to the **Shortcut Keys** node. For details on working with the dialog, with the page displayed, click **Help** (or on the keyboard press H).

#### 3.10.2.1 Common Navigation Keys

The following table describes the common methods of moving the cursor in the ADF Skin Editor:

***Table 3–1 Common Methods of Moving the Cursor***

| Key | Cursor Movement | Ctrl+cursor Movement |
|---|---|---|
| Left Arrow | Left one unit (e.g., a single character) | Left one proportionally larger unit (e.g., a whole word) |
| Right Arrow | Right one unit | Right one proportionally larger unit |
| Up Arrow | Up one unit or line | Up one proportionally larger unit |
| Down Arrow | Down one unit or line | Down one proportionally larger unit |
| Home | Beginning of the line | To the beginning of the data (top-most position) |
| End | End of the line | To the end of the data (bottom-most position) |

*Table 3–1 (Cont.) Common Methods of Moving the Cursor*

| Key | Cursor Movement | Ctrl+cursor Movement |
|---|---|---|
| Tab | Next field or control, except when in a text area or field. In this case, press Ctrl+Tab to navigate out of the control.<br>Where there are fields and controls ordered horizontally as well as vertically, pressing Tab moves the cursor first horizontally to the right, then at the end of the line, down to the left of the next line. | To the next pane which may be a navigator, an editor, or a palette, except when in a text area or field. In this case, press Ctrl+Tab to navigate out of the control |
| Shift+Tab | Previous field | To previous tab position. In property sheets, this moves the cursor to the next page |
| Enter | Selects and highlights the default button, except when in a combo box, shuttle button, or similar control.<br>**Note**: The default button changes as you navigate through controls. | n/a |

### 3.10.2.2 Navigation In Standard Components

This section describes keyboard navigation in the standard ADF Skin Editor components.

#### Buttons

The following table describes the keyboard actions to perform navigation tasks involving buttons.

*Table 3–2 Keyboard Navigation for Buttons*

| Navigation | Keys |
|---|---|
| Navigate forward to or from button | Tab |
| Navigate backward to or from button | Shift+Tab |
| Activate the default button (when the focus is not on a button) | Enter |
| Activate any button while it has focus | Enter, Spacebar, or keyboard shortcut (if one has been defined) |
| Activate **Cancel** or **Close** buttons on a dialog | Esc |

#### Checkboxes

The following table describes the keyboard actions to perform navigation tasks involving checkboxes.

*Table 3–3 Keyboard Navigation for Checkboxes*

| Navigation | Keys |
|---|---|
| Navigate forward to or from checkbox | Tab |
| Navigate backward to or from checkbox | Shift+Tab |

*Table 3–3 (Cont.) Keyboard Navigation for Checkboxes*

| Navigation | Keys |
|---|---|
| Select or deselect (when the focus is on the checkbox) | Spacebar or keyboard shortcut (if one has been defined) |
| Navigate to checkbox and select or deselect (when the focus is not on the checkbox) | Keyboard shortcut (if one has been defined) |

### Dropdown Lists And Combo Boxes

The following table describes the keyboard actions to perform navigation tasks involving dropdown lists and combo boxes.

*Table 3–4 Keyboard Navigation for Dropdown Lists and Combo Boxes*

| Navigation | Keys |
|---|---|
| Navigate forward to or from a combo box or dropdown list | Tab or keyboard shortcut (if one has been defined) |
| Navigate backward to or from a combo box or dropdown list | Shift+Tab |
| Toggle list open and closed | Spacebar (the current selection receives the focus) |
| Open a list | Down Arrow to open (first item on list receives focus) |
| Move up or down within list | Up and Down Arrow keys (highlighted value has focus) |
| Move right and left within the initial entry on a combo box | Right and Left Arrow keys |
| Select list item | Enter<br>**Note**: The first time you press Enter, the item in the list is selected. The second time you press Enter, the default button is activated. |
| Close list (with the highlighted value selected) | Esc |

### List Boxes

The following table describes the keyboard actions to perform navigation tasks involving list boxes.

*Table 3–5 Keyboard Navigation for List Boxes*

| Navigation | Keys |
|---|---|
| Navigate forward into or out of a list | Tab |
| Navigate backward into or out of list | Shift+Tab |
| Make a selection | Up Arrow, Down Arrow, Spacebar, or Enter<br>**Note**: The first time you press Enter, the highlighted item in the list is selected. The second time you press Enter, the default button is activated. |

*Table 3–5 (Cont.) Keyboard Navigation for List Boxes*

| Navigation | Keys |
|---|---|
| Move within list | Up Arrow or Down Arrow |
| Move to beginning of list | Home or Ctrl+Home |
| Move to end of list | End or Ctrl+End |
| Select all entries | Ctrl+A |
| Toggle (select or deselect) an item | Spacebar or Ctrl+Spacebar |
| Select next item up in list without deselecting item with current focus | Shift+Up Arrow Key |
| Select next item down in list without deselecting item with current focus | Shift+Down Arrow Key |
| Select current item and all items up to the top of the list | Shift+Home |
| Select current item and all items up to the bottom of the list | Shift+End |
| Select current item and all items visible above that item | Shift+Page Up |
| Select current item and all items visible below that item | Shift+Page Down |
| Select item with current focus without deselecting other items (to select items that are not adjacent) | Ctrl+Spacebar |
| Navigate through list without deselecting item with current focus. | Ctrl+Up Arrow or Ctrl+Down Arrow |

### Radio Buttons

*Table 3–6 Keyboard Navigation for Radio Buttons*

| Navigation | Keys |
|---|---|
| Navigate forward to or from radio button | Tab |
| Navigate backward to or from radio button | Shift+Tab |
| Navigate forward from radio button | Arrow Keys |
| Navigate backward from radio button | Shift+Arrow Keys |
| Select radio button | Arrow key (navigating to a radio button via arrows selects it) or keyboard shortcut (if one has been defined) |
| Deselect radio button | Select a different radio button in the group using one of the commands above |

### Shuttles

The following table describes the keyboard actions to perform navigation tasks involving shuttles.

*Table 3–7 Keyboard Navigation for Shuttles*

| Navigation | Keys |
|---|---|
| Navigate forward into or out of a list | Tab |
| Navigate backward into or out of list | Shift+Tab |
| Make a selection | Up Arrow or Down Arrow |
| Move within list | Up Arrow or Down Arrow |
| Move to beginning of list | Home or Ctrl+Home |
| Move to end of list | End or Ctrl+End |
| Select all entries | Ctrl+A |
| Toggle (select or deselect) an item | Spacebar or Ctrl+Spacebar |
| Select next item up in list without deselecting item with current focus | Select next item up in list without deselecting item with current focus |
| Select next item down in list without deselecting item with focus | Shift+Down Arrow Key |
| Select current item and all items up to the top of the list | Shift+Home |
| Select current item and all items up to the bottom of the list | Shift+End |
| Select current item and all items visible above that item | Shift+Page Up |
| Select current item and all items visible below that item | Shift+Page Down |
| Select item with current focus without deselecting other items (to select items that are not adjacent) | Ctrl+Spacebar |
| Navigate through list without deselecting item with current focus. | Ctrl+Up Arrow or Ctrl+Down Arrow |

### Sliders

The following table describes the keyboard actions to perform navigation tasks involving sliders.

*Table 3–8 Keyboard Navigation for Sliders*

| Navigation | Keys |
|---|---|
| Navigate forward to or from slider | Tab |

*Table 3–8 (Cont.) Keyboard Navigation for Sliders*

| Navigation | Keys |
|---|---|
| Navigate backward to or from slider | Shift+Tab |
| Increase value | Up Arrow or Right Arrow |
| Decrease value | Left Arrow or Down Arrow |
| Minimum value | Home |
| Maximum value | End |

### Spin Controls

The following table describes the keyboard actions to perform navigation tasks involving spin controls.

*Table 3–9 Keyboard Navigation for Spin Controls*

| Navigation | Keys |
|---|---|
| Navigate forward to or from spin control | Tab |
| Navigate backward to or from spin control | Shift+Tab |
| Increase value | Up Arrow or Right Arrow, or type the value you want |
| Decrease value | Left Arrow or Down Arrow, or type the value you want |
| Minimum value | Home |
| Maximum value | End |

### Text Fields

The following table describes the keyboard actions to perform navigation tasks involving text fields.

*Table 3–10 Keyboard Navigation for Text Fields*

| Navigation | Keys |
|---|---|
| Navigate forward into or out of text box | Tab or keyboard shortcut (if one has been defined) |
| Navigate backward into or out of text box | Shift+Tab |
| Move to previous/next character within text box | Left Arrow/Right Arrow |
| Move to start/end of box | Home/End |
| Select all text | Ctrl+A |
| Deselect all text | Left Arrow or Right Arrow |
| Select current item and all items up to the Left/Right | Shift+Left Arrow, Shift+Right Arrow |
| Select current item and all items up to the Start/End | Shift+Home, Shift+End |
| Select current item and all items up to the previous/next word | Ctrl+Shift+Left Arrow, Ctrl+Shift+Right Arrow |

*Table 3–10 (Cont.) Keyboard Navigation for Text Fields*

| Navigation | Keys |
|---|---|
| Copy selection | Ctrl+C |
| Cut selection | Ctrl+X |
| Paste from clipboard | Ctrl+V |
| Delete next character | Delete |
| Delete previous character | Backspace |

### 3.10.2.3 Navigating Complex Controls

This section contains information about keyboard shortcuts for complex UI components.

#### Dockable Windows

The following table describes the keyboard actions to perform navigation tasks involving dockable windows.

*Table 3–11 Keyboard Navigation for Dockable Windows*

| Navigation | Keys |
|---|---|
| Navigate forward in or out of dockable window | Ctrl+Tab |
| Navigate backward in or out of dockable window | Ctrl+Shift+Tab |
| Display context menu | Shift+F10 |
| Navigate between tabs within a dockable window | Alt+Page Down, Alt+Page Up |
| Move between elements including dropdown lists, search fields, panels, tree structure (but not individual elements in a tree), individual component buttons | Tab |
| Move up/down through dockable window contents (scrollbar) | Up Arrow, Down Arrow<br>This scrolls the window contents if the focus moves beyond visible area of canvas. |
| Move left/right (scrollbar) | Up Arrow, Down Arrow<br>This scrolls the pane contents if focus moves beyond visible area of canvas. |
| Move to start/end of data (component buttons) | Ctrl+Home, Ctrl+End |
| Select an element | Enter or Spacebar |
| Scroll left/right within the canvas area (without moving through the window contents) | Ctrl+Left/Ctrl+Right |
| Scroll Up/Down within the canvas area (without moving through the window contents) | Ctrl+Up/Ctrl+Down |

### Menus

Context menus are accessed using Shift+F10. Menus from the main menu bar are accessed using the keyboard shortcut for the menu.

The following table describes the keyboard actions to perform navigation tasks involving the menu bar.

***Table 3–12 Keyboard Navigation for Menus***

| Navigation | Keys |
|---|---|
| Navigate to menu bar | F10 |
| Navigate out of menu bar | Esc |
| Navigate between menus in menu bar | Right Arrow, Left Arrow |
| Navigate to menu item | Up Arrow, Down Arrow |
| Navigate from menu item | Up Arrow, Down Arrow |
| Activate item | Enter, Spacebar, or keyboard shortcut (if one has been defined) |
| Open submenu | Right Arrow |
| Retract submenu | Left Arrow or Esc |

### Panels

The following table describes the keyboard actions to perform navigation tasks involving panels.

***Table 3–13 Keyboard Navigation for Panels***

| Navigation | Keys |
|---|---|
| Navigate in/out forward | Tab |
| Navigate in/out backward | Shift+Tab |
| Expand panel (when focus on header) | Right Arrow |
| Collapse panel (when focus on header) | Left Arrow |
| Navigate within panel | Up Arrow, Down Arrow |
| Navigate to panel header from contents (when focus is on top item in list) | Up Arrow |
| Navigate to panel contents from header (when focus is on header) | Down Arrow |

### Tables

Arrow keys move focus in the direction of the arrow, except when a web widget has focus; in that case, the down arrow or enter key initiates the widget control action, such as opening a choice list. tab moves the focus right, shift+tab moves the focus left.

The following table describes the keyboard actions to perform navigation tasks involving tables.

*Table 3–14 Keyboard Navigation for Tables*

| Navigation | Keys |
|---|---|
| Navigate forward in or out of table | Ctrl+Tab |
| Navigate backward in or out of table | Shift+Ctrl+Tab |
| Move to next cell (wrap to next row if in last cell) | Tab Arrow or Right Arrow |
| Move to previous cell (wrap to previous row if in first cell) | Shift+Tab or Left Arrow |
| Controls in cells open | Down Arrow or Enter |
| Block move left | Ctrl+Page Up |
| Block move right | Ctrl+Page Down |
| Block move up | Page Up |
| Block move down | Page Down |
| Move to first cell in row | Home |
| Move to last cell in row | End |
| Move to first cell in table | Ctrl+Home |
| Move to last cell in table | Ctrl+End |
| Select all cells | Ctrl+A |
| Deselect current selection (and select alternative) | Any navigation key |
| Extend selection on row | Shift+Up Arrow |
| Extend selection one column | Shift+Down Arrow |
| Extend selection to beginning of row | Shift+Home |
| Extend selection to end of row | Shift+End |
| Extend selection to beginning of column | Ctrl+Shift+Home |
| Extend selection to end of column | Ctrl+Shift+End |
| Edit cell without overriding current contents, or show dropdown list in combo box | F2 |
| Reset cell content prior to editing | Esc |

### Tabs

This section refers to the tabs that appear within a dockable window, view or dialog. The following table describes the keyboard actions to perform navigation tasks involving tabs in dockable windows, views and dialogs.

*Table 3–15 Keyboard Navigation for Tabs*

| Navigation | Keys |
|---|---|
| Navigate forward into or out of tab control | Tab |
| Navigate backward into or out of tab control | Ctrl+Tab |
| Move to tab (within control) left/right | Left Arrow/Right Arrow |
| Move to tab (within control) above/below | Up Arrow/Down Arrow |
| Move from tab to page | Ctrl+Down |
| Move from page to tab | Ctrl+Up |
| Move from page to previous page (while focus is within page) | Ctrl+Page Up |
| Move from page to next page (while focus is within page) | Ctrl+Page Down |

### Trees

The following table describes the keyboard actions to perform navigation tasks involving trees.

*Table 3–16 Table Navigation for Trees*

| Navigation | Keys |
|---|---|
| Navigate forward into or out of tree control | Tab |
| Navigate backward into or out of tree control | Shift+Tab |
| Expand (if item contains children) | Right Arrow |
| Collapse (if item contains children) | Left Arrow |
| Move to parent from child (if expanded) | Left Arrow |
| Move to child from parent (if already expanded) | Right Arrow |
| Move up/down one item | Up Arrow, Down Arrow |
| Move to first item | Home |
| Move to last entry | End |
| Select all children of selected parent | Ctrl+A |
| Select next item down in list without deselecting that item that currently has focus | Shift+Down Arrow |
| Select next item up in list without deselecting that item that currently has focus | Shift+Up Arrow |

*Table 3–16 (Cont.) Table Navigation for Trees*

| Navigation | Keys |
|---|---|
| Select current item and all items up to the top of the list | Shift+Home |
| Select current item and all items up to the bottom of the list | Shift+End |
| Select the item with current focus without deselecting other items (to select items that are not adjacent) | Ctrl+Spacebar |
| Navigate through list without deselecting item with current focus | Ctrl+Up/Down Arrow |

#### Wizards

The Following Table Describes The Keyboard Actions To Perform Navigation Tasks Involving Wizards.

*Table 3–17 Keyboard Navigation for Wizards*

| Navigation | Keys |
|---|---|
| Navigate between stops on the roadmap or between pages | Up Arrow, Down Arrow (these do not wrap) |
| Navigate forward between components on wizard panel, wizard navigation bar buttons, and navigation panel | Tab |
| Navigate backward between components on wizard panel, wizard navigation bar buttons, and navigation panel | Shift+Tab |
| Navigate between buttons on Navigation Bar | Right and Left Arrow Key (does not wrap) |
| Navigate between stops on Roadmap/between wizard pages | Ctrl Page Up and Ctrl Page Down |

### 3.10.2.4 Navigation in Specific Components

This section contains information about keyboard shortcuts for the ADF Skin Editor-specific UI components.

#### Dialogs

The following table describes the keyboard actions to perform navigation tasks involving dialogs.

*Table 3–18 Keyboard Navigation for Dialogs*

| Navigation | Keys |
|---|---|
| Close dialog without making any selections or changes | Esc |
| Activate the default button (if one is defined) | Enter |

### Overview Editor (Form + Mapping)

The following table describes the keyboard actions to perform navigation tasks involving overview editors.

*Table 3–19 Keyboard Navigation for the Overview Editor*

| Navigation | Keys |
|---|---|
| Navigate into or out of overview editor from other pages in editor (for example Source or History) | Alt+Tab |
| Navigate from the tab group to next control in editor) | Tab or Ctrl+Down Arrow |
| Navigate forward or backwards between controls on overview editor | Tab or Alt+Tab |
| Move between tabs in the side tab control (when the focus in the tab group) | Up Arrow, Down Arrow |
| Move between tabs in side tab control (when focus on Page) | Ctrl+Page Up/Ctrl+Page Down |
| Move from page to tab group (from next control in editor) | Ctrl+Tab |
| Move from page to tab group (from any control in editor) | Ctrl+Up Arrow |
| Open and close Sections (when focus is on a section header) | Enter, Spacebar, Right Arrow/Left Arrow |

### Component and Resource Palettes

The following table describes the keyboard actions to perform navigation tasks involving palettes.

*Table 3–20 Keyboard Navigation for Component and Resource Palettes*

| Navigation | Keys |
|---|---|
| Navigate forward in or out of palette | Ctrl+Tab<br>This moves you into first item within the pane. |
| Navigate backward in or out of palette | Ctrl+Shift+Tab |

*Table 3–20 (Cont.) Keyboard Navigation for Component and Resource Palettes*

| Navigation | Keys |
|---|---|
| Move between elements including dropdown lists, search fields, panels, tree structure (but not individual elements in a tree), individual component buttons | Tab, Shift+Tab |
| Move up/down elements in a list or tree | Up Arrow/Down Arrow |
| Move left/right elements in a list or tree | Left Arrow/Right Arrow |
| Move to start/end of data (component buttons) | Ctrl+Home/Ctrl+End |
| Select a component button | Enter |

### Navigators

The following table describes the keyboard actions to perform navigation tasks involving navigators.

*Table 3–21 Keyboard Navigation for Navigators*

| Navigation | Keys |
|---|---|
| Navigate forward in or out of navigator | Ctrl+Tab<br>This moves you into first item within the pane. |
| Navigate backward in or out of navigator | Ctrl+Shift+Tab |
| Move between elements including dropdown lists, search fields, panels, tree structure (but not individual elements in a tree), individual component buttons | Tab |
| Move up/down elements in a list or tree | Up Arrow/Down |
| Move left/right elements in a list or tree | Left Arrow/Right Arrow |
| Move to start/end of data (component buttons) | Ctrl+Home/Ctrl+End |
| Select a component button | Enter |
| Select an element | Enter |

### Property Inspector

The following table describes the keyboard actions to perform navigation tasks involving the Property Inspector.

*Table 3–22 Keyboard Navigation for the Property Inspector*

| Navigation | Keys |
|---|---|
| Navigate forward into or out of Property Inspector | Ctrl+Tab |
| Navigate backward into or out of Property Inspector | Ctrl+Shift+Tab |
| Navigate from side tab group to page | Tab |
| Navigate backward and forwards between elements on page | Tab, Shift+Tab |
| Move to tab above/below (when focus is on the side tab) | Up Arrow, Down Arrow |
| Move to tab right or left, above or below (when focus is on the internal tab group) | Up Arrow, Down Arrow, Right Arrow, Left Arrow |
| Move from side tab group to page | Ctrl+Down Arrow |
| Move from page to side tab group | Ctrl+Up Arrow |
| Move to side tab above (previous) when focus on page | Ctrl+Page Up |
| Move to side tab below (next) when focus on page | Move to side tab below (next) when focus on page |
| Open and Close sections (when focus is on a section header) | Enter |

### Text Editors

The following table describes the keyboard actions to perform navigation tasks involving the pane elements of text editors.

*Table 3–23 Keyboard Navigation for Text Editors*

| Navigation | Keys |
|---|---|
| Navigate forward in or out of editor | Ctrl+Tab |
| Navigate backward in or out of editor | Ctrl+Shift+Tab |
| Move from page to previous page | Alt+Page Up |
| Move from page to next page | Alt+Page Down |

The following table describes the keyboard actions to perform navigation tasks involving the text or canvas areas of text editors.

*Table 3–24 Keyboard Navigation for Canvas Areas of Text Editors*

| Navigation | Keys |
|---|---|
| Move up/down one line | Up Arrow, Down Arrow |
| Move left/right one character | Left Arrow, Right Arrow |
| Move to start/end of line | Home, End |
| Move to previous/next word | Ctrl+Left Arrow, Ctrl+Right Arrow |
| Move to start/end of text area | Ctrl+Home/Ctrl+End |
| Move to beginning/end of data | Ctrl+Home/Ctrl+End |
| Move up/down one vertical block | Page Up/Page Down |
| Block move left | Ctrl+Page Up |
| Block move right | Ctrl+Page Down |
| Block extend up | Shift+Page Up |
| Block extend down | Shift+Page Down |
| Block extend left | Ctrl+Shift+Page Up |
| Block extend right | Ctrl+Shift+Page Down |
| Select all | Ctrl+A |
| Deselect all | Up Arrow, Down Arrow, Left Arrow, Right Arrow |
| Extend selection up/down one line | Shift+Up Arrow/Shift+Down Arrow |
| Extend selection left/right one component or char | Shift+Left Arrow/Shift+Right Arrow |
| Extend selection to start/end of line | Shift+Home/Shift+End |
| Extend selection to start/end of data | Ctrl+Shift+Home/Ctrl+Shift+End |
| Extend selection up/down one vertical block | Shift+Page Up/Shift+Page Down |
| Extend selection to previous/next word | Ctrl+Shift+Left Arrow /Ctrl+Shift+Right Arrow |
| Extend selection left/right one block | Ctrl+Shift+Page Up/Ctrl+Shift+Page Down |
| Copy selection | Ctrl-C |
| Cut selection | Ctrl-X |
| Paste selected text | Ctrl-V |

### Graphical Editors

The following table describes the keyboard actions to perform navigation tasks involving graphical editors.

*Table 3–25 Keyboard Navigation for Graphical Editors*

| Navigation | Keys |
|---|---|
| Navigate forward in or out of editor | Ctrl-Tab |
| Navigate backward in or out of editor | Ctrl+Shift+Tab |
| Move from page to previous page | Alt+Page Up |
| Move from page to next page | Alt+Page Down |

The following table describes the keyboard actions to perform navigation tasks involving the canvas areas of graphical editors.

*Table 3–26 Keyboard Navigation for Canvas Areas of Graphical Editors*

| Navigation | Keys |
|---|---|
| Move to the next focusable element within editor area | Up Arrow, Down Arrow, Left Arrow, Right Arrow |
| Select element | Spacebar |
| Activate context menu | Shift+F10 |

# 4

# Creating the Source Files for an ADF Skin

This chapter describes how to create the source files for an ADF skin in the ADF Skin Editor and in JDeveloper. Information on how to open an application or project in the ADF Skin Editor that was created in a prior release of JDeveloper and how to import an ADF skin from an ADF Library JAR file is also provided.

This chapter includes the following sections:



## 4.1 About Creating an ADF Skin

An ADF skin defines the properties for the selectors that ADF Faces and ADF Data Visualization components expose. Using the visual editor in JDeveloper or the ADF Skin Editor, you can create a source file for an ADF skin. As a source file for an ADF skin is a type of CSS file, you could create it without using an editor. However, when you use the editor, associated configuration files get created (the first time that you create an ADF skin) or modified (when you create subsequent ADF skins). For more information about these configuration files, see Section 11.3, "Configuration Files for an ADF Skin."

## 4.2 Creating ADF Skin Applications and ADF Skin Projects

New ADF skin applications and ADF skin projects can be created in the ADF Skin Editor.

### 4.2.1 How to Create an ADF Skin Application

This section describes how to create an ADF skin application and a project within it in the ADF Skin Editor.

**To create a new ADF skin application:**

1. Open the Create ADF Skin Application dialog by choosing **File > New > ADF Skin Application**.

2. In the Create ADF Skin Application dialog, enter application details like the name and directory. For help with the wizard, press F1.
3. Click **Next** to open the ADF Skin Project page where you specify the name of your ADF skin project and the directory to store it.
4. In the Target Application Release list, select the release of Oracle ADF that the application you want to skin uses.

   The ADF Skin Editor configures your ADF skin project appropriately for the release you specify. For example, the ADF Skin Editor filters the list of ADF skins that you can extend from, as described in Section 4.4.1, "How to Create an ADF Skin in the ADF Skin Editor." The ADF Skin Editor also filters the list of skin selectors to display only those that the release you target supports. It will not display a skin selector introduced in a later release if you target your ADF skin project at an earlier release.
5. When you are done, click **Finish**.

## 4.2.2 How to Create a New ADF Skin Project

You use the Application Navigator to keep track of the ADF skin projects (collections of source files for ADF skins, images, and related files) you use while developing your ADF skin application.

You can create a new empty ADF skin project in an ADF skin application.

All ADF skin projects inherit the settings specified in the Default Project Properties dialog. As soon as you create the ADF skin project, it is added to the active ADF skin application.

**To create a new ADF skin project:**

1. In the Application Navigator, select the ADF skin application within which the project will appear.
2. Open the Create ADF Skin Project dialog by choosing **File > New > ADF Skin Project**.
3. In the Create ADF Skin Project dialog, enter project details like the name and directory.
4. In the Target Application Release list, select the release of Oracle ADF that the application you want to skin uses.

   The ADF Skin Editor configures your ADF skin project appropriately for the release you specify. For example, the ADF Skin Editor filters the list of ADF skins that you can extend from, as described in Section 4.4.1, "How to Create an ADF Skin in the ADF Skin Editor." The ADF Skin Editor also filters the list of skin selectors to display only those that the release you target supports. It will not display a skin selector introduced in a later release if you target your ADF skin project at an earlier release.
5. When you are done, click **Finish**.

The new ADF skin project appears in the Application Navigator. It inherits whatever default properties you've already set. To alter project properties for this project, either double-click the filename or right-click and choose **Project Properties**.

## 4.3 Opening an Application Created Outside of the ADF Skin Editor

When you open an application or project that was created in a prior release of JDeveloper, the ADF Skin Editor will prompt you to migrate the project to JDeveloper 11g format. Depending on the content of the project, the ADF Skin Editor may display additional prompts to migrate some specific source files as well. Oracle recommends that you make a backup copy of your projects before you open them in the ADF Skin Editor or migrate them using the ADF Skin Editor.

## 4.4 Creating an ADF Skin File

You can create an ADF skin file in the ADF Skin Editor or in JDeveloper that defines how ADF Faces and ADF Data Visualization components render at runtime. The ADF skin that you create must extend either one of the ADF skins that Oracle ADF provides or from an existing ADF skin that you created. The ADF skins that Oracle ADF provides vary in the level of customization that they define for ADF Faces and ADF Data Visualization components. For information about the inheritance relationship between the ADF skins that Oracle ADF provides, see Section 1.5, "Inheritance Relationship of the ADF Skins Provided by Oracle ADF." For information about the levels of customization in the ADF skins provided by Oracle ADF and for a recommendation about the ADF skin to extend, see Section 11.4, "ADF Skins Provided by Oracle ADF."

The visual editor of the ADF Skin Editor and in JDeveloper supports the creation of ADF skins for the `org.apache.myfaces.trinidad.desktop` render kit.

You can create ADF skins for other render kits using the source editor in JDeveloper and in the ADF Skin Editor. For more information, see Section 11.2, "ADF Skinning Framework and Supported Render Kits."

After you create an ADF skin, you set values for the selectors that the ADF Faces and ADF Data Visualization components expose. Otherwise, the ADF skin that you create defines the same appearance as the ADF skin from which it extends. For more information, see Chapter 5, "Working with Component-Specific Selectors."

### 4.4.1 How to Create an ADF Skin in the ADF Skin Editor

You can create an ADF skin in the ADF Skin Editor.

**To create an ADF skin in the ADF Skin Editor:**

1. In the Application Navigator, right-click the project where you want to create the new ADF skin and choose **New > ADF Skin File**.
2. In the Create ADF Skin File dialog, enter the following:
   - **File Name:** Enter a file name for the new ADF skin.
   - **Directory:** Enter the path to the directory where you store the CSS source file for the ADF skin or accept the default directory proposed by the editor.
   - **Family:** Enter a value for the family name of your skin.

     You can define a new family or select an existing family by entering a value in the input field. A *family* groups together ADF skins for an application. You configure an application to use a particular family of ADF skin.

     The value you enter must be unique. You can use an EL expression to select an ADF skin for your application at runtime by referencing this value.

- **Use as the default skin family for this project:** Deselect this checkbox if you do not want to make the ADF skin the default for your project immediately.
- **Extends:** Select the ADF skin that you want to extend. ADF Faces provides a number of ADF skins that you can extend. For more information and a recommendation on the ADF skin to extend, see Section 11.4, "ADF Skins Provided by Oracle ADF."

---

**Note:** The value you select for Target Application Release, as described in Section 4.2, "Creating ADF Skin Applications and ADF Skin Projects," determines the list of ADF skins from which you can extend.

---

- **Skin Id:** A read-only field that displays a concatenation of the value you enter in **File Name** and the ID of the render kit (`desktop`) for which you create your ADF skin. You select this value from the **Extends** list if you want to create another ADF skin that extends from this one.

  The ADF Skin Editor writes the value to the `<id>` element in the `trinidad-skins.xml` file.

3. Click **OK**.

## 4.4.2 How to Create an ADF Skin in JDeveloper

You can create an ADF skin in JDeveloper.

**To create an ADF skin in JDeveloper:**

1. In the Application Navigator, right-click the project that contains the code for the user interface and choose **New**.
2. In the New Gallery, expand **Web Tier**, select **JSF/Facelets** and then **ADF Skin**, and click **OK**.
3. In the Create ADF Skin File dialog, enter the following:
   - **File Name:** Enter a file name for the new ADF skin.
   - **Directory:** Enter the path to the directory where you store the CSS source file for the ADF skin or accept the default directory proposed by the editor.
   - **Family:** Enter a value for the family name of your skin.

     You can define a new family or select an existing family by entering a value in the input field. A *family* groups together ADF skins for an application. You configure an application to use a particular family of ADF skin.

     The value you enter must be unique. You can use an EL expression to select an ADF skin for your application at runtime by referencing this value.
   - **Use as the default skin family for this project:** Deselect this checkbox if you do not want to make the ADF skin the default for your project immediately. If you select the checkbox, the `trinidad-config.xml` file is updated, as described in Section 4.4.3, "What Happens When You Create an ADF Skin."
   - **Extends:** Select the ADF skin that you want to extend. ADF Faces provides a number of ADF skins that you can extend. For more information and a recommendation on the ADF skin to extend, see Section 11.4, "ADF Skins Provided by Oracle ADF."

- **Skin Id:** A read-only field that displays a concatenation of the value you enter in **File Name** and the ID of the render kit (`desktop`) for which you create your ADF skin. You select this value from the **Extends** list if you want to create another ADF skin that extends from this one.

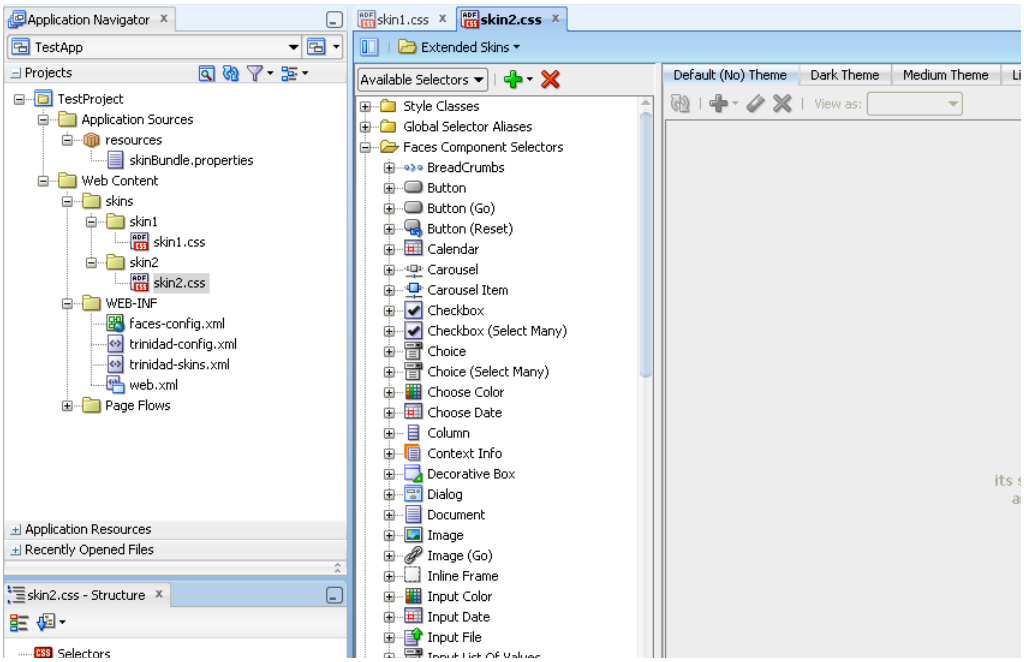
  JDeveloper writes the value to the `<id>` element in the `trinidad-skins.xml` file.

4. Click **OK**.

## 4.4.3 What Happens When You Create an ADF Skin

If you accepted the default value proposed for the Directory field, a file with the extension `.css` is generated in a subdirectory of the `skins` directory in your project. This file is opened in the visual editor for the ADF skin, as illustrated in Figure 4–1.

***Figure 4–1 Newly-Created ADF Skin***




The `trinidad-skins.xml` file is modified to include metadata for the ADF skin that you create, as illustrated in Example 4–1.

***Example 4–1 trinidad-skins.xml File***

```
<?xml version="1.0" encoding="windows-1252"?>
<skins xmlns="http://myfaces.apache.org/trinidad/skin">
  ....
  <skin>
    <id>skin2.desktop</id>
    <family>skin2</family>
    <extends>fusionFx-v1.desktop</extends>
    <render-kit-id>org.apache.myfaces.trinidad.desktop</render-kit-id>
    <style-sheet-name>skins/skin2/skin2.css</style-sheet-name>
  </skin>
</skins>
```

If you select the **Use as the default skin family for this project** check box in the Create New ADF Skin file dialog, the `trinidad-config.xml` file is modified to make the new ADF skin the default skin for your application. Example 4–2 shows a `trinidad-config.xml` file that makes the ADF skin in Example 4–1 the default for an application.

***Example 4–2 trinidad-config.xml File***

```
<?xml version="1.0" encoding="windows-1252"?>
<trinidad-config xmlns="http://myfaces.apache.org/trinidad/config">
  <skin-family>skin2</skin-family>
</trinidad-config>
```

The source file for the ADF skin contains a comment and namespace references, as illustrated in Example 4–3. These entries in the source file for the ADF skin distinguish the file from non-ADF skin files with the `.css` file extension. A source file for an ADF skin requires these entries in order to open in the visual editor for the ADF skin.

***Example 4–3 Default Entries for a Newly Created ADF Skin File***

```
/**ADFFaces_Skin_File / DO NOT REMOVE**/
@namespace af "http://xmlns.oracle.com/adf/faces/rich";
@namespace dvt "http://xmlns.oracle.com/dss/adf/faces";
```

The first time that you create an ADF skin in your project, a resource bundle file (`skinBundle.properties`) is generated, as illustrated in Figure 4–1. For more information about using resource bundles, see Chapter 7, "Working With Text in an ADF Skin."

## 4.5 Versioning ADF Skins

You can specify version numbers for your ADF skins in the `trinidad-skins.xml` file using the `<version>` element. Use this capability if you want to distinguish between ADF skins that have the same value for the `<family>` element in the `trinidad-skins.xml` file. Note that when you configure an application to use a particular ADF skin, you do so by specifying values in the `trinidad-config.xml` file, as described in section Section 10.4, "Applying an ADF Skin to Your Web Application."

### 4.5.1 How to Version an ADF Skin

You specify a version for your ADF skin by entering a value for the `<version>` element in the `trinidad-skins.xml` file.

**To version an ADF skin:**

1. In the Application Navigator, double-click the `trinidad-skins.xml` file. By default, this is in the **Web Content/WEB-INF** node.
2. In the Structure window, right-click the **skin** node for the ADF skin that you want to version and choose **Insert inside skin** > **version**.
3. In the Insert version dialog, select **true** from the default list if you want your application to use this version of the ADF skin when no value is specified in the `<skin-version>` element of the `trinidad-config.xml` file, as described in Section 10.4, "Applying an ADF Skin to Your Web Application."
4. Enter a value in the name field. For example, enter `v1` if this is the first version of the ADF skin.

5. Click **OK**.

### 4.5.2 What Happens When You Version ADF Skins

Example 4–4 shows an example `trinidad-skins.xml` that references three source files for ADF skins (`skin1.css`, `skin2.css`, and `skin3.css`). Each of these ADF skins have the same value for the `<family>` element (`test`). The values for the child elements of the `<version>` elements distinguish between each of these ADF skins. At runtime, an application that specifies `test` as the value for the `<skin-family>` element in the application's `trinidad-config.xml` file uses `skin3` because this ADF skin is configured as the default skin in the `trinidad-skins.xml` file (`<default>true</default>`). You can override this behavior by specifying a value for the <skin-version> element in the trinidad-config.xml file, as described in Section 10.4, "Applying an ADF Skin to Your Web Application."

***Example 4–4 trinidad-skins.xml with versioned ADF skin files***

```
<?xml version="1.0" encoding="windows-1252"?>
<skins xmlns="http://myfaces.apache.org/trinidad/skin">
  <skin>
    <id>skin1.desktop</id>
    <family>test</family>
    <extends>fusionFx-simple-v1.desktop</extends>
    <render-kit-id>org.apache.myfaces.trinidad.desktop</render-kit-id>
    <style-sheet-name>skins/skin1/skin1.css</style-sheet-name>
    <version>
      <name>v1</name>
    </version>
  </skin>
  <skin>
    <id>skin2.desktop</id>
    <family>test</family>
    <extends>skin1.desktop</extends>
    <render-kit-id>org.apache.myfaces.trinidad.desktop</render-kit-id>
    <style-sheet-name>skins/skin2/skin2.css</style-sheet-name>
    <version>
      <name>v2</name>
    </version>
  </skin>
  <skin>
    <id>skin3.desktop</id>
    <family>test</family>
    <extends>skin2.desktop</extends>
    <render-kit-id>org.apache.myfaces.trinidad.desktop</render-kit-id>
    <style-sheet-name>skins/skin3/skin3.css</style-sheet-name>
    <version>
      <default>true</default>
      <name>v3</name>
    </version>
  </skin>
</skins>
```

## 4.6 Managing Working Sets

Working sets allow you to configure the Application Navigator to show you a subset of files from your project. This is particularly useful when working with large projects.

Before you define your own working sets the only one available is Default, and it is a working set which includes all the files in the current application.

You can define a working set by selecting from files or containers in the Application Navigator, or by providing include and exclude filter patterns through the Manage Working Sets dialog.

#### To group objects in the Application Navigator into a working set:

1. In the Application Navigator, select the objects that you want to include in a new working set.
2. In the Application Navigator, click the **Working Sets** icon and select **New from Selection**.

   This opens a Save As dialog. For more information at any time, press F1 or click **Help** from within the Save As dialog.
3. Enter a name for the working set, then click **OK**.

#### To create a working set by defining file and directory filters:

1. In the Application Navigator, click the **Working Sets** icon and select **Manage Working Sets**.

   This opens the Working Sets dialog. Use the tree on the left to select the projects to include. In the right panel, select which files in the current project to include. For more information at any time, press F1 or click **Help** from within the Working Sets dialog.
2. Click **Save As** to save the working set.

#### To create a working set from the results of a search in the Log window:

1. In the Log window, right-click and choose **Save as Working Set** from the context menu.
2. In the Create Working Set dialog, enter a name for the working set.

#### To see which working set you are currently using:

- In the Application Navigator, hover the mouse over the **Working Sets** icon. The name of the current working set is displayed as a tooltip. Alternatively, click the **Working Sets** icon to bring up a menu in which the active working set is checked.

#### To change the active working set:

- In the Application Navigator, click the **Working Sets** icon and select the working set you want to open.

  Files not belonging to the working set are removed from view.

#### To edit files and projects in a working set:

1. In the Application Navigator, click the **Working Sets** icon and select **Manage Working Sets**.

   This opens the Working Sets dialog. For more information at any time, press F1 or click Help from within the Working Sets dialog.
2. Select the working set that you want to change from the Working Set drop-down list.
3. Make the changes as required.

**To restore the view in the Application Navigator to show all files:**

- In the Application Navigator, click the **Working Sets** icon and select **(All Files)**.

## 4.7 Importing ADF Skins from an ADF Library JAR

You can import ADF skins into your project that have been packaged in a JAR file. When you import an ADF skin into your project, the imported ADF skin is available to extend from when you create a new ADF skin, as described in Section 4.4, "Creating an ADF Skin File."

The recommended type of JAR file to use to package an ADF skin is an ADF Library JAR file. For information about how to package an ADF skin into this type of JAR file, see Section 10.3, "Packaging an ADF Skin into an ADF Library JAR."

You can import an ADF skin that you have packaged in other types of JAR file. For these ADF skins to appear in the user interface as a choice to extend from when you create a new ADF skin, your JAR file must have the same directory structure shown in Example 4–5. Your JAR file must also include an `oracle.adf.common.services.ResourceService.sva` file. You can generate this file by following the instructions in Section 10.3, "Packaging an ADF Skin into an ADF Library JAR."

Images referenced by the ADF skin you want to import must appear under a directory named `adf`, as shown in Example 4–5.

***Example 4–5 Required Directory Structure and Files for a non-ADF Library JAR File***

```
META-INF
|   MANIFEST.MF
|   oracle.adf.common.services.ResourceService.sva
|   trinidad-skins.xml
|
+---adf
|   \---skins
|       \---jarredskin
|           \---images
|               \---af_column
|                       sort_des_selected.png
|
\---skins
    \---jarredskin
            jarredskin.css
```

### 4.7.1 How to Import an ADF Skin from an ADF Library JAR

You can import ADF skins into your project that have been packaged in a JAR file.

**To import an ADF skin from an ADF Library JAR:**

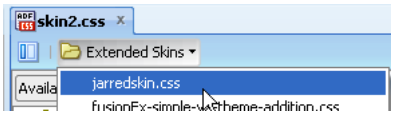
1. From the main menu, choose **Application > Project Properties**.
2. In the Project Properties dialog, select the **Libraries and Classpath** page and click **Add JAR/Directory**.
3. In the Add Archive or Directory dialog, navigate to the JAR file that contains the ADF skin you want to import and click **Select**.

   The JAR file appears in the Classpath Entries list.
4. When finished, click **OK**.

## 4.7.2 What Happens When You Import an ADF Skin from an ADF Library JAR

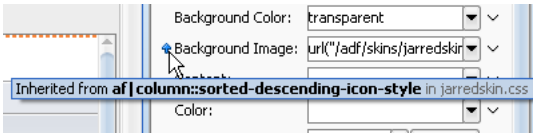
The ADF skin(s) that you import from the JAR file appear in the Extends list when you create a new ADF skin, as described in Section 4.4, "Creating an ADF Skin File." After you create a new ADF skin by extending an ADF skin that you imported from a JAR file, the Extended Skins list in the Preview Pane displays the name of the ADF skin that you imported. For example, in Figure 4–2 the `skin2.css` ADF skin has been created by extending the ADF skin, `jarredskin.css`, that was imported into the project from a JAR file.

***Figure 4–2 Imported ADF Skin in the Extended Skins List***




Properties that have been defined in the ADF skin that you imported appear with a blue upward pointing arrow in the Property Inspector. An information tip about the inheritance relationship displays when you hover the mouse over the property, as illustrated in Figure 4–3.

***Figure 4–3 Property Inherited from an Imported ADF Skin***



# 5

# Working with Component-Specific Selectors

This chapter describes how to change the appearance of ADF Faces and ADF Data Visualization components by specifying properties for the selectors that the ADF skinning framework exposes for these components. Features such as the ability to configure ADF skin properties to apply to messages, themes that you can apply to ADF Faces components, and how to configure an ADF skin for accessibility are also described.

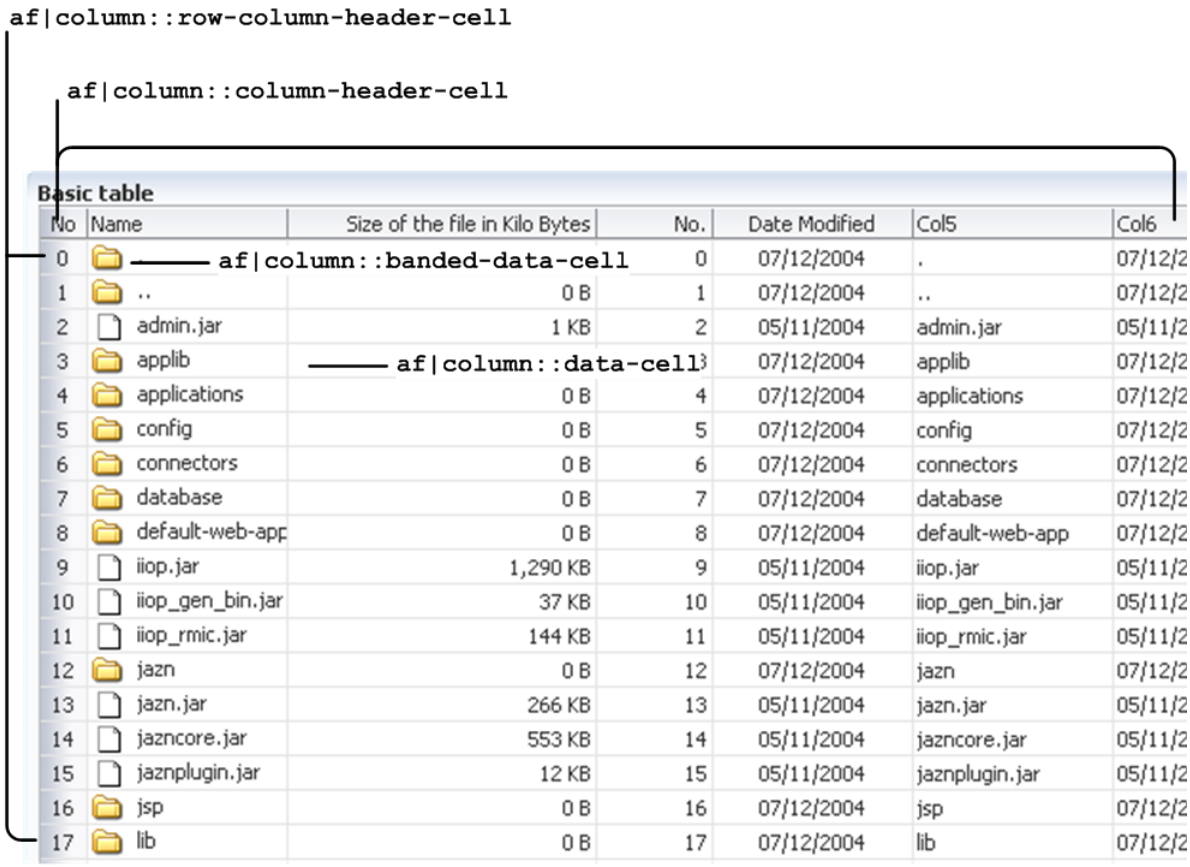
This chapter includes the following sections:



## 5.1 About Working with Component-Specific Selectors

You customize the appearance of an ADF Faces component or ADF Data Visualization component by defining style properties for the selectors that the component exposes. To achieve the appearance you want, you need to become familiar with the component-specific selectors that the ADF Faces and ADF Data Visualization components expose, plus the global selector aliases and descendant selectors that a component-specific selector may reference. The ADF skins that you extend from when you create an ADF skin define many global selector aliases and descendant selectors. You also need to become familiar with the component itself and how it relates to other components. For example, customizing the appearance of the ADF Faces `table` component shown in Figure 5–1 requires you to define style properties for selectors exposed by the `af:column` component in addition to selectors exposed by the ADF Faces `table` component. You may also need to modify the style properties for one or more of the icon or message global selector aliases that the ADF skin you extend defines.

***Figure 5–1 Selectors for an ADF Faces table Component***




| No | Name | Size of the file in Kilo Bytes | No. | Date Modified | Col5 | Col6 |
|---|---|---|---|---|---|---|
| 0 | | | 0 | 07/12/2004 | . | 07/12/2 |
| 1 | .. | 0 B | 1 | 07/12/2004 | .. | 07/12/2 |
| 2 | admin.jar | 1 KB | 2 | 05/11/2004 | admin.jar | 05/11/2 |
| 3 | applib | | 3 | 07/12/2004 | applib | 07/12/2 |
| 4 | applications | 0 B | 4 | 07/12/2004 | applications | 07/12/2 |
| 5 | config | 0 B | 5 | 07/12/2004 | config | 07/12/2 |
| 6 | connectors | 0 B | 6 | 07/12/2004 | connectors | 07/12/2 |
| 7 | database | 0 B | 7 | 07/12/2004 | database | 07/12/2 |
| 8 | default-web-app | 0 B | 8 | 07/12/2004 | default-web-app | 07/12/2 |
| 9 | iiop.jar | 1,290 KB | 9 | 05/11/2004 | iiop.jar | 05/11/2 |
| 10 | iiop_gen_bin.jar | 37 KB | 10 | 05/11/2004 | iiop_gen_bin.jar | 05/11/2 |
| 11 | iiop_rmic.jar | 144 KB | 11 | 05/11/2004 | iiop_rmic.jar | 05/11/2 |
| 12 | jazn | 0 B | 12 | 07/12/2004 | jazn | 07/12/2 |
| 13 | jazn.jar | 266 KB | 13 | 05/11/2004 | jazn.jar | 05/11/2 |
| 14 | jazncore.jar | 553 KB | 14 | 05/11/2004 | jazncore.jar | 05/11/2 |
| 15 | jaznplugin.jar | 12 KB | 15 | 05/11/2004 | jaznplugin.jar | 05/11/2 |
| 16 | jsp | 0 B | 16 | 07/12/2004 | jsp | 07/12/2 |
| 17 | lib | 0 B | 17 | 07/12/2004 | lib | 07/12/2 |

Use the tools that the visual editor for ADF skins provides to customize the appearance of the ADF Faces components and ADF Data Visualization components. For more information, see Chapter 3, "Working with the Oracle ADF Skin Editor."

Other sources of information that may help you as you change the selectors of ADF Faces and ADF Data Visualization components include the following:

- **Images**: An ADF skin can reference images that render icons and logos, for example, in a page. For more information about how to work with images in an ADF skin, see Chapter 6, "Working with Images in Your ADF Skin."
- **Text**: An ADF skin does not include text strings that render in your page. However, you can specify a resource bundle that defines the text strings you want to appear in the page. For more information, see Chapter 7, "Working With Text in an ADF Skin."
- **Global selector aliases**: A global selector alias specifies style properties that you can apply to multiple ADF Faces components simultaneously. For more information about global selector aliases, see Chapter 8, "Working With Global Selector Aliases."
- **Style Classes**: A style class in an ADF skin specifies a number of style properties that an ADF Faces component can reference as a value if it exposes a style-related attribute (`styleClass` and `inlineStyle`). For more information about style classes, see Chapter 9, "Working with Style Classes."
- **ADF Faces Rich Client Components Hosted Demo**: The Oracle Technology Network (OTN) web site provides a link to an application that demonstrates how

ADF skins change the appearance of ADF Faces and ADF Data Visualization components. For more information, navigate to http://www.oracle.com/technetwork/developer-tools/adf/overview/index.html

## 5.2 Changing ADF Faces Components' Selectors

ADF Faces components render user interface controls such as command buttons, command links and check boxes in your web application. ADF Faces components also include components that render calendars, panels to arrange other user interface controls and tables in your web application. For more information about ADF Faces components and the functionality that they provide, see the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* (for the release that pertains to the application you are skinning).

You can change the runtime appearance of ADF Faces components by editing the properties that each ADF Faces skin selector exposes. The number of selectors that an ADF Faces component exposes varies by component. For example, the ADF Faces components, `af:image` and `af:popup`, expose one selector each. In contrast, the ADF Faces component, `af:panelHeader`, exposes a variety of selectors that enable you to change the appearance of different parts of the user interface of that component. There are, for example, selectors that allow you to change the `af:panelHeader` component's instruction text, help icons, and titles.

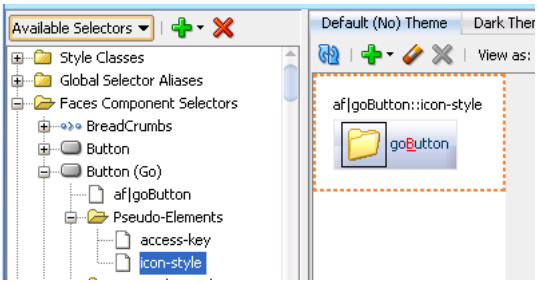
The process to follow to change the runtime appearance of an ADF Faces component is the same for each component; the only difference is the number of selectors that each ADF Faces component exposes. Figure 5–2 and Figure 5–3 take the `goButton` component as an example and illustrate how you can customize the appearance of this component using pseudo-elements and the component's selector. Figure 5–2 shows the application of the default ADF Faces' `fusion` skin on the `goButton` component and the component icon.

***Figure 5–2 goButton Component Default Appearance with Fusion ADF Skin***


Figure 5–3 shows the appearance of the component after you set values for the following pseudo-elements on the component's selector:

- **access-key**: The Color property is set to `red`
- **icon-style**: The Border property is set to `1px solid black`

***Figure 5–3 goButton Component with Modified Selectors***



Reference information about the selectors that ADF Faces components expose can be found in the *Oracle Fusion Middleware Tag Reference for Oracle ADF Faces Skin Selectors* (for the release that pertains to the application you are skinning).

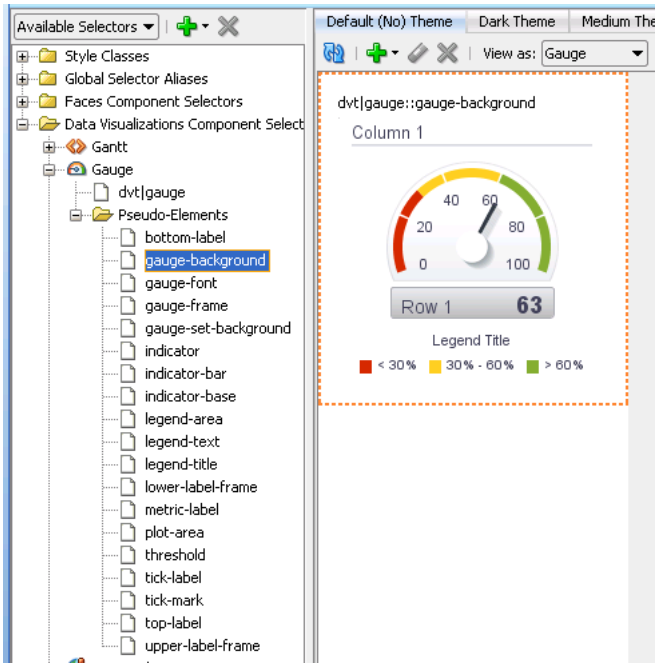
## 5.3 Changing ADF Data Visualization Components' Selectors

The ADF Data Visualization components are a set of components that provide functionality to represent data in graphical and tabular formats. Examples of the ADF Data Visualization components include the following: graph, gantt, pivot table, and hierarchy viewer. For more information about ADF Data Visualization components and the functionality that they provide, see the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* (for the release that pertains to the application you are skinning).

You can change the runtime appearance of ADF Data Visualization components by editing the properties that each ADF Data Visualization component selector exposes. The number of selectors exposed by an ADF Data Visualization component varies by component.

Figure 5–4 shows an ADF skin with the nodes expanded to show the selectors that you can customize for the ADF Data Visualization `gauge` component.

***Figure 5–4 ADF Data Visualization Component Selectors***




You customize the appearance of ADF Data Visualization components by defining style properties for the selectors that each ADF Data Visualization component exposes. Using the tools provided by JDeveloper's visual editor for ADF skins or the ADF Skin Editor, you customize the appearance of the ADF Data Visualization components. For more information, see Chapter 3, "Working with the Oracle ADF Skin Editor."

To achieve the appearance you want, you need to become familiar with the selectors that the ADF Data Visualization component exposes, the global selector aliases that the component may reference and which are defined in the ADF skin that you extend when you create an ADF skin. You also need to become familiar with the component itself and how it relates to other components. For example, customizing the

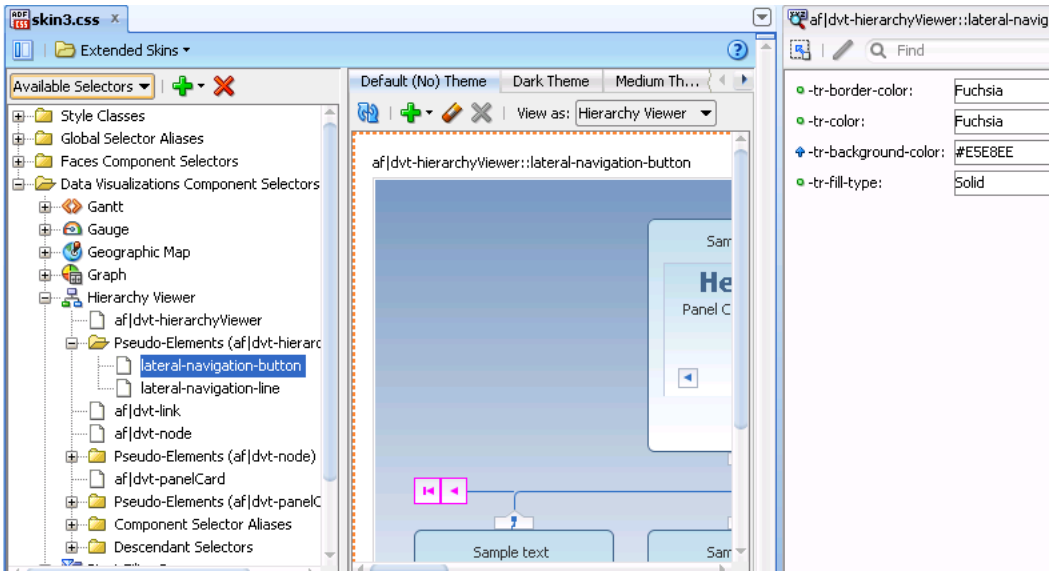
appearance of the ADF Data Visualization `pivotTable` component shown in Figure 5–5 requires you to define style properties for this selector's pseudo-elements. You may also need to modify the style properties for one or more of the global selector aliases that the ADF skin you extend defines.

***Figure 5–5 ADF Data Visualization pivotTable Component***

| | | Sales | | Units | |
|---|---|---|---|---|---|
| | | All Channels | | All Channels | |
| | | World | Boston | World | Boston |
| 2007 | Tents | 20,000 | 500 | 200 | 50 |
| | Canoes | 15,000 | 1,500 | 75 | 8 |
| 2006 | Tents | 10,000 | 250 | 100 | 25 |
| | Canoes | 7,500 | 750 | 40 | 4 |
| 2005 | Tents | 5,000 | 125 | 50 | 15 |
| | Canoes | 3,750 | 375 | 20 | 2 |

Many ADF Data Visualization component selectors, such as the selectors for the `graph` and `hierarchyViewer` components, expose pseudo-elements for which you configure ADF skin properties. These ADF skin properties modify the appearance of the area specified by the pseudo-element. The characters `-tr-` preface the names of ADF skin properties. For example, Figure 5–6 shows the properties of the hierarchy viewer's `lateral-navigation-button` selector, all of which are prefaced by `-tr-`.

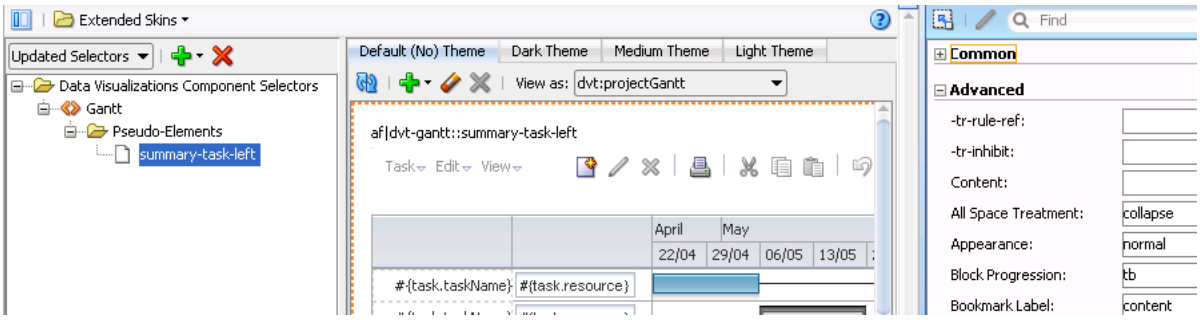
***Figure 5–6 Properties for the hierarchyViewer Component lateral-navigation-button Pseudo-Element***




In contrast, the `gantt` component's `summary-task-left` selector, shown in Figure 5–7, exposes only two ADF skin properties (`-tr-rule-ref` and `-tr-inhibit-`) as the majority of the properties that you configure for this selector are CSS properties.

For more information about ADF skin properties, see Section 2.3, "Properties in the ADF Skinning Framework."

***Figure 5–7 Properties for the gantt Component summary-task-left Pseudo-Element***




Reference information about the selectors, pseudo-elements, and pseudo-classes that ADF Data Visualization components expose can be found in the *Oracle Fusion Middleware Data Visualization Tools Tag Reference for Oracle ADF Skin Selectors* (for the release that pertains to the application you are skinning).

## 5.4 Changing a Component-Specific Selector

The process to change a component-specific selector is the same for both the ADF Faces and ADF Data Visualization components. In the Selector Tree, you expand the Faces Components Selectors or Data Visualization Selectors node to select the selector of the component you want to modify. You then set values for this selector using the Property Inspector. You can also set properties for any pseudo-elements, component style classes, component selector aliases, or descendant selectors that the selector you select references. In addition, you can add pseudo-classes that the component-specific supports. For more information about pseudo-classes, see Section 2.2, "Pseudo-Classes in the ADF Skinning Framework." Figure 5–8 shows a view of the skin selector for the ADF Faces `table` component in the Selector Tree with the different items that you can configure for this skin selector.

***Figure 5–8 Selector for the table Component***

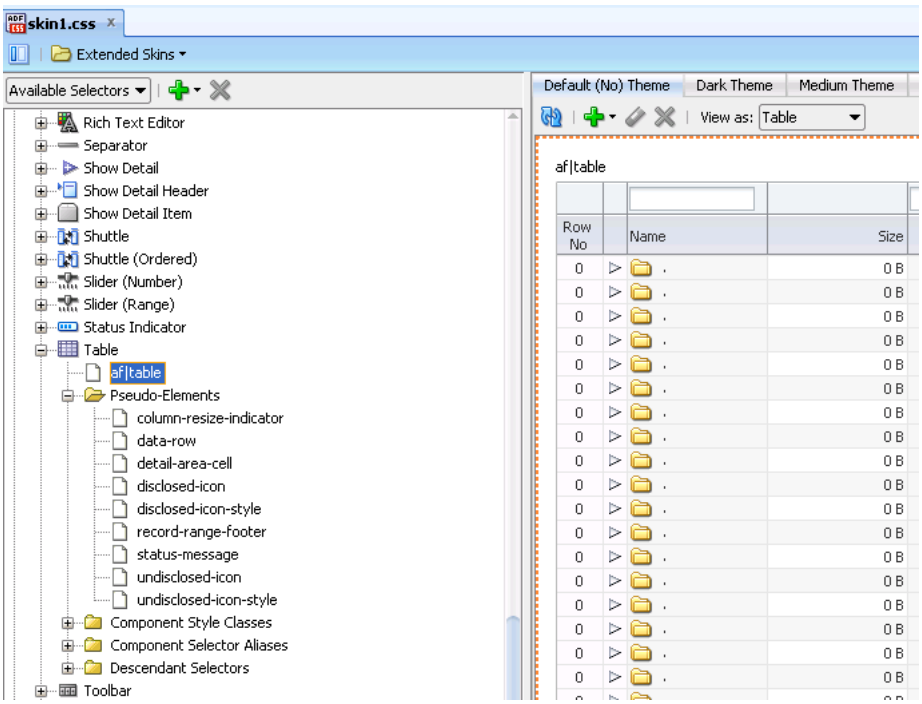
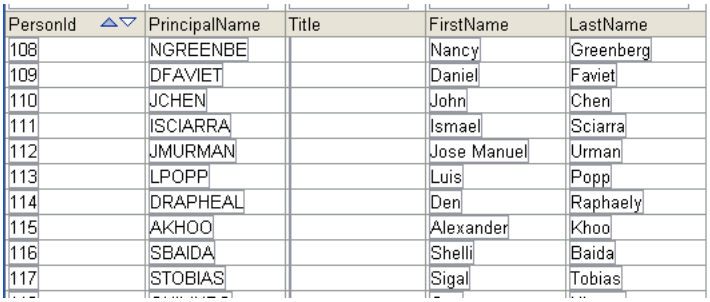


Figure 5–9 shows a runtime view of an ADF Faces `table` component that renders data using the style properties provided by the ADF Faces `simple` skin.

***Figure 5–9 ADF Faces table Component Rendered By the simple Skin***



| PersonId | PrincipalName | Title | FirstName | LastName |
|---|---|---|---|---|
| 108 | NGREENBE | | Nancy | Greenberg |
| 109 | DFAVIET | | Daniel | Faviet |
| 110 | JCHEN | | John | Chen |
| 111 | ISCIARRA | | Ismael | Sciarra |
| 112 | JMURMAN | | Jose Manuel | Urman |
| 113 | LPOPP | | Luis | Popp |
| 114 | DRAPHEAL | | Den | Raphaely |
| 115 | AKHOO | | Alexander | Khoo |
| 116 | SBAIDA | | Shelli | Baida |
| 117 | STOBIAS | | Sigal | Tobias |

## 5.4.1 How to Change a Component-Specific Selector

You change a component-specific selector by selecting the selector in the Selector Tree and setting values for the selector, its pseudo-elements, or descendant selectors in the Property Inspector. In addition, you can add a pseudo-class if the component-specific selector supports one.

**To change a component-specific selector:**

1. In the Selector Tree of the visual editor, choose the appropriate option:
   - Expand the **Faces Component Selectors** node if you want change a selector for an ADF Faces component.
   - Expand the **Data Visualization Selectors** node if you want to change a selector for an ADF Data Visualization component.

   For example, expand the Faces Component Selectors node, the Column node, the Pseudo-Elements node, and select the **column-header-cell** selector.
2. In the Property Inspector, specify values for the properties that the selector you selected in step 1 supports.

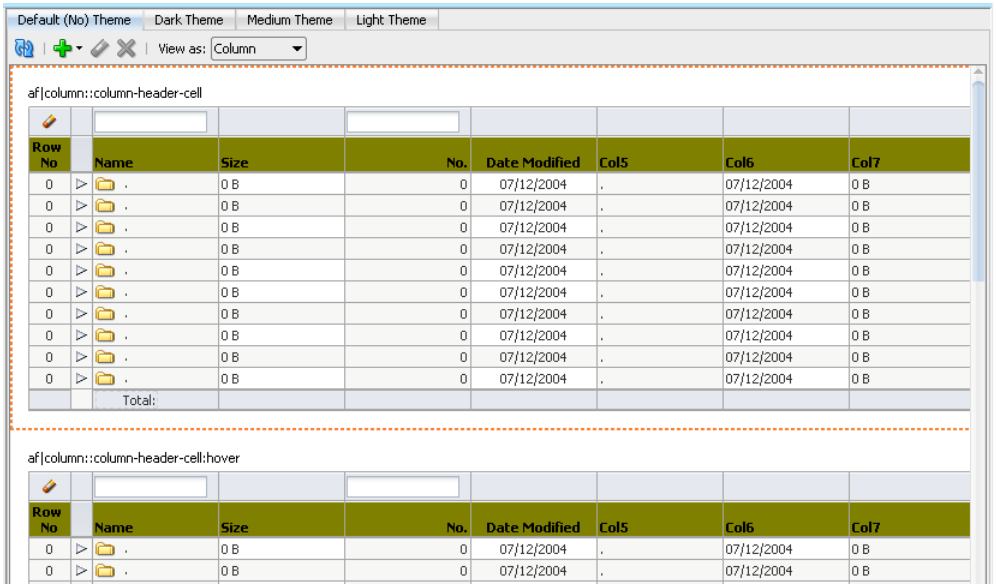
   For example, in the Common section of the Property Inspector, specify values for the following attributes:
   - **Background Color**: Specify the background color that you want to appear in the header row of the table.
   - **Color**: Specify the color that you want to apply to text that appears in the header row of the table's column.
3. In the Preview Pane, click the **Add Pseudo-Class** icon to choose a supported pseudo-class from the displayed list of supported pseudo-classes that appears.

## 5.4.2 What Happens When You Change a Component-Specific Selector

The visual editor displays the changes that you make to the selector after you click the **Refresh** icon in the Preview Pane. If you add a pseudo-class to the selector, the Preview Pane also displays an entry for the selector with the added pseudo-class. For example, Figure 5–10 shows an entry for a selector with the `:hover` pseudo-class added.

**Note:** The Preview Pane for the `af|document` selector only displays one entry even if you add a pseudo-class to this selector.

***Figure 5–10 Preview Pane with a Component Specific Selector and a Pseudo-Class***




The visual editor also writes the values that you specify for the selectors in the Property Inspector to the source file for the ADF skin. Example 5–1 shows the changes that appear in the source file after making some of the changes described in Section 5.4.1, "How to Change a Component-Specific Selector."

***Example 5–1 Selector Values to Skin the Header Row in a Column***

```
af|column::column-header-cell
{

  color: Black;
  background-color: Olive;
  font-weight: bold;
}
```

When a web application uses an ADF skin that contains the values shown in Example 5–1, header rows in the columns of a table rendered by the ADF Faces `table` component appear as illustrated by Figure 5–11 where the table uses a skin that extends the `simple` skin.

***Figure 5–11 ADF Faces table with a Header Row Skinned***

| PersonId | PrincipalName | Title | FirstName | LastName |
|---|---|---|---|---|
| 108 | NGREENBE | | Nancy | Greenberg |
| 109 | DFAVIET | | Daniel | Faviet |
| 110 | JCHEN | | John | Chen |
| 111 | ISCIARRA | | Ismael | Sciarra |
| 112 | JMURMAN | | Jose Manuel | Urman |
| 113 | LPOPP | | Luis | Popp |
| 114 | DRAPHEAL | | Den | Raphaely |
| 115 | AKHOO | | Alexander | Khoo |
| 116 | SBAIDA | | Shelli | Baida |
| 117 | STOBIAS | | Sigal | Tobias |

## 5.5 Configuring ADF Skin Properties to Apply to Messages

You can apply styles to ADF Faces input components based on whether or not the input components have certain types of message associated with them. When a message of a particular type is added to a component, the styles of that component are automatically modified to reflect the new status. If you do not define styles for the type of message in question, the component uses the default styles defined in the ADF skin.

The types of message property are:

- `:fatal`
- `:error`
- `:warning`
- `:confirmation`
- `:info`

Figure 5–12 shows an `inputText` component rendered using the `simple` ADF skin. In Figure 5–12, the `simple` ADF skin defines style values for the `:warning` message property to apply to the `inputText` component when an end user enters values that generate a warning.

***Figure 5–12 inputText Component Displaying Style for :warning Message Property***

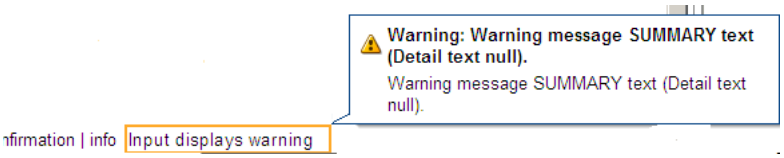
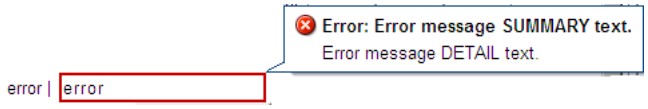


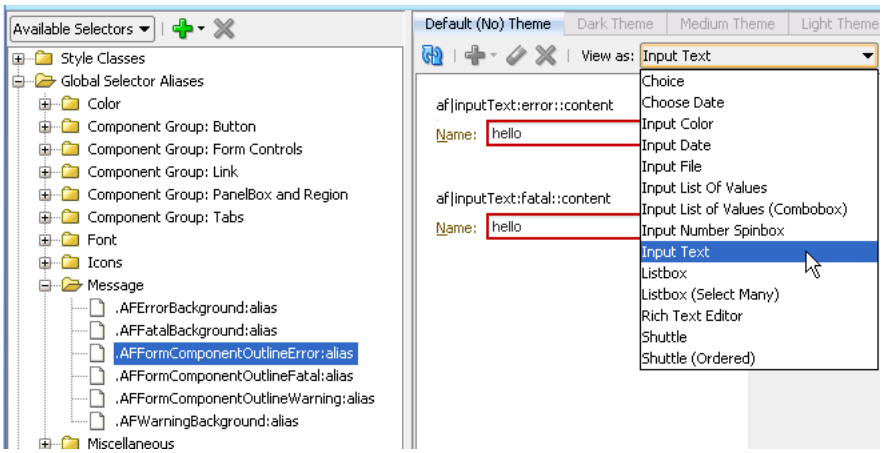

Figure 5–13 shows the same `inputText` component as in Figure 5–12. In Figure 5–13, the end user entered a value that generated an error. As a result, the `inputText` component renders using the style properties configured for the `:error` message property.

***Figure 5–13 inputText Component Displaying Style for :error Message Property***




The ADF skinning framework defines a number of global selector aliases that define style properties to apply to messages. Figure 5–14 shows a list of global selector aliases under the **Message** node in the Selector Tree. The Preview Pane, on the right of Figure 5–14, shows how the style properties defined for the global selector alias currently selected in the Selector Tree render the component selected from the **View as** list.

***Figure 5–14 Global Selector Aliases for Messages***




You can customize the global selector aliases that the ADF skinning framework provides for messages by defining style properties in your ADF skin. The style properties that you define for the global selector alias affect all ADF Faces components that reference the global selector alias. For example, if you change the border color for the global selector alias shown in Figure 5–14 to green, all the ADF Faces components shown in the **View as** list render with a border that is green. For more information about global selector aliases, see Chapter 8, "Working With Global Selector Aliases."

For more information about configuring messages for ADF Faces components, see the "Displaying Tips, Messages, and Help" chapter in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* (for the release that pertains to the application you are skinning).

## 5.5.1 How to Configure an ADF Skin Property to Apply to a Message

You add a pseudo-class to the component's selector for the message type that you want to configure. You then define style properties for the pseudo-class using the Property Inspector.

**To configure an ADF skin property to apply to a message:**

1. In the Selector Tree of the visual editor, expand the Faces Component Selectors section and select the selector for the ADF Faces component for which you want to configure the style properties to apply to a message.

   For example, select the **af | inputText** selector to configure the style properties to apply to the ADF Faces `inputText` component.

2. Click the **Add Pseudo- Class** icon to display a list of the available pseudo-classes for the selector that you selected in step 1.

3. Select the pseudo-class that corresponds to the message for which you want to configure style properties. The following pseudo-classes are available for the ADF Faces components:

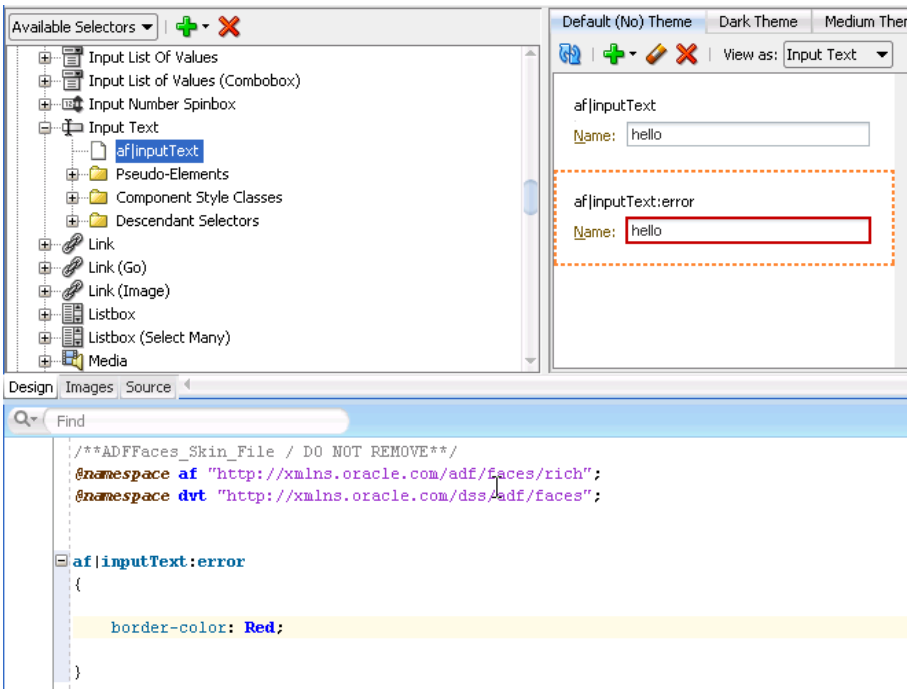
   - `fatal`
   - `error`
   - `warning`
   - `confirmation`
   - `info`

4. Configure the style properties that you want to apply to the component at runtime when the application displays a message with the component.

## 5.5.2 What Happens When You Configure an ADF Skin Property to Apply to a Message

The visual editor writes the values that you specify for the selector's pseudo-class in the Property Inspector to the source file for the ADF skin. For example, assume that you set the value of the Border Color property to `Red` for the af|inputText selector's error pseudo-class. Figure 5–15 shows the change that appears in the source file and in the Preview Pane of the ADF skin.

***Figure 5–15 Style Properties for an inputText Component's Error Message***




If you want to define style properties to appear when the component displays an error message, use the error pseudo class. For example, set the content pseudo element's background color property for the `af:inputText` component's to `Red` when an error occurs as follows:

```
af|inputText::content:error
{
  background-color:Red
}
```

## 5.5.3 What You May Need to Know About Configuring an ADF Skin Property to Apply to a Message

The visual editor does not support the addition or the configuration of the pseudo-classes for message types to a selector's pseudo-element. Use the source editor for the ADF skin if you want to add a pseudo-class for a message type to a selector's pseudo-element. Example 5–2 demonstrates the syntax to write if you want to define red as the background color for the af|inputText selector's content pseudo-element

***Example 5–2 Adding a Message Pseudo-Class to a Pseudo-Element***

```
af|inputText::content:error
{
  background-color:Red
}
```

## 5.6 Applying Themes to ADF Faces Components

Themes are a way of implementing a look and feel at a component level. A theme's purpose is to provides a consistent look and feel across multiple components for a portion of a page. A common usage for themes is in a JSF page template where certain areas have a distinct look. For example, a page may have a branding area at the top with a dark background and light text, a navigation component with a lighter background, and a main content area with a light background.

A component that sets a theme exposes that theme to its child components and therefore the theme is inherited. Themes can be set (started or changed) by the `af:document` and `af:decorativeBox` components.

The Fusion ADF skins support the following themes:

- `dark`
- `medium`
- `light`
- None (`default`)

You can also create your own theme by entering syntax similar to the following in the source file of an ADF skin:

```
af|document[theme=UserCreated] {}
```

Figure 5–16 shows how the visual editor renders tabs where you can configure style properties for each theme provided by the Fusion ADF skins in addition to a user-created theme.

***Figure 5–16 Tabs in the Visual Editor for Themes***

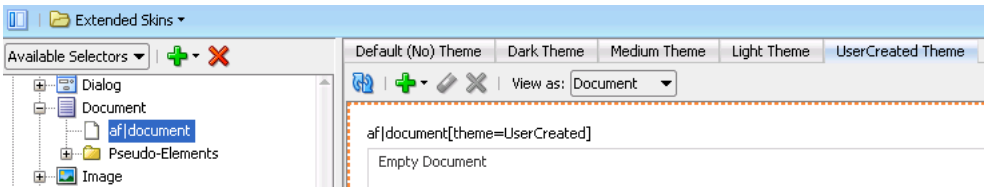
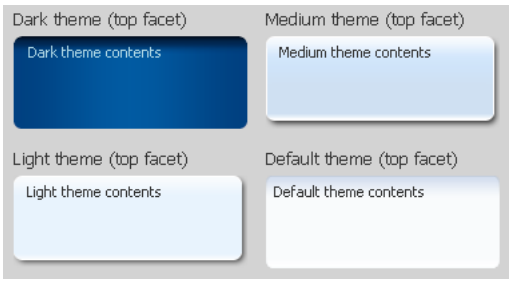


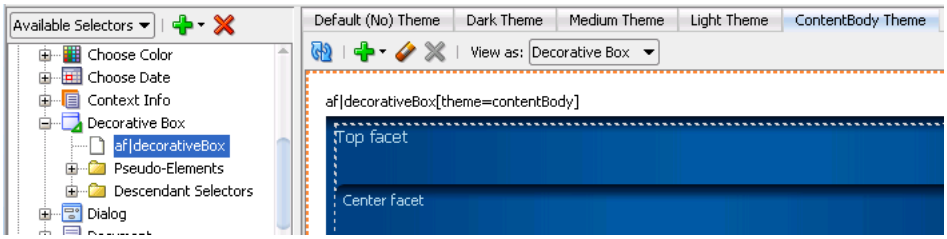

Figure 5–17 shows how the different themes contrast to each other.

***Figure 5–17 Default Appearance of Themes***




In addition to the themes listed previously, there is one other theme (`contentBody`) that the `af|decorativeBox` selector uses, as shown in Figure 5–18. The `af:decorativeBox` component is the only component that ever renders using the style properties defined for this theme.

***Figure 5–18 Themes for the decorativeBox Selector***




In your application, you start the theme by specifying it as an attribute of the `af:document` component in the JSF page, as shown in the following example:

```
<af:document theme="dark">
  <af:panelTabbed>...</af:panelTabbed>
</af:document>
```

> **Note:** You can also start a theme by specifying it as an attribute of the `af:decorativeBox` component.

You can prevent a component inheriting modifications made to a parent component. For example, you can prevent the `af:panelTabbed` child component inheriting the `dark` theme defined for the `af:document` parent component in the JSF page. For more information, see the Section 5.6.3, "How to Prevent a Component Inheriting a Theme from a Parent Component."

By default, themes are not set for components or their child components. Because themes are inherited, the following values are supported when a component has a `theme` attribute that is not set:

- not given - If no theme is given, the theme is inherited, as in `<af:decorativeBox>...`
- `#{null}`- The theme is inherited; same as not given.
- `inherit` - The theme is inherited; same as null.
- `default` - The theme is removed for the component and its child components.

- empty string - If the theme is set to a blank string, it has the same behavior as `default`. For example, `<af:decorativeBox theme="">` will remove the theme for the component and its child components.

Because the themes are added to every HTML element of a component that supports themes and that has style classes, there is no need for containment-style CSS selectors for themes. With the exception of `:ltr` and `:rtl`, all theme selectors should always appear on the last element of the selector. For example, the selector to apply a dark theme to each step of an `af:breadCrumbs` component is:

```
af|breadCrumbs::step:disabled[theme="dark"] {
  color:#FFFFFF;
}
```

Color incompatibility may occur if a component sets its background color to a color that is not compatible with its encompassing theme color. For example, if a `panelHeader` component is placed in a dark theme, the CSS styles inside the `panelHeader` component will set its component background to a light color without changing its foreground color accordingly. The result is a component with a light foreground on a light background. Many other components also set their foreground color to a light color when placed in a dark theme.

If color incompatibility occurs, you can resolve color incompatibility between parent and child components by setting a value for the `-tr-children-theme` property. For components that do not have a parent-child relationship, you can manually set the component's theme color to a color that is compatible with the surrounding theme color. You do this by inserting the `panelGroupLayout` or `panelStretchLayout` component inside the component and by setting the `panelGroupLayout` or `panelStretchLayout` theme to a compatible color.

```
<af:panelHeader text="Header Level 0">
            <af:panelGroupLayout layout="vertical" theme="default">
              ...
            </af:panelGroupLayout>
</af:panelHeader>
```

## 5.6.1 How to Enable Themes for Components

You enable themes on a per-component basis in an ADF skin. Enabling themes on a per-component basis means that you do not generate unnecessary HTML attributes that the ADF skin will not use.

**To enable themes for components:**

1. In the source editor, enter syntax for the component's selector to enable themes for the component in the ADF skin. For example, to enable theme support in an ADF skin for the `outputLabel` component, enter the following:

```
af|outputLabel {
   -tr-enable-themes: true;
 }
```

## 5.6.2 How to Set Theme Properties for a Component in an ADF Skin

You set theme properties for a component using the tab in the visual editor that corresponds to the theme you want to configure.

**To set theme properties for a component in an ADF skin:**

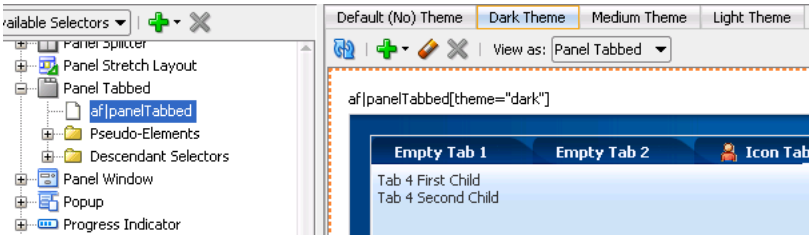
1. In the Selector Tree of the visual editor, expand the appropriate node for which you want to set theme properties.

   You can configure items under the Style Classes, Faces Component Selectors, Data Visualization Component Selectors, and Other nodes.

2. Click the tab that corresponds to the theme for which you want to set properties.

   For example, if you want to set a property for the dark theme, click **Dark Theme**, as shown in Figure 5–19.

***Figure 5–19 Dark Theme in Visual Editor***




3. In the Property Inspector, set values for the properties that you want to configure for the selected theme.

   Example 5–3 shows the entry that appears in the source file of an ADF skin if you set the `background-color` property of the `af:panelTabbed` component's `dark` theme to `Red`.

***Example 5–3 Defining a Theme for a Component in an ADF Skin***

```
af|panelTabbed[theme="dark"] {
    background-color: Red;
}
```

## 5.6.3 How to Prevent a Component Inheriting a Theme from a Parent Component

If you do not want a child component to inherit modifications made to a parent component in a JSF page, set a value for the `-tr-children-theme` property in the source file of the ADF skin. For example, you do not want the `af:panelTabbed` child component to inherit the `dark` theme defined for the `af:document` parent component in the JSF page. Set the `-tr-children-theme` property in the source file for the ADF skin as shown in Example 5–4.

Note that if you do not want a component to inherit modifications for a specific theme, you must specify the themed version. In Example 5–4, this is the `dark` theme. If you wanted to prevent the inheritance of modifications for the `medium` theme, you also set the `-tr-children-theme` property in the source file for the `medium` theme.

***Example 5–4 Child Component Preventing Inheritance of a Theme from a Parent Component***

```
af|panelTabbed[theme="dark"] {
    -tr-children-theme: default;
}
```

## 5.7 Configuring an ADF Skin for Accessibility

Oracle ADF provides application accessibility support to make applications developed using ADF Faces components usable for persons with disabilities. You can define style properties in your ADF skin specifically for persons with disabilities as part of efforts to make your application accessible. You preface these style properties with the `@accessibility-profile` rule.

The `@accessibility-profile` rule allows you to define style properties for the `high-contrast` and `large-fonts` accessibility profile settings that you can specify in the `trinidad-config.xml` file. For more information about the `trinidad-config.xml` file, Section 11.3, "Configuration Files for an ADF Skin."

Define style properties for the `high-contrast` accessibility profile where you want background and foreground colors to contrast highly with each other. Define style properties for the `large-fonts` accessibility profile for cases where the user must be allowed to increase or decrease the text scaling setting in the web browser. Defining large-fonts does not mean that the fonts are large, but rather that they are scalable fonts or dimensions instead of fixed pixel sizes.

Example 5–5 shows style properties that get applied to the `af|column::sort-ascending-icon` pseudo-element when an application renders using the `high-contrast` accessibility profile.

***Example 5–5 Style Properties Defined Using the @accessibility-profile***

```
@accessibility-profile high-contrast {
  af|column::sort-ascending-icon {
     content: url("/afr/fusion/sort_asc_ena.png");
   }
   af|column::sort-ascending-icon:hover {
     content: url("/afr/fusion/sort_asc_ovr.png");
   }
   af|column::sort-ascending-icon:active {
     content: url("/afr/fusion/sort_asc_selected.png");
   }
   af|column::sort-descending-icon {
     content: url("/afr/fusion/sort_des_ena.png");
   }
   af|column::sort-descending-icon:hover {
     content: url("/afr/fusion/sort_des_ovr.png");
   }
   af|column::sort-descending-icon:active {
     content: url("/afr/fusion/sort_des_selected.png");
   }
   af|column::sorted-ascending-icon {
     content: url("/afr/fusion/sort_asc_selected.png");
   }
   af|column::sorted-descending-icon {
     content: url("/afr/fusion/sort_des_selected.png");
   }
```

For more information about developing accessible ADF Faces pages and accessibility profiles, see the "Developing Accessible ADF Faces Pages" chapter in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* (for the release that pertains to the application you are skinning).

### 5.7.1 How to Configure an ADF Skin for Accessibility

You define style properties for the selector or selector´s pseudo-elements that you want to configure and preface these style properties with the `@accessibility-profile` rule.

**To configure an ADF skin for accessibility:**

1. Define style properties for the selectors and selectors´ pseudo-elements that you want to configure, as described in Section 5.4, "Changing a Component-Specific Selector."
2. In the source file for the ADF skin, preface the skinning keys that you configured with the `@accessibility-profile` rule, as illustrated in Example 5–5.

# 6

# Working with Images in Your ADF Skin

This chapter describes how to work with images in an ADF skin. Key features such as how you change an image for a component selector are described in addition to how to work with the Images window that is enabled when your ADF skin extends from the Fusion Simple family of ADF skins.

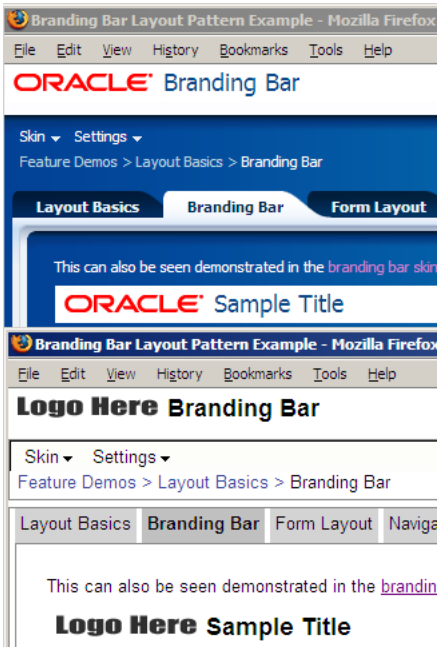
This chapter includes the following sections:



## 6.1 About Working with Images in an ADF Skin

You can reference images in an ADF skin by using a URL to specify the location of the image. You do this when you want to specify an image as a company logo for use in multiple web pages, for example. Other scenarios where you use images include when you want to display an image in conjunction with a warning or error message to capture your end user's attention or you want to render an image to make your application more visually appealing. Figure 6–1 shows an example that illustrates how using images referenced by an ADF skin can change the user interface that an application renders. The page in Figure 6–1 is the same page rendered by the same application using two different ADF skins.

***Figure 6–1 ADF Skin Using an Image***




The ADF skin that the web page in the foreground of Figure 6–1 uses does not reference an image while the ADF skin that the web page in the background renders a company logo. It renders the company logo by defining the location of the image as a value for the `background-image` property in the `.AFBrandingBarLogo` style class.

Apart from defining images as the values for the `background-image` property, there are a range of selectors that reference images. These images appear, for example, as icons in ADF Faces components that render at runtime. For more information, see Section 2.1.2, "ADF Skin Selectors and Icon Images."

## 6.2 Changing an Image for a Component Selector

Many ADF Faces and ADF Data Visualization components reference images using selectors. These images display in the background of the component or render as icons or controls on the component. When you create an ADF skin, the ADF skin that you extend from provides the values for these selectors such as, for example, the relative path to an image, sizes for height, width, and so on.

Figure 6–2 shows a runtime view of the ADF Faces `table` component rendering a control that sorts the data in a table column in ascending order. The image that renders this control is referenced by the ADF Faces `column` component's `sort-ascending-icon-style` selector.

***Figure 6–2 Image Referenced by the sort-ascending-icon-style Selector***

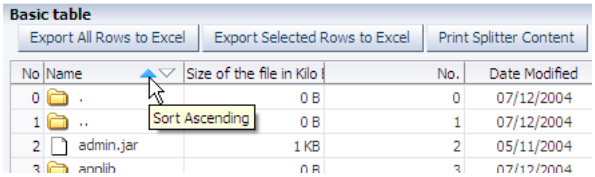
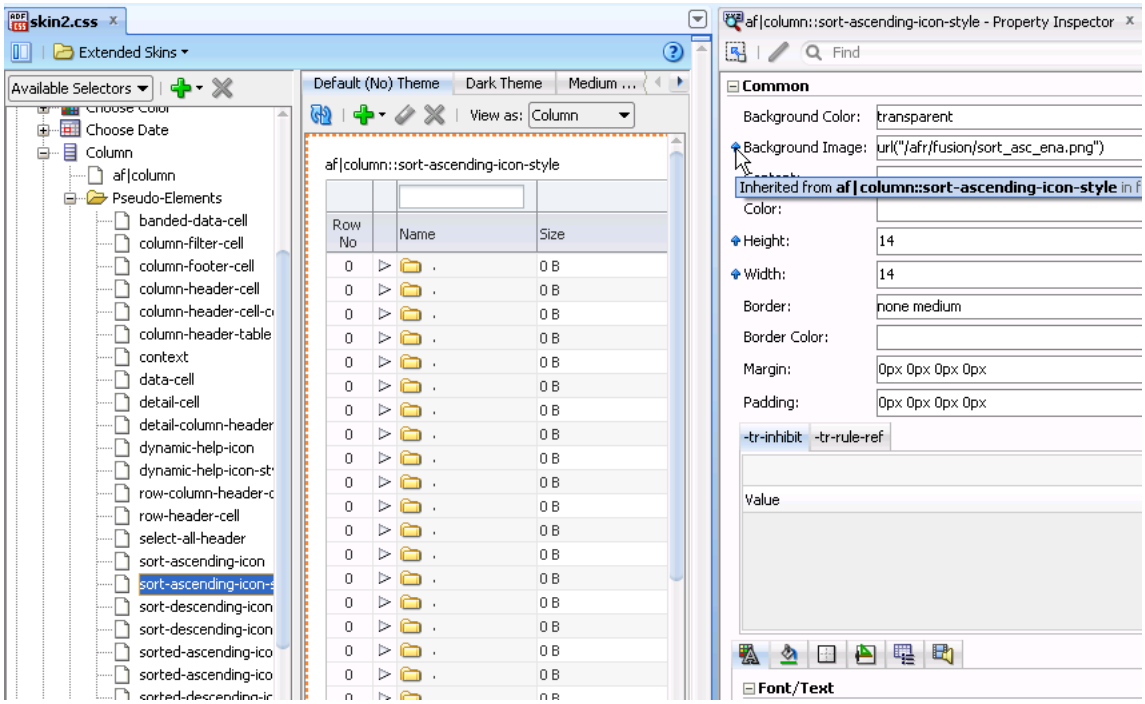



Figure 6–3 shows a design-time view where an ADF skin inherits values for the ADF Faces `column` component's `sort-ascending-icon-style` selector from the

extended ADF skin. The values inherited include the file name for the image used as an icon (`sort_asc_ena.png`), the height, and the width for the image.

***Figure 6–3 Inherited Values for the sort-ascending-icon-style Selector***




Other examples of ADF Faces and ADF Data Visualization components that expose selectors which reference images associated with the component include the following:

- ADF Faces `progressIndicator` component exposes the `determinate-empty-icon-style` selector.
- ADF Faces `panelAccordion` component exposes the `disclosed-icon-style` selector.
- ADF Data Visualization `mapToolbar` component exposes the `zoomin-enable-icon` selector.

If you decide that you want to modify the image that is associated with a component selector, you need to modify the selector in your ADF skin and copy the image into the project for your ADF skin. You can copy images individually using the procedure in Section 6.2.1, "How to Copy an Image into the Project" or you can import multiple images using the Images window, as described in Section 6.3, "Working with the Images Window."

After you import an image into your project, the selector that references the image uses a URL in the source file of the ADF skin to refer to this image. Note that this URL is updated when you deploy your ADF skin (and associated files) in an ADF Library JAR, as described in Section 10.3, "Packaging an ADF Skin into an ADF Library JAR."

> **Tip:** Associate an image with a global selector alias. If multiple component selectors reference the global selector alias, you only need to make one change if you want to use a different image at a later time (change the image associated with the global selector alias). For more information about global selector aliases, see Section 8.2, "Creating a Global Selector Alias."

## 6.2.1 How to Copy an Image into the Project

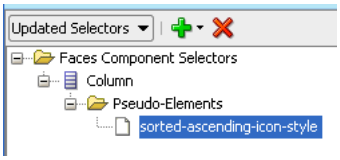
You use a context menu to copy an image that an extended ADF skin references into a directory of the project for your ADF skin. You then make the changes that you want to the image.

**To copy an ADF skin image into your project:**

1. In the Selector Tree of the visual editor, select the selector that references the image you want to change.

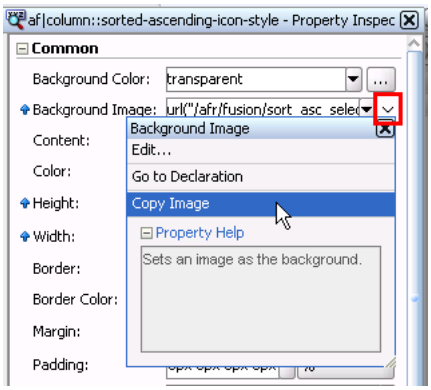
   For example, select the ADF Faces `column` component's **sort-ascending-icon-style** selector to change the sort ascending icon, as shown in Figure 6.2.

***Figure 6–4 Column Component's sort-ascending-icon-style Selector***




2. In the Property Inspector, expand the **Common** section and select **Copy Image** from the Background Image list, as shown in Figure 6–5.

***Figure 6–5 Copy Image Menu to Import an Image into ADF Skin Project***




   This copies the image into the project for your ADF skin.

## 6.2.2 What Happens When You Copy an Image into the Project

The image is copied into a subdirectory that is generated in the project of your ADF skin. For example, if you decided to copy the image that the ADF Faces `column` component's `sort-ascending-icon-style` selector references, the `sort_asc_ena.png` file is copied to the following directory:

`/public_html/skins/skin1/images/af_column`

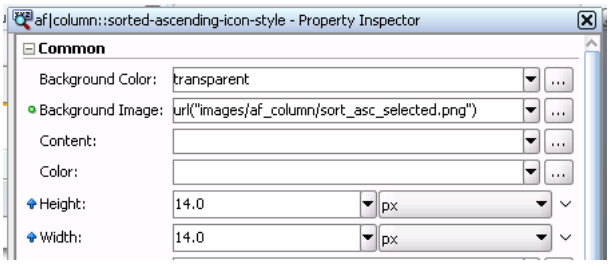
where *`af_column`* refers to the ADF Faces `column` component.

The relative URL value of the property in the Property Inspector is modified to reference the new location of the image. Figure 6–6 shows an example.

In addition, the Property Inspector indicates that the selector no longer inherits the image from the extended ADF skin by displaying a green icon to the left of the property label. Figure 6–6 shows the Property Inspector after importing the `sort_`

`asc_ena.png` file into the project for the ADF skin. Note that the ADF skin still inherits the values for the Height and Width properties from the extended ADF skin.

***Figure 6–6 Property Inspector After Importing an Image into an ADF Skin***




Finally, CSS syntax appears in the source file of your ADF skin. Example 6–1 shows the CSS syntax that corresponds to the values shown in Figure 6–6.

***Example 6–1 CSS Syntax in Source File of ADF Skin After Importing an Image***

```
af|column::sorted-ascending-icon-style
{
  background-image: url("images/af_column/sort_asc_selected.png");
}
```

## 6.3 Working with the Images Window

The Images window helps you manage the images that you want to use with an ADF skin that extends from the Fusion Simple family of ADF skins. You access it by clicking the **Images** tab of an open ADF skin.

> **Note:** Your ADF skin must extend the Fusion Simple family of ADF skins if you want to use the functionality in the Images window. You cannot use the Images window if you extend ADF skins from other skin families.

Figure 6–7 shows the Images window that appears when you first click the Images tab in your ADF skin. The Generate Images Using list displays the following options:

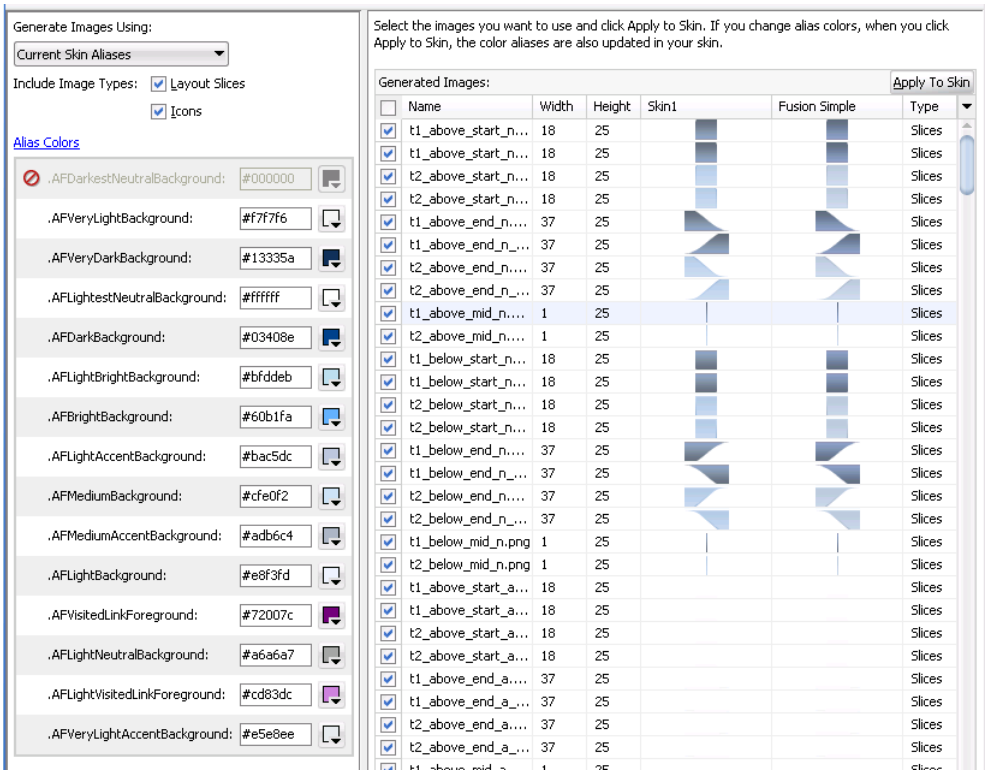
- **Current Skin Aliases**: Select to start with a colorized version using the global selector aliases that appear in the Color category of the current ADF skin. Choosing this option displays the Alias Color list where you can modify the values of these global selector aliases.
- **Desaturated Fusion Simple Colors**: Select to start with a desaturated version of the set of images for the Fusion Simple skin.
- **Fusion Simple Colors**: Select to start with a set of images for the Fusion Simple skin.

> **Tip:** Selecting **Desaturated Fusion Simple Colors** from the Generate Images Using list and clicking **Apply to Skin** is a useful method to retrieve all the current images if you want to modify them manually in another tool.

The Generated Images list displays the available images that you can apply to your ADF skin by clicking the **Apply to Skin** button. When you click the **Apply to Skin**

button, the selected images in the Generated Images list are imported into an images directory that is a subdirectory of the directory in your project where you store your ADF skin.

***Figure 6–7 Images Window for an ADF Skin***




The Alias Colors list that appears when you select **Current Skin Aliases** in the Generate Images Using list displays the color aliases that impact the color of layout and icon images. These color aliases are a subset of the available color aliases. Changing the color aliases in this subset can have a significant impact on the appearance of your application. Figure 6–8 shows a page from an application where the parts of a page that use these color aliases are labeled. For example, **Bookmarkable Link** uses the `.AFLightVisitedLinkForeground` color alias after a user clicks the link.

***Figure 6–8 Application Page Using Color Aliases***

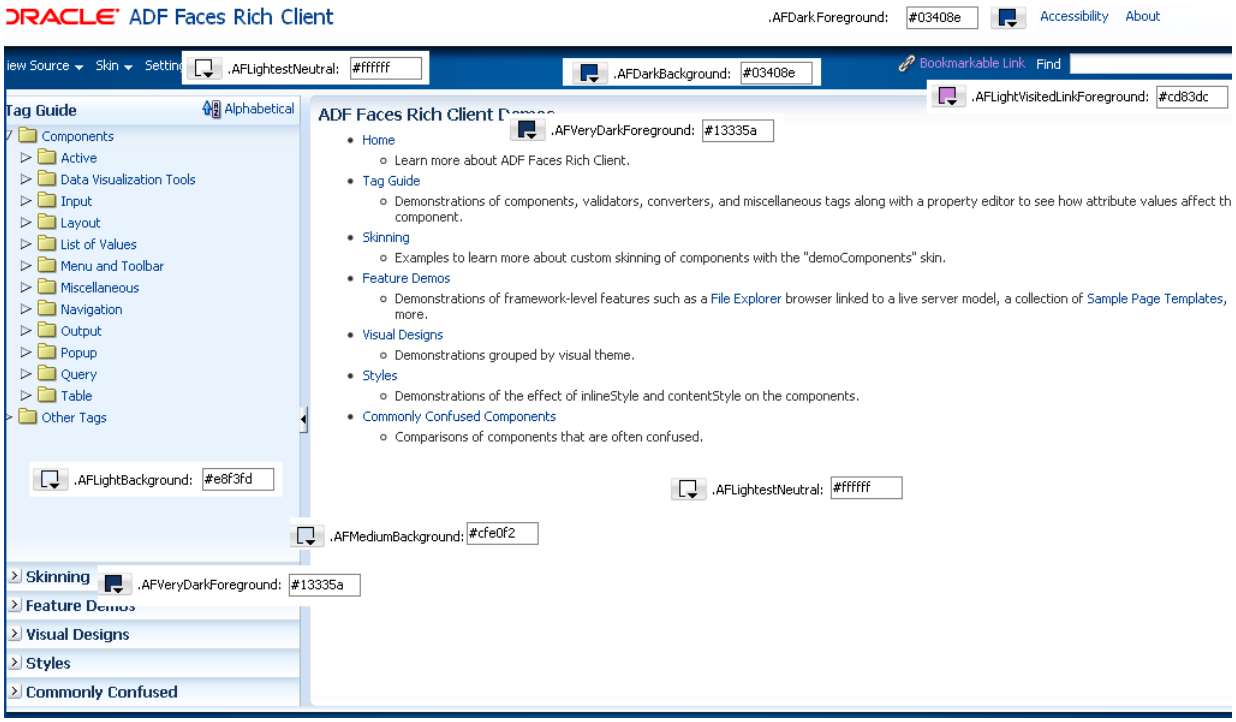
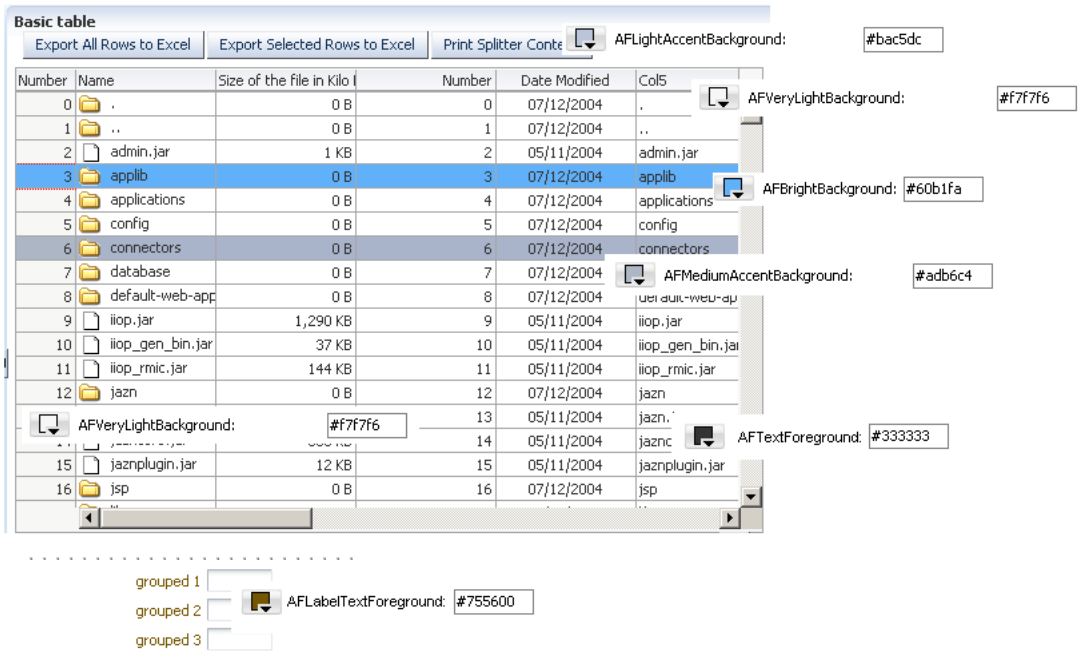



Figure 6–9 shows another example where the usage of color aliases is labeled.

***Figure 6–9 ADF Faces Table Component Using Color Aliases***




For more information about the Color category of global selector aliases, see Section 8.1, "About Global Selector Aliases."

The Oracle Technology Network (OTN) web site provides an online demonstration that shows you how to change the color aliases in the Color Alias list as part of the

process of developing an ADF skin. For more information, navigate to http://www.oracle.com/technetwork/developer-tools/adf/overview/index.html.

### 6.3.1 How to Generate Images Using the Images Window

You generate images using the Images window by choosing one of the supported methods and using it to apply changes to your ADF skin.

**To generate images using the Images window:**

1. Create an ADF skin that extends the Fusion Simple family of ADF skins.

   For more information about creating an ADF skin, see Section 4.4, "Creating an ADF Skin File." For more information about the Fusion Simple family of ADF skins, see Section 11.4, "ADF Skins Provided by Oracle ADF."

2. Click the **Images** tab for the newly-created ADF skin.

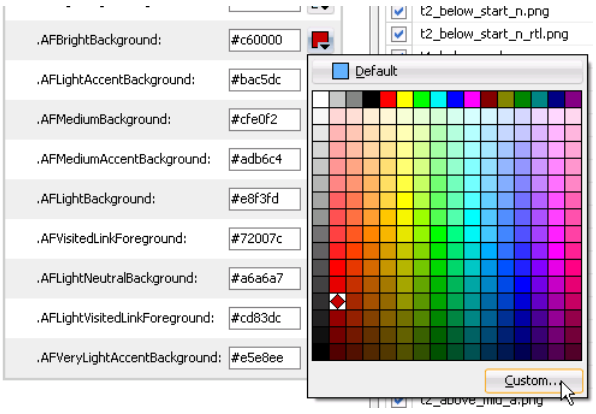
3. Choose the method that you want to use to generate the images from the Generate Images Using list.

4. Choose the appropriate option for the image types that you want to include:

   - **Layout Slices**: select this checkbox to include this type of image in your ADF skin.
   - **Icons**: select this checkbox to include this type of image in your ADF skin.

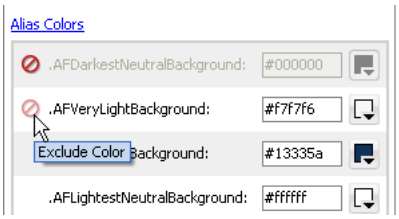
5. (Optional) If you selected **Current Skin Aliases** from the Generate Images Using list, modify the values for the entries in the Alias Color list by entering a Hex code directly in the input field for the global selector alias that you want to modify. Alternatively, invoke a color picker by clicking the dropdown list beside the input field. You can also invoke the Select Custom Color dialog by clicking the **Custom** button in the color picker or reset the value of the global selector alias using the **Default** button. Figure 6–10 shows these buttons and the dropdown list that initially displays the buttons.

***Figure 6–10 Color Picker for Color Aliases***




6. (Optional) If you selected **Current Skin Aliases** from the Generate Images Using list, you can click the Exclude Color icon to inhibit the usage of a color alias when you generate images. The Exclude Color icon appears when you move your mouse over a color alias, as shown in Figure 6–11.

*Figure 6–11 Exclude Color Icon for Color Aliases*




7. In the Generated Images list, select the images that you want to apply to the ADF skin. Use the checkboxes on the Generated Images list to select or deselect all the images or to select one or more images. By default, the selected images are those that have been modified as a result of changes to the color aliases.

> **Note:** Scroll to the bottom of the Generated Images list to verify that all the images that you want to apply to the skin are selected.

8. Click **Apply to Skin**.

## 6.3.2 What Happens When You Generate Images Using the Images Window

The image files that you selected in the Generated Images list are imported into the project. Entries appear for each image that you generate in the source file of the ADF skin. Entries also appear for each global selector alias that you modify in the Alias Colors list if you chose to generate the images using the **Current Skin Aliases** option. Example 6–2 shows some entries that appear in the source file of an ADF skin where images were generated using the Current Skin Aliases option with values modified for the `AFDarkestNeutralBackground` and `AFVeryLightBackground` global selector aliases.

*Example 6–2 Entries in the Source File of an ADF Skin after Generating Images*

```
af|panelWindow::footer-end
{
  background-image: url(images/generated/adf/images/fusion/dialog-footer-small-right.png);
}
...

af|train::stop-icon-unvisited:hover, af|train-vertical::stop-icon-unvisited:hover
{
  background-image: url(images/generated/adf/images/fusion/train_unvisited_ovr.png);
}
...

.AFDarkestNeutralBackground:alias
{
  background-color: #00ff00;
}
.AFVeryLightBackground:alias
{
  background-color: #00ff00;
}
```

# 7

# Working With Text in an ADF Skin

This chapter describes how to work with text in an ADF skin. Key concepts such as how the resource strings that ADF Faces components render at runtime are stored in resource bundles are described in addition to how you can specify additional resource bundles with different resource strings.

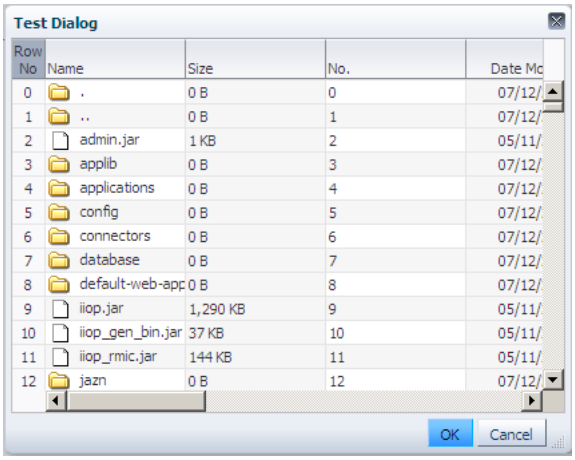
This chapter includes the following sections:



## 7.1 About Working with Text in an ADF Skin

The source file for an ADF skin does not store any text that ADF Faces components render in the user interface of your application. Applications that render ADF Faces components abstract the text that these components render as resource strings and store the resource strings in resource bundles. For example, Figure 7–1 shows an ADF Faces `dialog` component that renders command buttons with **OK** and **Cancel** labels.

***Figure 7–1 ADF Faces dialog Component***




The text that appears as the labels for these command buttons (**OK** and **Cancel**) is defined in a resource bundle and referenced by a resource string. If you want to change the text that appears in the command button labels, you create a resource bundle where you define the values that you want to appear by specifying alternative text for the following resource strings:

- `af_dialog.LABEL_OK`

- `af_dialog.LABEL_CANCEL`

> **Note:** By default, a resource bundle (`skinBundle.properties`) is created in your project when you create a new ADF skin, as described in Section 4.4, "Creating an ADF Skin File."

In addition to the resource strings that define the text to appear in the user interface for specific components, there are a range of resource strings that define text to appear that is not specific to any particular component. These resource strings are referred to as *global resource strings*. For more information about the resource strings for ADF Faces components and global resource strings, see the *Oracle Fusion Middleware Tag Reference for Oracle ADF Faces Skin Selectors* (for the release that pertains to the application you are skinning).

ADF Faces components provide automatic translation. The resource bundle used for the ADF Faces components' skin is translated into 28 languages. If, for example, an end user sets the browser to use the German (Germany) language, any text contained within the components automatically displays in German. For this reason, if you create a resource bundle for a new ADF skin, you must also create localized versions of that resource bundle for any other languages your web application supports.

For more information about creating resource bundles, resource strings, and localizing ADF Faces components, see the "Internationalizing and Localizing Pages" chapter in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* (for the release that pertains to the application you are skinning).

## 7.2 Using Text From Your Own Resource Bundle

If you enter alternative text in a resource bundle to override the default text values that render in the user interface of the ADF Faces components in your application, you need to specify this resource bundle for your application. At runtime, the application renders the alternative text in your resource bundle for the resource strings that you override. For resource strings that you do not override, the application renders the text defined in the base resource bundle. For example, Figure 7–4 shows an ADF Faces dialog component where the application developer overrides the default value for the `af_dialog.LABEL_OK` resource string from `OK` to `Yay` while the default value for the `af_dialog.LABEL_CANCEL` resource string remains unchanged. That is, the application developer did not define a value for the `af_dialog.LABEL_CANCEL` resource string in a resource bundle; the application references the base resource bundle for this resource string's value.

***Figure 7–2 Overridden and Default Values Resource Strings***


For more information about how to create a resource bundle and how to define string key values, see the "Internationalizing and Localizing Pages" chapter in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* (for the release that pertains to the application you are skinning).

## 7.2.1 How to Specify an Additional Resource Bundle for an ADF Skin

You specify a resource bundle for your ADF skin by adding its name and location as a value to the `bundle-name` property in the `trinidad-skins.xml` file.

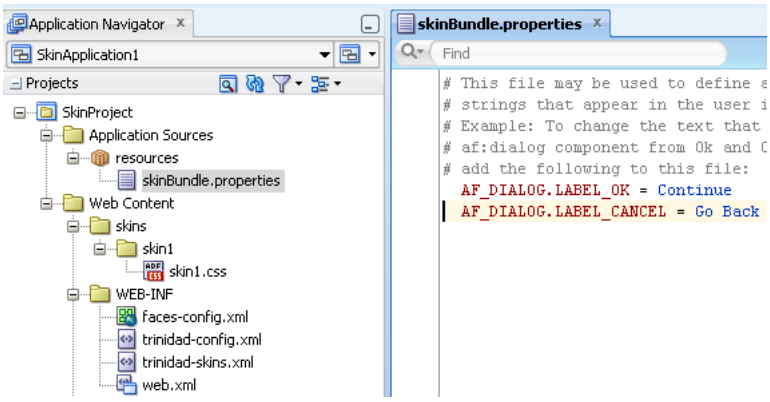
**To specify an additional resource bundle for an ADF skin:**

1. In the Application Navigator, double-click the `trinidad-skins.xml` file for your application. By default, this is under the **Web Content/WEB-INF** node.
2. In the Structure window, right-click the skin node for which you want to add an additional resource bundle and choose **Insert inside skin** > **bundle-name**.
3. In the Property Inspector, specify the name and location for your resource bundle as a value for the `bundle-name` property.

   For example, the resource bundle that is created by default after you create the first ADF skin in your project, as illustrated in Figure 7–3, specifies the following value for the `<bundle-name>` element:

   ```
   <bundle-name>resources.skinBundle</bundle-name>
   ```

***Figure 7–3 Default Resource Bundle for an ADF Skin***




## 7.2.2 What Happens When You Specify an Additional Resource Bundle for an ADF Skin

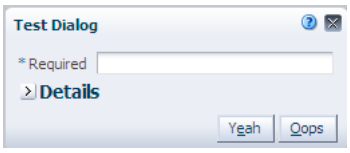
The `trinidad-skins.xml` file references the resource bundle that you specified as a value for the `bundle-name` property, as shown in Example 7–1.

***Example 7–1 Specifying an Additional Resource Bundle in trinidad-skins.xml***

```
<skin>
    <id>skin1.desktop</id>
    <family>skin1</family>
    <extends>fusionFx-simple-v1.desktop</extends>
    <render-kit-id>org.apache.myfaces.trinidad.desktop</render-kit-id>
    <style-sheet-name>skins/skin1/skin1.css</style-sheet-name>
    <bundle-name>resources.skinBundle</bundle-name>
</skin>
```

At runtime, the application renders text values that you specified in your resource bundle to override the default text values. For example, assume that you defined a resource bundle where you specified `Yeah` as the value for the `af_dialog.LABEL_OK` resource sting and `Oops` as the value for the `af_dialog.LABEL_CANCEL`. Example 7–4 shows a `dialog` component that renders labels using these values.

***Figure 7–4 Dialog Rendering Labels Defined in a Custom Resource Bundle***



# 8

# Working With Global Selector Aliases

This chapter describes how to work with global selector aliases. Information on how to create, modify, and apply a global selector alias is provided in addition to describing how to reference a property value from another selector.

This chapter includes the following sections:



## 8.1 About Global Selector Aliases

A *global selector alias* defines style properties in one location in the ADF skin that you can apply to multiple ADF Faces and ADF Data Visualization components. A global selector alias may also be referred to as a *selector alias*, or simply a *selector*. The ADF skins provided by Oracle ADF, described in Section 1.5, "Inheritance Relationship of the ADF Skins Provided by Oracle ADF" and Section 11.4, "ADF Skins Provided by Oracle ADF" make extensive use of global selector aliases to define common style properties for text, messages, icons, colors and different groups of components. Many component-specific selectors inherit the styles defined for these global selector aliases. For example, the `.AFDefaultFontFamily:alias` global selector alias defines a default font family for all ADF Faces components in your application that display text. Any ADF skin that you create by extending from one of the ADF skins provided by Oracle ADF inherits the properties defined in the `.AFDefaultFontFamily:alias` global selector alias. Figure 8–1 shows how the visual editor displays that the `af|commandButton` selector inherits the value for font family from the `.AFDefaultFontFamily:alias` global selector alias.

***Figure 8–1 Component Selector Inheriting Value from Global Selector Alias***

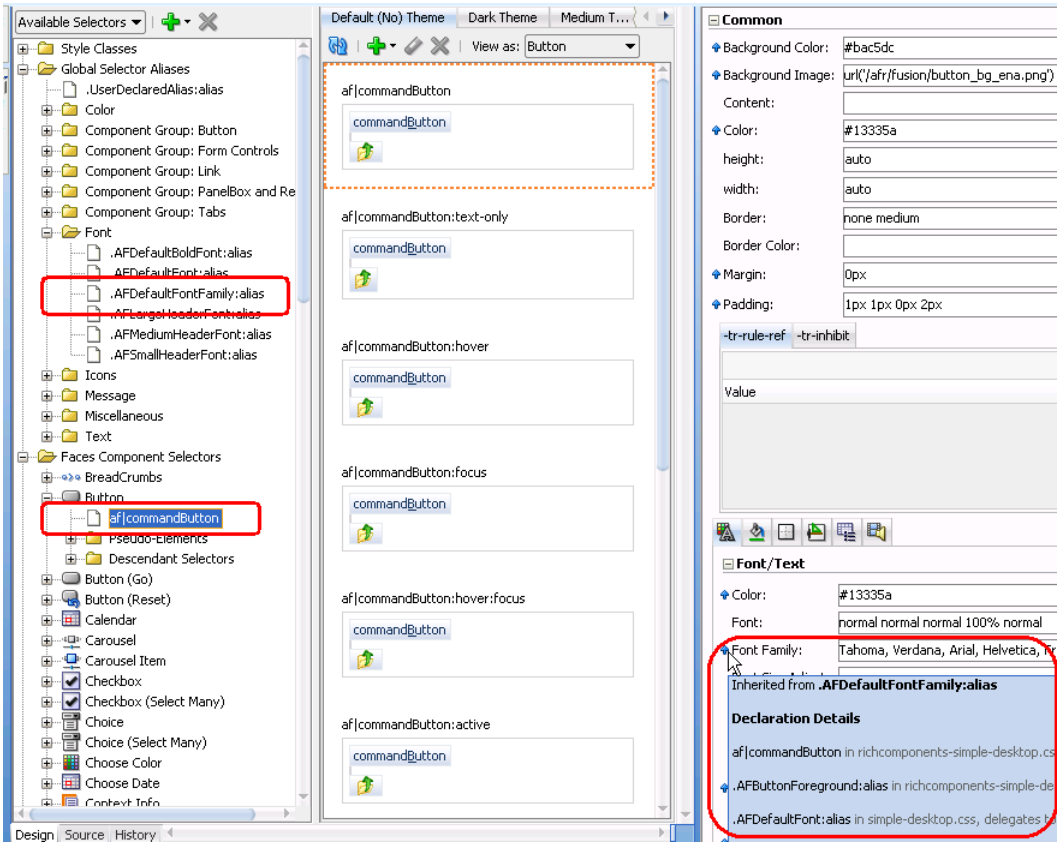



Figure 8–1 also shows the different categories of global selector aliases. Each category groups global selector aliases according to their purpose:

- **Color**: Defines colors used by the ADF skins provided by Oracle ADF. Many global selector aliases that you may want to override appear in this category because they determine most of the colors that appear in a Fusion web application. Changes that you make to these global selector aliases have the most effect if you extend the Fusion Simple family of ADF skins described in Section 11.4, "ADF Skins Provided by Oracle ADF." As with other global selector aliases, you can view which component-specific selectors inherit the values defined in a specific global selector using the **View as** list.
- **Component Group: Button**: Defines style properties inherited by selectors for many of the ADF Faces components that render buttons. For example, the `.AFButtonAccessKeyStyle:alias` global selector alias defines style properties for the access key rendered by the ADF Faces button and dialog components among others.
- **Component Group: Form Controls**: Defines style properties for form controls.
- **Component Group: Link**: Defines style properties for many of the link components.
- **Component Group: PanelBox and Region**: Defines style properties for the `panelBox` and `region` components.
- **Component Group: Tabs**: Defines style properties for many of the ADF Faces components that render tabs. For example, the `.AFFormAccessKeyStyle:alias` global selector alias defines the style

properties for access keys that render in the ADF Faces `panelTabbed` and `navigationPane` components.

- **Font**: Defines style properties for fonts. For example, the `.AFDefaultFontFamily:alias` global selector alias defines the style properties inherited by many of the ADF Faces component selectors.
- **Icons:** Defines the style properties that apply to icons that render in multiple components.
- **Message:** Defines style properties for messages that ADF Faces input components display when they render different types of messages. For more information, see Section 5.5, "Configuring ADF Skin Properties to Apply to Messages."
- **Miscellaneous:** Defines global selector aliases that do not fit in the other categories. For example, the `.AFDynamicHelpIconStyle:alias` global selector alias defines the style to use for the dynamic help icon.
- **Text:** Defines style properties to use for text.

For detailed descriptions of the global selector aliases, see the *Oracle Fusion Middleware Tag Reference for Oracle ADF Faces Skin Selectors* (for the release that pertains to the application you are skinning). Global selector aliases that you define appear under the Global Selector Aliases node, as shown by the entry for the `.UserDefined:alias` in Figure 8–1.

The **View as** list displays the list of components that reference a global selector alias when you select a global selector alias in the Selector Tree. In Figure 8–2, the user selected Panel Window from the list because the `panelWindow` component references the global selector alias.

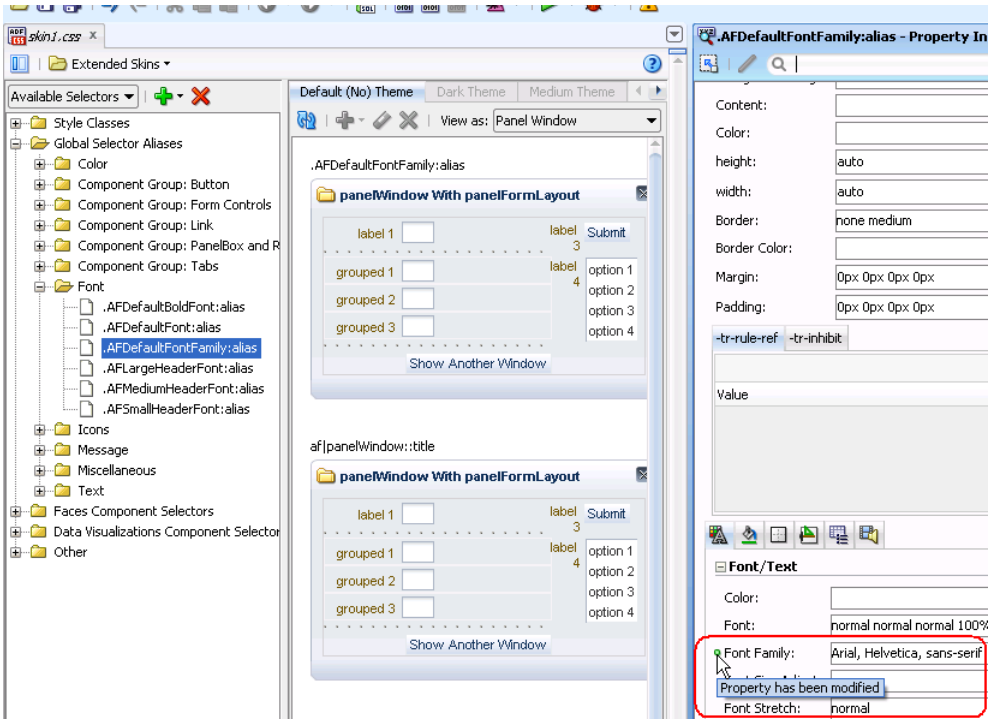
---

**Note:** Sometimes components appear in the **View as** list for which the style properties defined in the global selector alias do not render in the component. This may be because the component initially referenced the global selector alias in an extended ADF skin and your ADF skin overrides the global selector alias for that component. Alternatively, it may be because the component itself overrides the global selector alias using one of its style-related attributes (`styleClass` or `inlineStyle`).

---

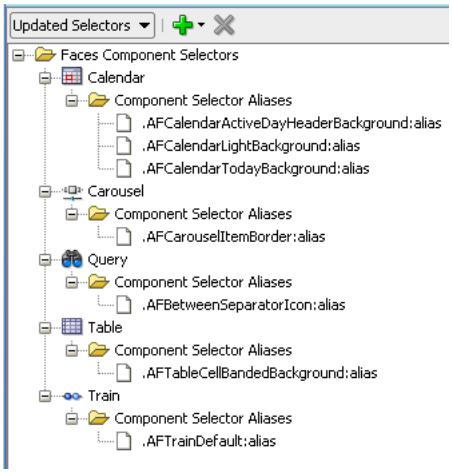
In Figure 8–2, the user has changed the inherited value for the `.AFDefaultFontFamily:alias` global selector alias and viewed the resulting change as it applies to the `panelWindow` component. All selectors that inherit the value of the `.AFDefaultFontFamily:alias` global selector alias will render at runtime using the font family defined in the ADF skin. For example, both the `dialog` and `panelWindow` components render using this font family.

***Figure 8–2 ADF Skin Changing a Global Selector Alias***




In addition to the global selector aliases already described, a number of component selectors define selector aliases that are specific to these components only. These selector aliases appear under the nodes for the component selectors in the Selector Tree. Figure 8–3 shows examples from a number of the component selectors that expose these types of selector aliases.

***Figure 8–3 Component Selector Aliases***




## 8.2 Creating a Global Selector Alias

You can create a global selector alias to define in one location the style properties that you want a number of selectors to reference. You enter the name of the new global selector alias in the Create Alias Selector dialog. The ADF Skin Editor appends the keyword `:alias` and prepends `.` to the name that you enter in the dialog. For

example, if you enter `myGlobalSelector` as the name in the dialog, the resulting name that appears in the user interface and in the source file of the ADF skin is:

```
.myGlobalSelector:alias
```

The keyword `:alias` identifies your global selector alias as a CSS pseudo-class and serves as a syntax aid to organize the CSS code in the source file of your ADF skin.

After you create a global selector alias, you modify it to define the style properties that you want it to contain. For more information, see Section 8.3, "Modifying a Global Selector Alias."

## 8.2.1 How to Create a Global Selector Alias

You can create a global selector alias that defines the style properties that you want a number of user interface components to use.

**To create a global selector alias:**

1. In the Selector Tree of the visual editor, select **New Alias Selector** from the Plus icon's list, as illustrated in Figure 8–4.

***Figure 8–4 New Alias Selector Option in the Selector Tree***


   The Create Alias Selector dialog opens.

2. Enter a name for the global selector alias in the Alias Selector Name field.

   **Tip:** Enter a name for the global selector alias that indicates the purpose it serves. For example, `MyLinkHoverColor` for a global selector alias that is to change the color of a link when an end user hovers over the link.

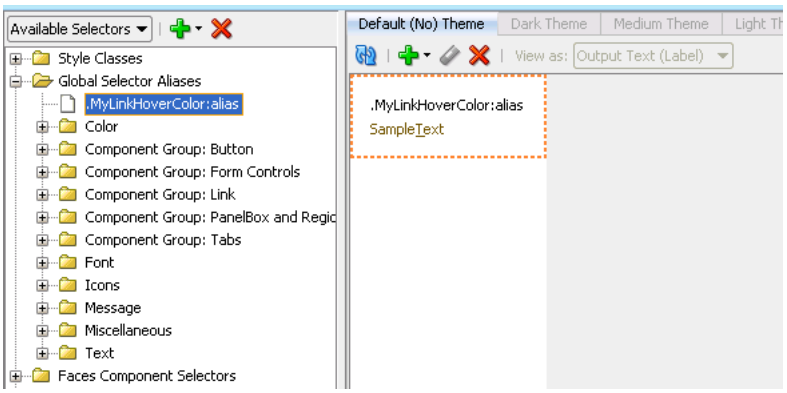
3. Click **OK**.

4. In the Property Inspector, set values for the properties that you want to configure in the global selector alias.

## 8.2.2 What Happens When You Create a Global Selector Alias

The global selector alias appears under the Global Selector Aliases node in the Selector Tree and a visual representation as it applies to a component appears in the Preview Pane, as illustrated in Figure 8–5.

***Figure 8–5 Newly-Created Global Selector Alias***




CSS syntax for the global selector alias that you create appears in the source file of the ADF skin. Figure 8–1 shows the entries that appear in the source file of the ADF skin in Figure 8–5.

***Example 8–1 CSS Syntax for a Newly-Created Global Selector Alias***

```
.MyLinkHoverColor:alias
{
}
```

## 8.3 Modifying a Global Selector Alias

You can modify any of the categories of global selector alias described in Section 8.1, "About Global Selector Aliases." Modifying a global selector alias that appears under the Global Selector Aliases node in the Selector Tree when you first create the ADF skin means that you override the inherited style properties defined in the parent ADF skin of your ADF skin. The parent ADF skin is the ADF skin from which your ADF skin extends. You chose the ADF skin from which to extend when you created an ADF skin, as described in Section 4.4, "Creating an ADF Skin File." After you modify a global selector alias, the component-specific selectors that inherit the style properties defined in the global selector alias use the modified values.

Modifying a global selector alias that you create in your ADF skin does not override any style properties inherited from the parent ADF skin.

### 8.3.1 How to Modify a Global Selector Alias

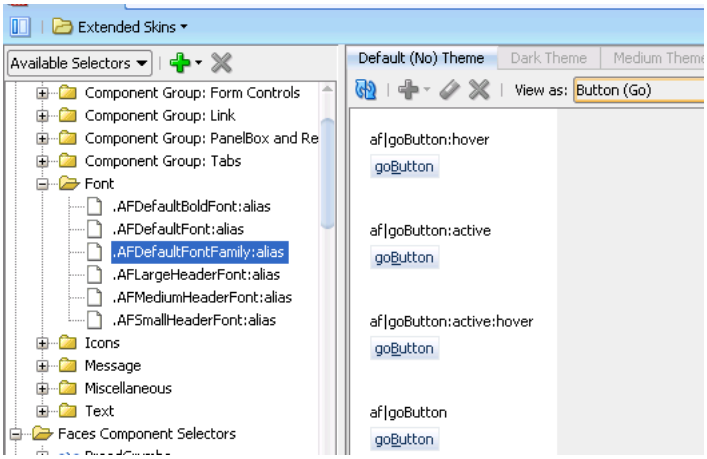
You modify a global selector alias by setting values for it in the Property Inspector. You then verify that the changes you make apply to the component-specific selectors as you intend.

**To modify a global selector alias:**

1. In the Selector Tree of the visual editor, select the global selector alias that you want to modify.

   For example, if you want to modify the global selector alias that defines the default font family, select it as illustrated in Figure 8–6.

*Figure 8–6 Modifying a Global Selector Alias*




2. In the Property Inspector, set values for the properties that you want to modify.
3. In the visual editor, click the **View as** list to select a component-specific selector that inherits the property values defined in the global selector alias that you have just modified.
4. In the visual editor, verify that the changes render for the component-specific selector as you intend. Repeat steps 1 to 3 until you achieve the changes you want for the component-specific selectors that inherit from the global selector alias.

## 8.4 Applying a Global Selector Alias

After you create a global selector alias in your ADF skin, you need to specify the ADF Faces and ADF Data Visualization components that you want to render at runtime using the style properties that you defined in the global selector alias.

Applying a global selector alias to an ADF Faces or ADF Data Visualization component requires you to:

- Select the selector, pseudo-element, or pseudo-class for each component that you want to apply the style properties defined in the global selector alias. If you want to apply the style properties defined in your global selector alias to another global selector alias, select the target global selector alias.
- Set the global selector alias as a value for the ADF skin `-tr-rule-ref-` property

### 8.4.1 How to Apply a Global Selector Alias

You apply a global selector alias by specifying it as a value for the ADF skin's `-tr-rule-ref-` property.

**To apply a global selector alias:**

1. In the Selector Tree of the visual editor, select the item to which you want to apply the global selector alias.

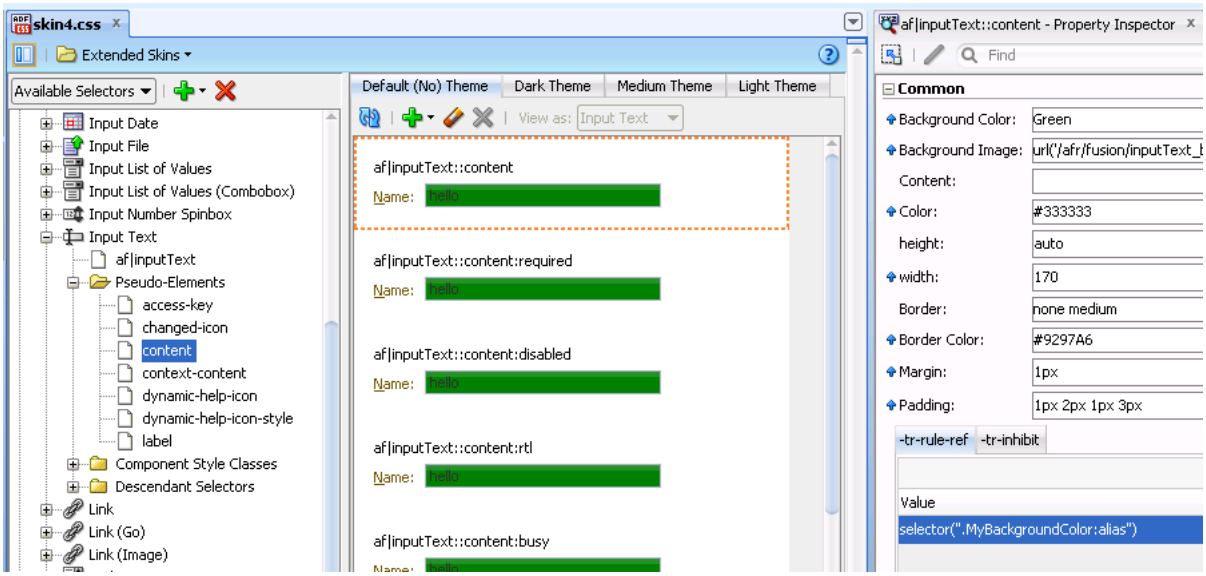
   For example, select the `inputText` component's **content** pseudo-element if you want to apply the style properties defined in your global selector alias to the label for that component, as shown in Figure 8–7.

2. In the Property Inspector, expand the **Common** section and then click the Add icon next to the **-tr-rule-ref-** field.

3. Enter the name of the global selector alias. Enter the name between quotes that you preface with the `selector` keyword in the **Value** field.

   For example, if the name of the global selector alias is `.MyBackgroundColor:alias`, enter `selector(".MyBackgroundColor:alias")`, as illustrated in Figure 8–7.

***Figure 8–7 inputText Component's content Pseudo-Element***



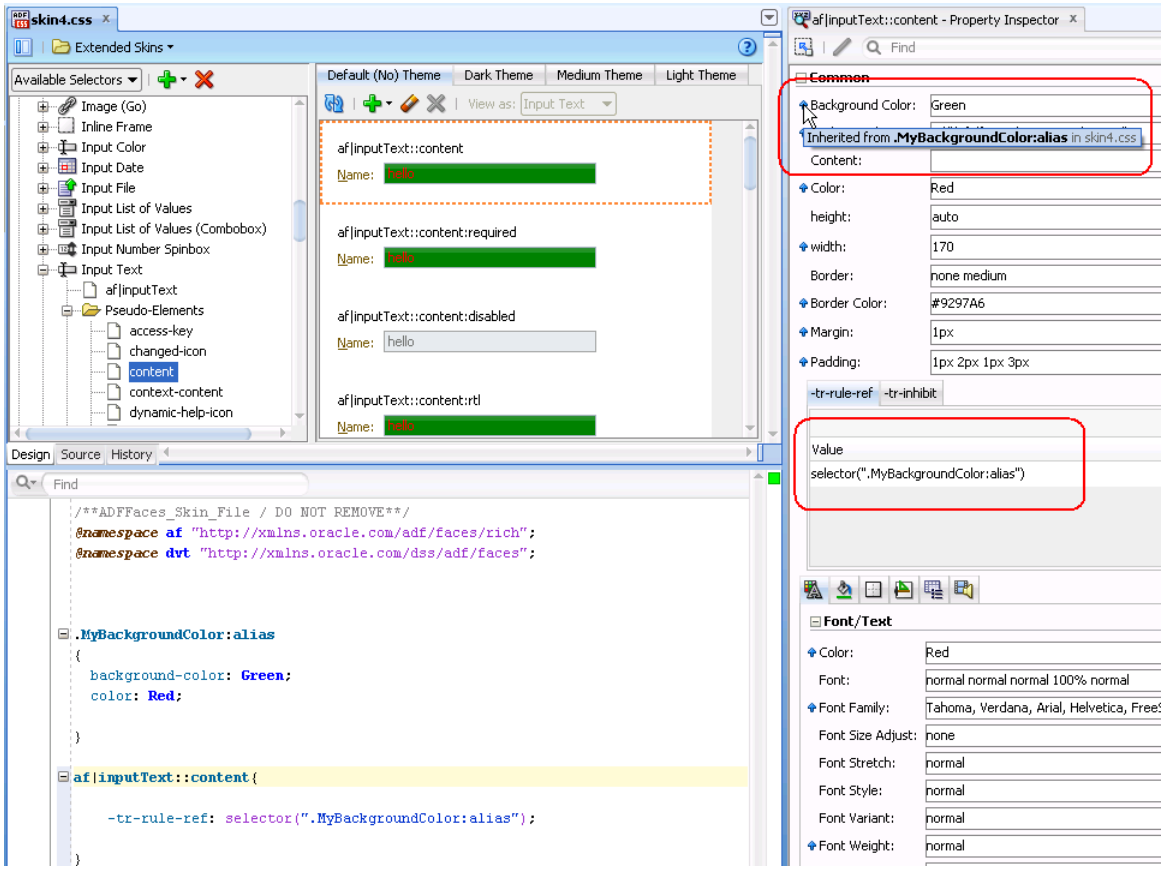

4. Click the Refresh icon in the Preview Pane to view the changes.

## 8.4.2 What Happens When You Apply a Global Selector Alias

The selector to which you applied the global selector alias inherits the style properties defined in the global selector alias. Figure 8–8 shows the `content` pseudo-element for the `inputText` component's selector that inherits the style properties defined in the `.MyBackgroundColor:alias` global selector alias. The properties that inherit their values from a global selector alias when you specify the global selector alias as a value for the ADF skin **-tr-rule-ref** property update to use the inheritance icon, as shown for the **Background Color** and **Color** fields in Figure 8–8.

At runtime, the `inputText` component's content area renders using the style properties defined in the global selector alias.

***Figure 8–8 Global Selector Alias Applied to inputText Component***




## 8.4.3 What You May Need to Know About Applying a Global Selector Alias

If you override a global selector alias in an extended ADF skin, component selectors that used the `-tr-rule-ref` ADF skin property to determine the value of a style property in the parent ADF skin use the overridden value of the global selector alias. Example 8–2 shows ADF Skin B that extends ADF Skin A. At runtime, the top of a `decorativeBox` component renders red for the `background-color` CSS property because the global selector alias in ADF Skin B overrides ADF Skin A.

***Example 8–2 Overriding an Inherited Global Selector Alias***

```
/** Skin A **/
/** ---------------------------------- **/
.MyBackColor:alias
{
  background-color: blue
}

af|decorativeBox::top
{
  -tr-rule-ref: selector(".MyBackColor:alias");
}


/** Skin B **/
/** ---------------------------------- **/
```

```
.MyBackColor:alias
{
  background-color: Red
}
```

If you specify a style property value in an extended ADF skin where the parent ADF skin also specifies a value for the style property, the ADF skinning framework applies the value in the extended ADF skin. Example 8–3 shows ADF Skin C where the `.myClass` style class specifies `Red` as the value for the `background-color` CSS property. If an application uses ADF Skin D (that extends ADF Skin C), components that apply the `.myClass` style class apply `Lime` for the `background-color` CSS property. This is because the ADF skinning framework calculates the values of statements that include values in an ADF skin (like `-tr-rule-ref`) first. The ADF skinning framework then calculates specific properties (for example, `background-color`) next. As a result, the value for the `background-color` CSS property in ADF Skin D (`Lime`) overrides the value for the `-tr-rule-ref` ADF skin property (`Blue`) or inherited values from ADF Skin C (`Red`).

---

**Note:** If you subsequently override the `.myClass` style class as follows in ADF Skin D, the value that the ADF skinning framework applies for the `background-color` CSS property is `Blue`:

```
.myClass {-tr-rule-ref: selector(".MyBlueColor:alias")}
```

---

***Example 8–3 Overriding a Local Global Selector Alias***

```
/** ADF Skin C **/
/** -------------------------------- **/
.myClass {
  background-color: Red
}

/** ADF Skin D **/
/** -------------------------------- **/
.MyBackColor:alias {
  background-color: Blue;
}

.myClass {
  background-color: Lime;
  -tr-rule-ref: selector(".MyBackColor:alias")
}
```

Consider using tools, such as Firebug for the Mozilla Firefox browser (or similar for your browser), when you run your application to determine what style property value the ADF skinning framework applies to a component selector at runtime. For more information, see Section 10.2, "Testing Changes in Your ADF Skin."

## 8.5 Referencing a Property Value from Another Selector

Rather than set a specific style property for each selector to which you want to apply the style property, you can reference the value of a property using the `-tr-property-ref` ADF skin property. You can configure this ADF skin property for global selector aliases and component-specific selectors. For example, you could define a value for the `background-color` property in a global selector alias and reference this value from multiple other selectors. If you decide at a later time to

change the value of the background-color property, you change the value in the global selector alias. All selectors that reference the background-color property using the -tr-property-ref ADF skin property update to use the change you make. The -tr-property-ref ADF skin property can also be used with compact CSS properties like, for example, border.

## 8.5.1 How to Reference a Property Value from Another Selector

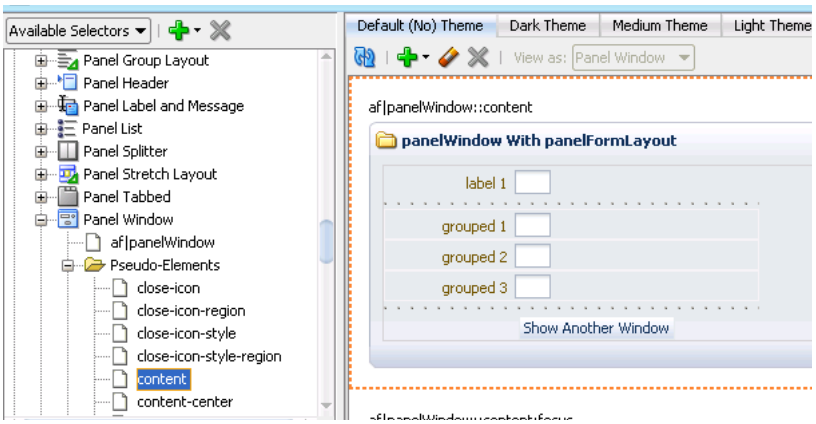
You reference the property value that you want to use for a selector using the -tr-property-ref ADF skin property.

**To reference a property value from another selector:**

1. In the Selector Tree of the visual editor, select the selector that you want to reference a property value from another selector.

   For example, if you want the content area of the panelWindow component to reference a style property defined in another selector, select **content** under the Pseudo-Elements node of the panelWindow component, as illustrated in Figure 8–9.

***Figure 8–9 Panel Window Component's content Pseudo-Element***




2. In the Property Inspector, specify the property value that you want to reference as a value for the selector's property using the -tr-property-ref ADF skin property.

   For example, assume that you created the following global selector alias:

```
.MyColor:alias {
     color: rgb(255,181,99);
     font-weight: bold;
}
```

   and that you want to reference the color property from this global selector alias for the background-color property of the content pseudo-element that you selected in step 1. In this scenario, enter the following value for the background-color property of the content pseudo-element,

```
-tr-property-ref(".MyColor:alias","color");
```

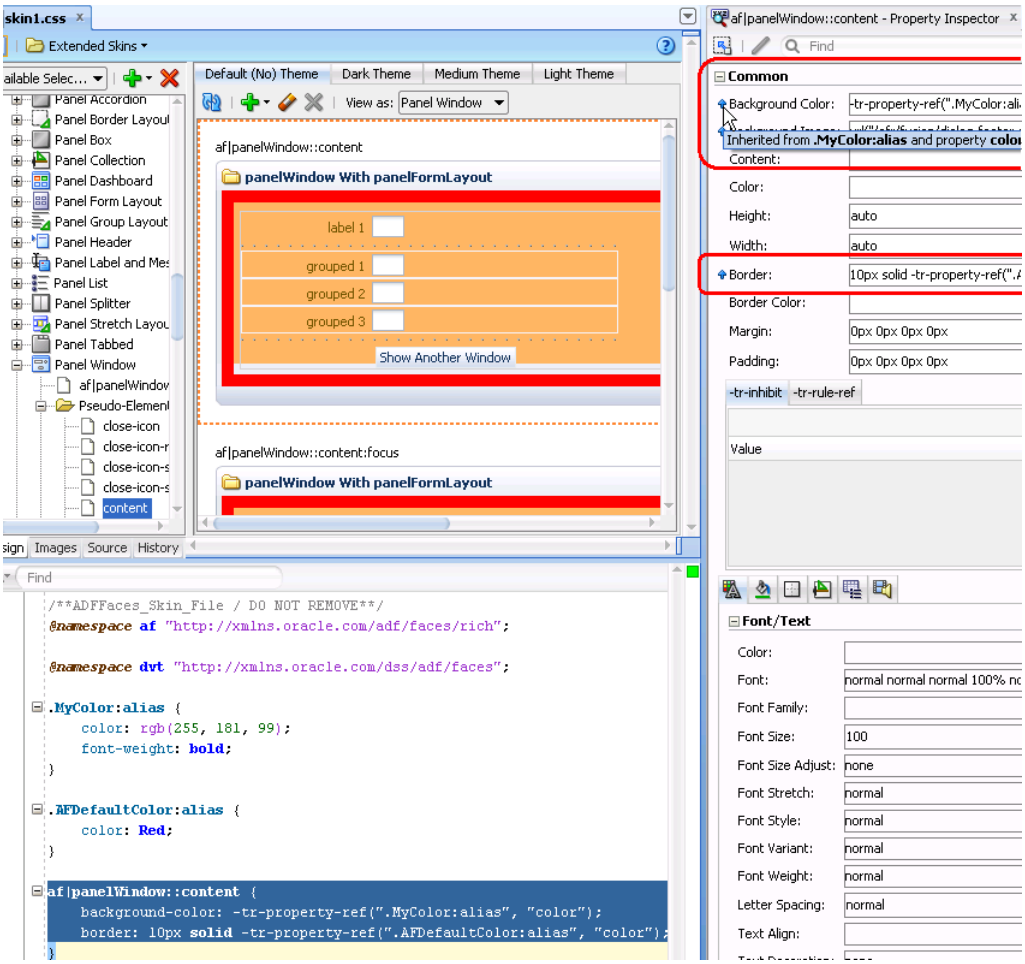
   If you want to use the -tr-property-ref in compact values, enter syntax similar to the following:

```
border: 10px solid -tr-property-ref(".AFDefaultColor:alias", "color");
```

## 8.5.2 What Happens When You Reference a Property Value from Another Selector

The Property Inspector shows that the property for which you set a value using the `-tr-property-ref` ADF skin property to reference a value from another selector inherits its value, as illustrated in Figure 8–10.

***Figure 8–10 Selector Property Referencing a Property Value from Another Selector***




Syntax similar to Example 8–4 appears in the source file of the ADF skin.

***Example 8–4 -tr-property-ref ADF Skin Property***

```
/**ADFFaces_Skin_File / DO NOT REMOVE**/
@namespace af "http://xmlns.oracle.com/adf/faces/rich";

@namespace dvt "http://xmlns.oracle.com/dss/adf/faces";

.MyColor:alias {
    color: rgb(255, 181, 99);
    font-weight: bold;
}

.AFDefaultColor:alias {
    color: Red;
}
```

```
af|panelWindow::content {
    background-color: -tr-property-ref(".MyColor:alias", "color");
    border: 10px solid -tr-property-ref(".AFDefaultColor:alias", "color");
}
```

# 9 Working with Style Classes

This chapter describes how to work with style classes. Information on how to create, modify, and apply a style class is provided in addition to describing how to configure a style class for a specific instance of a component.

This chapter includes the following sections:

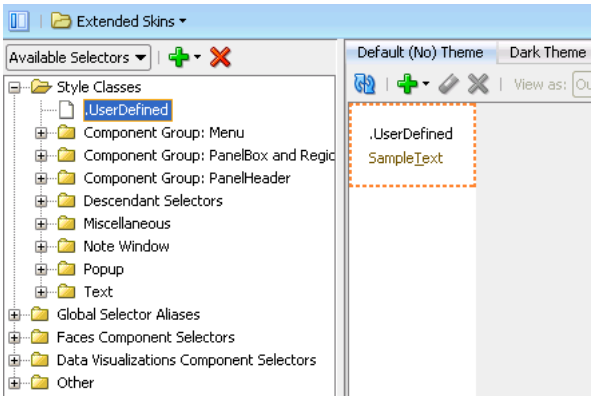


## 9.1 About Style Classes

A style class allows you to specify a number of style properties in one location in an ADF skin that you want to apply to specific instances of ADF Faces or ADF Data Visualization components. The style properties that you define for a style class take precedence over the style properties that you define for the component's selectors. Application developers can specify a style class as a value for the `styleClass` and `inlineStyle` attributes that many ADF Faces components expose. At runtime, the style properties that you defined in the style class get applied to the ADF Faces component rather than other style properties defined in the ADF skin. Style classes differ from the global selector aliases, described in Chapter 8, "Working With Global Selector Aliases," which enable you to define style properties that you want to apply to multiple ADF Faces components.

Figure 9–1 shows an ADF skin with the nodes expanded for the different categories of style classes.

***Figure 9–1 Categories of Style Class***




Each category of style class serves a purpose:

- **Component Group: Menu**: Style classes inherited from the extended ADF skins that affect menu items.
- **Component Group: PanelBox and Region**: Style classes inherited from the extended ADF skins that affect `panelBox` and `region` components.
- **Component Group: PanelHeader**: Style classes inherited from the extended ADF skins that affect `panelHeader` components.
- **Descendant Selectors**: Style classes inherited from the extended ADF skins for descendant selectors.
- **Miscellaneous**: Miscellaneous style classes inherited from the extended ADF skins. For example, this category includes the `.AFBrandingBar` style class that can be used for a branding bar containers.
- **Note Window**: Style classes inherited from the extended ADF skins that affect the `noteWindow` component.
- **Popup**: Style classes inherited from the extended ADF skins that affect the `popup` component.
- **Text**: Style classes inherited from the extended ADF skins that determine the appearance of various types of text (for example, address fields and instruction text).

Style classes that you or other users define appear under the Style Classes node as shown by the entry for the `.UserDefined` style class in Figure 9–1. For detailed descriptions of the style classes in the ADF skins that Oracle ADF provides, see the *Oracle Fusion Middleware Tag Reference for Oracle ADF Faces Skin Selectors* (for the release that pertains to the application you are skinning).

## 9.2 Creating a Style Class

You can create a new style class in your ADF skin or override a style class that your ADF skin inherits from the ADF skin that it extends.

After you create a style class, you modify it to define the style properties that you want it to contain. For more information, see Section 9.3, "Modifying a Style Class."

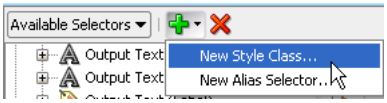
## 9.2.1 How to Create a Style Class

You can create a style class that defines the style properties you want an application developer to apply to an ADF Faces or ADF Data Visualization component using the component's `styleClass` or `inlineStyle` attribute.

**To create a style class:**

1. In the Selector Tree of the visual editor, select **New Style Class** from the Plus icon's list, as shown in Figure 9–2.

***Figure 9–2 New Style Class Option in the Selector Tree***




   The Create Style Class dialog opens.

2. Choose the appropriate option:

   - Enter a new name if you want to create a new style class that does not inherit style properties from an ADF skin that your ADF skin extends.

     **Tip:** Enter a name for the style class that indicates the purpose it serves.

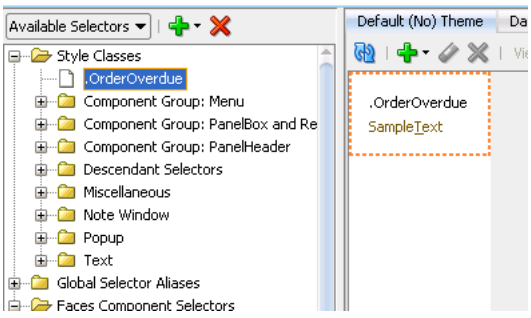
   - Enter the name of a style class that inherits style properties from an ADF skin that your ADF skin extends and for which you want to override style properties in your ADF skin.

3. Click **OK**.

## 9.2.2 What Happens When You Create a Style Class

The style class appears under the Style Classes node in the Selector Tree and a visual representation as it applies to a component appears in the Preview Pane, as shown in Figure 9–3.

***Figure 9–3 Newly-Created Style Class***




CSS syntax for the style class that you create appears in the source file of the ADF skin. Example 9–1 shows the entries that appear in the source file for the ADF skin in Figure 9–3.

***Example 9–1 CSS Syntax for a Newly-Created Style Class***

```
.OrderOverdue
```

```
{
}
```

## 9.3 Modifying a Style Class

The process to modify a style class is the same for the different categories of style class that appear in the visual editor. You select the style class in the Selector Tree and use the menus in the Preview Pane to add or remove pseudo-classes to the style class or use the Property Inspector to set or override style properties for the style class.

### 9.3.1 How to Modify a Style Class

You select the style class under the Style Classes node in the Selector Tree and modify its properties using the Property Inspector.

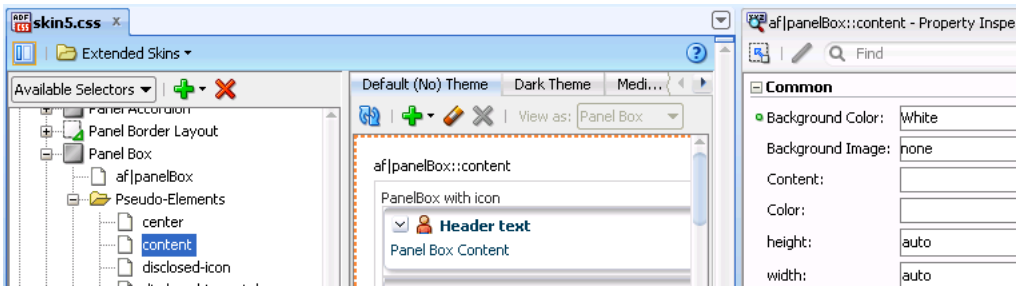
**To modify a style class:**

1. In the Selector Tree, navigate to the style class that you want to modify.
2. In the Property Inspector, make changes to the properties that you want to configure for the style class.
3. Click the **Refresh** icon to update the Preview Panel after you make changes to the style class.

## 9.4 Configuring a Style Class for a Specific Instance of a Component

You can define a style class where you define style properties to apply to a specific instance of a component. Consider, for example, a `panelBox` component that application developers use to show or hide content on a page. One page can render multiple instances of a `panelBox` component. You decide to make white the default background color for `panelBox` components, as shown in Figure 9–4

***Figure 9–4 Setting Background Color for a panelBox Component***




However, you decide that you want to render one or more instances of the `panelBox` component without the disclosure link control that allows end users to show and hide the content in the component. Additionally, you decide that you want the background color of these instances of the `panelBox` component to render with the background color set to red. To achieve this, you define style properties for a style class in the ADF skin. You then specify the style class as the value for the `styleClass` attribute for each instance of the `panelBox` component that you want to render using these style properties. Example 9–2 shows the syntax that appears in the source file of the ADF skin to achieve the outcome just described.

***Example 9–2 Syntax for a Style Class in an ADF Skin***

```
.panelBoxInstanceClass af|panelBox::disclosure-link{display:none;}
.panelBoxInstanceClass af|panelBox::content{background-color:red}
```

## 9.4.1 How to Configure a Style Class for a Specific Instance of a Component

You specify the style class as the value for the `styleClass` attribute for each instance of a component that you want to render using the style class.

**To configure a style class for a specific instance of a component:**

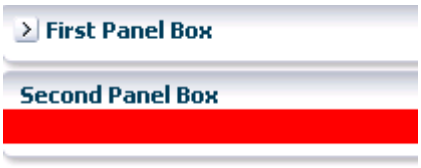
1. Create a style class, as described in Section 9.2, "Creating a Style Class."
2. In JDeveloper, set the component's `styleClass` attribute to the name of the style class you created in step 1.

   For more information about setting the component's `styleClass` attribute, see the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* (for the release that pertains to the application you are skinning).

## 9.4.2 What Happens When You Configure a Style Class for a Specific Instance of a Component

At runtime, instances of the component for which you do not specify instance-specific style properties using a style class render using the style properties defined in the component-specific selectors and global selector aliases. In Figure 9–5, this is the `panelBox` component labeled **First Panel Box**. Instances of the component for which you specify a style class as a value for the `styleClass` attribute render using the style properties defined in this style class. In Figure 9–5, this is the `panelBox` component labeled **Second Panel Box**.

***Figure 9–5 Component Rendering with Style Properties Defined in Style Class***



# 10

# Applying the Finished ADF Skin to Your Web Application

This chapter describes how to completes tasks that you need to do once you finish your ADF skin. Information is provided on how to test your ADF skin, package the completed ADF skin in an ADF Library JAR, and configure an ADF application so that it uses the completed ADF skin.

This chapter includes the following sections:



## 10.1 About Applying a Finalized ADF Skin to an Application

After you create an ADF skin where you define style properties for one or more ADF skin selectors, you may want to test the changes that you make in the ADF skin. Once you complete testing the changes in your ADF skin and are satisfied with the final ADF skin, you can package the ADF skin and associated files (images, resource bundles, and configuration files) into an ADF Library JAR to distribute for inclusion to the application projects that use the final ADF skin. Once you have distributed the final ADF skin, you configure the application to apply the ADF skin to it.

## 10.2 Testing Changes in Your ADF Skin

Once you have created an ADF skin and defined style properties that you want for one or more selectors, you may want to test how these style properties render at runtime in a browser. To do this, you apply the ADF skin to your application and run a page that renders the ADF Faces component which exposed the selector.

Consider using tools, such as Firebug for the Mozilla Firefox browser (or similar for your particular browser), when you run your application. These tools provide useful information that can help you as you iteratively develop your ADF skin. For example, in addition to inspecting changes that you have already made, these tools can help you identify the ADF skin selectors that correspond to a particular DOM element.

You can also configure context initialization parameters in the `web.xml` file of your application that allow you to:

- View changes in an ADF skin without having to restart the application

  Set the value of the following context initialization parameter to `true`:

`org.apache.myfaces.trinidad.CHECK_FILE_MODIFICATION`

- Display the full uncompressed CSS style class name at runtime

  Set the value of the following context initialization parameter to `true`:

  `org.apache.myfaces.trinidad.DISABLE_CONTENT_COMPRESSION`

Note that not all changes that you make to an ADF skin in your Fusion web application appear immediately if you set the `CHECK_FILE_MODIFICATION` to `true`. You must restart the Fusion web application to view changes that you make to icon and ADF skin properties.

For more information about context initialization parameters, see the "ADF Faces Configuration" appendix in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* (for the release that pertains to the application you are skinning).

Figure 10–1 demonstrates how the name of a style class (`fndGlobalSearchCategory`) defined in an ADF skin, and applied to an ADF Faces `commandLink` component using the component's `styleClass` attribute, is compressed when it renders in a browser.

***Figure 10–1 Compressed Style Class Name from an ADF Skin***

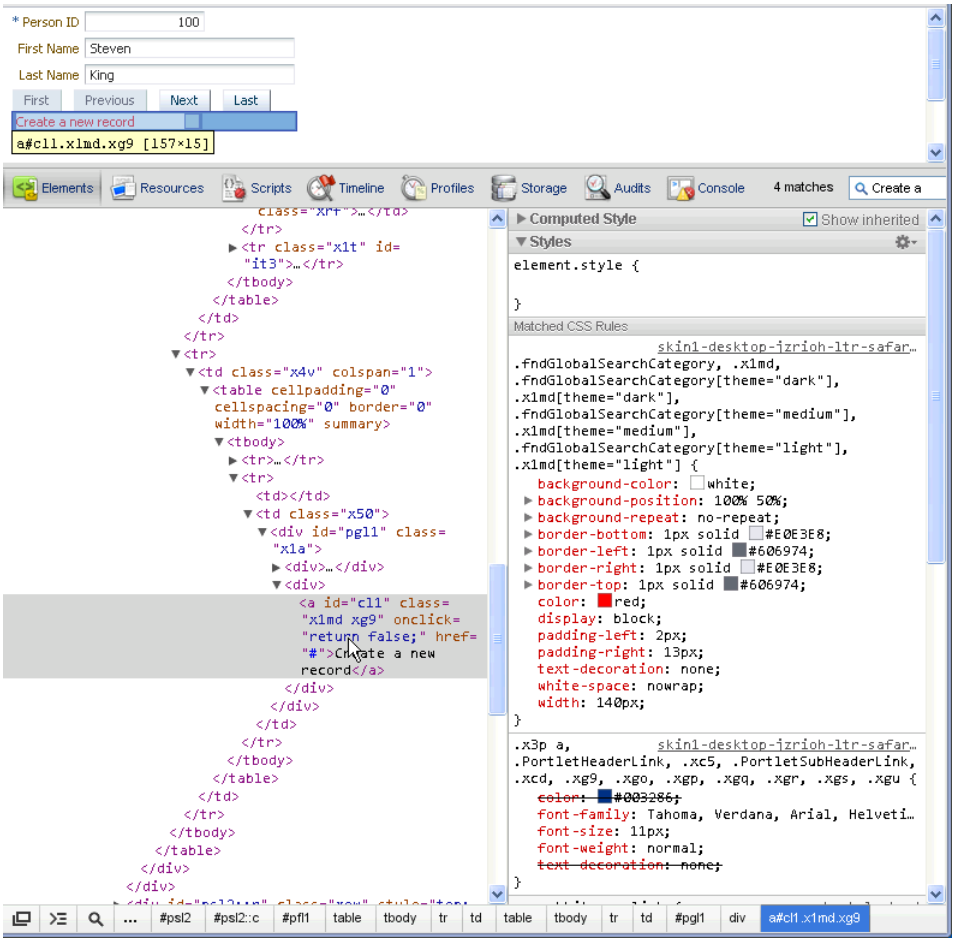



Figure 10–2 shows how the browser renders the full uncompressed name of the style class and the ADF Faces component when you set the `DISABLE_CONTENT_COMPRESSION` parameter to `true`. In Figure 10–2, the uncompressed style class `af_`

`commandLink` corresponds to the `af|commandLink` selector documented in the *Oracle Fusion Middleware Tag Reference for Oracle ADF Faces Skin Selectors* (for the release that pertains to the application you are skinning).

The uncompressed style classes that correspond to the pseudo-elements that an ADF skin selector exposes can also be identified. For example, the `tab-end` pseudo-element exposed by the `af|panelTabbed` selector (`af|panelTabbed::tab-end`) translates to the uncompressed `af_panelTabbed_tab-end` style class at runtime.

Similarly, changes that you make to the appearance of a component when it is in a specific state can also be identified or inspected using browser tools. For example, the following entry in the source file of an ADF skin allows you to define the style for the ADF Faces `panelTabbed` component when a user selects the right-hand side of the component:

```
af|panelTabbed::tab:selected af|panelTabbed::tab-end
```

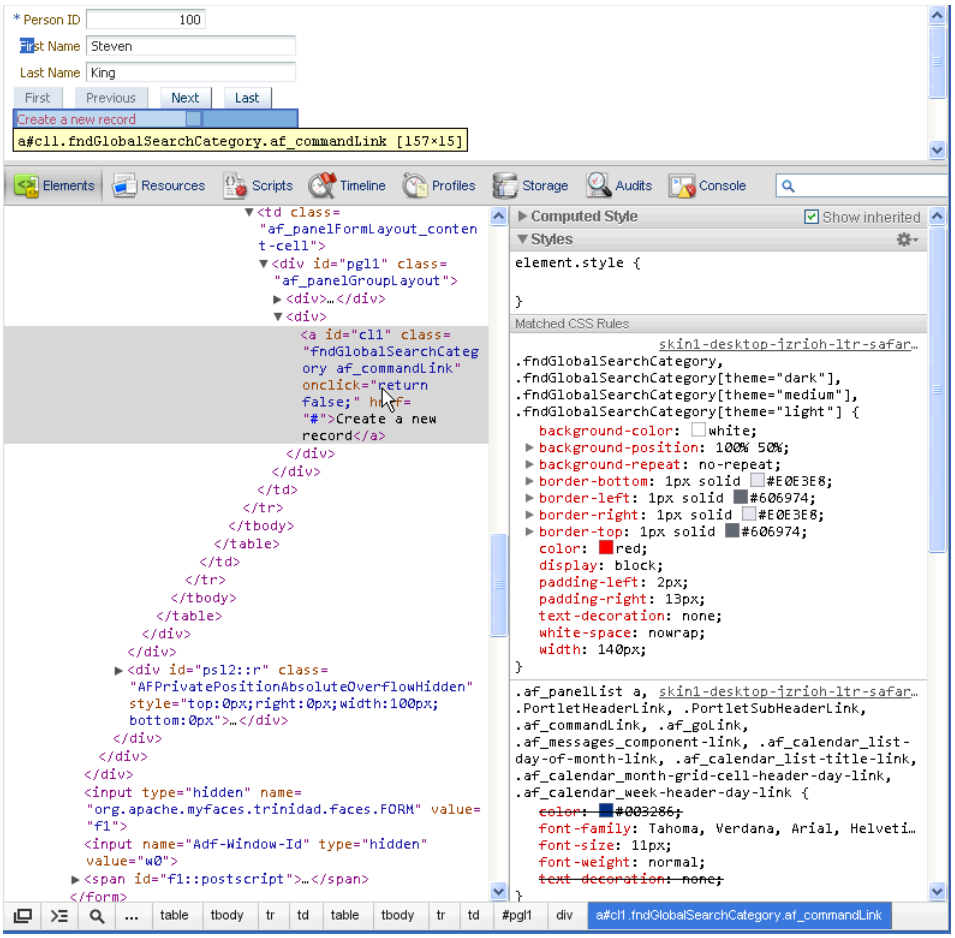
At runtime, the uncompressed style class name translates to the following:

```
.af_panelTabbed_tab.p_AFSelected .af_panelTabbed_tab-end
```

Note that `:selected` translates to `p_AFSelected` although sometimes the generated CSS does not have a `p_AFSelected` equivalent because some browsers have built-in support for that particular state, as is the case for other pseudo-classes like `:hover`.

It is recommended that you only customize the ADF skin selectors, pseudo-elements, and pseudo-classes documented in the *Oracle Fusion Middleware Tag Reference for Oracle ADF Faces Skin Selectors* and the *Oracle Fusion Middleware Data Visualization Tools Tag Reference for Oracle ADF Faces* (for the release that pertains to the application you are skinning). Customizing other ADF skin selectors may result in unexpected or inconsistent behavior for your application.

***Figure 10–2 Uncompressed Style Class Name from an ADF Skin***




## 10.2.1 How to Set Parameters for Testing Your ADF Skin

You set the `CHECK_FILE_MODIFICATION` and `DISABLE_CONTENT_COMPRESSION` context initialization parameters to `true` in the `web.xml` file of your application.

**To set parameters for testing your ADF skin:**

1. In the Application Navigator, double-click **web.xml** to open the file.
2. Add the following context initialization parameter entries and set to `true`:
   - `org.apache.myfaces.trinidad.CHECK_FILE_MODIFICATION`
   - `org.apache.myfaces.trinidad.DISABLE_CONTENT_COMPRESSION`
3. Save and close the `web.xml` file.

## 10.2.2 What Happens When You Set Parameter for Testing Your ADF Skin

Entries appear in the `web.xml` file for your application, as illustrated in Example 10–1.

***Example 10–1 web.xml Entry***

```
<context-param>
  <param-name>org.apache.myfaces.trinidad.CHECK_FILE_MODIFICATION</param-name>
  <param-value>true</param-value>
</context-param>
```

```
<context-param>
  <param-name>org.apache.myfaces.trinidad.DISABLE_CONTENT_COMPRESSION</param-name>
  <param-value>true</param-value>
</context-param>
```

Changes that you make to a selector for an ADF Faces component (other than changes to icon and skin properties) render immediately when you refresh a Fusion web application's page that renders the ADF Faces component. Using Firebug if your browser is Mozilla Firefox or Google Chrome's developer tools, you can see the uncompressed style class names that render at runtime and establish what ADF skin selector it corresponds to. Remember that setting `org.apache.myfaces.trinidad.DISABLE_CONTENT_COMPRESSION` to `true` incurs a performance cost for your Fusion web application so set it to `false` when you finish testing your changes.

## 10.3 Packaging an ADF Skin into an ADF Library JAR

You can deploy an ADF skin and associated files (for example, image files, configuration files, and resource bundles) in an ADF Library JAR. This enables you to package files required to apply an ADF skin to an application. The benefits of packaging ADF skins into an ADF Library JAR as compared to bundling them into the application are the following:

- An ADF skin can be deployed and developed separately from the application. This also helps to reduce the number of files to be checked in case some changes must be applied to the ADF skin.
- The source files for an ADF skin and images can be separated into their own ADF Library JARs. Therefore, you can partition the image base into separate ADF Library JARs, so that not all files have to be deployed with all applications.

### 10.3.1 How to Package an ADF Skin into an ADF Library JAR

Create an ADF Library JAR file deployment profile to package the ADF skin into an ADF Library JAR.

**To create an ADF Library JAR file deployment profile:**

1. In the Application Navigator, right-click the project that contains the ADF skins and choose **Deploy** > **New Deployment Profile**.
2. In the Create Deployment Profile dialog, choose **ADF Library JAR File** in the Profile Type dropdown list.
3. Enter a name for the deployment profile in the Deployment Profile Name input field and click **OK**.
4. Review the options in the Edit ADF Library JAR Deployment Profile Properties dialog that appears. For more information at any time, click **Help**.
5. Click **OK**.

**To package an ADF skin into an ADF Library JAR:**

1. In the Application Navigator, right-click the project that contains the ADF skin and choose **Deploy** > ***deployment***, where ***deployment*** is the name of the ADF Library JAR file deployment profile.
2. In the Deploy dialog Deployment Action page, click **Next** and then click **Finish**.

### 10.3.2 What Happens When You Package an ADF Skin into an ADF Library JAR

An ADF Library JAR file is written to the directory specified by the deployment profile. This ADF Library JAR contains the source file for the ADF skin, the `trinidad-skins.xml` file, image files, and any resource bundles that you created to define resource strings or to override the default strings defined for ADF Faces components. The ADF Library JAR file also contains other files from the ADF skin's project not related to skinning.

Example 10–2 shows the directory structure for a project that contains the following items for an ADF skin:

- The `trinidad-skins.xml` file
- An image file (`sort_des_ena.png`) copied into the ADF skin project
- The source file for an ADF skin (`skin1.css`)
- An .sva file (`oracle.adf.common.services.ResourceService.sva`) that is used to inspect the content of the ADF Library JAR when you import it into a project, as described in Section 4.7, "Importing ADF Skins from an ADF Library JAR."
- A resource bundle (`skinBundle.properties`) that contains string values to override strings from the default resource bundle

  For information about how to specify resource bundles that contain string values you define, see Section 7.2.1, "How to Specify an Additional Resource Bundle for an ADF Skin."

***Example 10–2 Directory Structure for an ADF Library JAR Containing an ADF Skin***

```
ADFLibraryJARRootDirectory
+---META-INF
|   |   MANIFEST.MF
|   |   oracle.adf.common.services.ResourceService.sva
|   |   trinidad-skins.xml
|   |
|   +---adf
|   |   \---skins
|   |       \---skin1
|   |           \---images
|   |               \---af_column
|   |                       sort_des_selected.png
|   |
|   \---skins
|       \---skin1
|               skin1.css
|
+---resources
|       skinBundle.properties
|
\---WEB-INF
        faces-config.xml
```

The directory paths for images in the ADF skin that appear in the ADF Library JAR are modified to include the directory path from the ADF skin project. Example 10–3 demonstrates an example of the changes that occur:

***Example 10–3 Modified Directory Path for Images in a Deployed ADF Skin***

```
// Reference to an image in an ADF skin prior to deployment to an ADF Library JAR
af|column::sorted-descending-icon-style
{
  background-image: url("images/af_column/sort_des_selected.png");
}

// Reference to an image in an ADF skin after deployment to an ADF Library JAR
af|column::sorted-descending-icon-style
{
  background-image: url("/adf/skins/skin1/images/af_column/sort_des_selected.png");
}
```

# 10.4 Applying an ADF Skin to Your Web Application

You configure an application to use an ADF skin by specifying values in the application's `trinidad-config.xml` file. You specify a value for the `<skin-family>` element that identifies the ADF skin family the application uses at runtime. If you created more than one ADF skin in the skin family, you can version these ADF skins, as described in Section 4.5, "Versioning ADF Skins." If you versioned multiple ADF skins in the same skin family, use the `<skin-version>` element to identify the specific version that you want the application to use.

Note that you can also configure an application page for your end users to dynamically select the ADF skin that they want the application to use. For more information, see the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* (for the release that pertains to the application you are skinning).

## 10.4.1 How to Apply an ADF Skin to an Application

You apply an ADF skin to an application by modifying the application's `trinidad-config.xml` file. You do this by editing the application's `trinidad-config.xml` file to specify the ADF skin family to use. Alternatively, you can select the ADF skin family from a list in the ADF View options of JDeveloper's Project Properties dialog.

**To apply an ADF skin to an application:**

1. In the Application Navigator, double-click the `trinidad-config.xml` file to open it in the source editor. By default, this file is in the **Web Content/WEB-INF** node.
2. In the `trinidad-config.xml` file, write entries to specify the value of the `<skin-family>` element and, optionally, the `<skin-version>` element as shown in Example 10–4.

## 10.4.2 What Happens When You Apply an ADF Skin to an Application

The values that you specify for the `<skin-family>` element and, optionally, the `<skin-version>` element in the `trinidad-config.xml` file determine the ADF skin that the Fusion web application uses at runtime, as shown in Example 10–4.

***Example 10–4 trinidad-config.xml File***

```
<?xml version="1.0" encoding="windows-1252"?>
<trinidad-config xmlns="http://myfaces.apache.org/trinidad/config">
```

```
    <skin-family>fusionFx</skin-family>
      <skin-version>v1.1</skin-version>
</trinidad-config>
```

# 11 Advanced Topics

This chapter provides information to help you if you make changes in the source file of an ADF skin or in the configuration files that control the usage of ADF skins. The chapter also lists and describes the ADF skins provided by Oracle ADF.

This chapter includes the following sections:



## 11.1 Referring to URLs in an ADF Skin's CSS File

An ADF skin's CSS file typically uses a URL to refer to a resource that is external to the file. For example, an image that an application uses to render with an error message. You can refer to a URL from an ADF skin's CSS file in a number of different formats. The supported formats are:

- Absolute

  You specify the complete URL to the resource. For example, a URL in the following format:

  `http://www.mycompany.com/WebApp/Skin/skin1/img/errorIcon.gif`

- Relative

  You can specify a relative URL if the URL does not start with `/` and no protocol is present. A relative URL is based on the location of the ADF skin's CSS file. For example, if the ADF skin's CSS file directory is `WebApp/Skin/skin1/` and the specified URL is `img/errorIcon.gif`, the final URL is `/WebApp/Skin/mySkin/img/errorIcon.gif`

- Context relative

  This format of URL is resolved relative to the context root of your web application. You start a context relative root with `/`. For example, if the context relative root of a web application is:

  `/WebApp`

  and the specified URL is:

  `/img/errorIcon.gif`

  the resulting URL is:

```
/WebApp/img/errorIcon.gif
```

- Server relative

  A server relative URL is resolved relative to the web server. This differs to the context relative URL in that it allows you reference a resource located in another application on the same web server. You specify the start of the URL using `//`. For example, write a URL in the following format:

  ```
  //WebApp/Skin/mySkin/img/errorIcon.gif
  ```

The format of URL that you use may be important if you create a Java Archive (JAR) file to package and distribute your ADF skin and its associated files. For more information, see Chapter 10.3, "Packaging an ADF Skin into an ADF Library JAR."

## 11.2 ADF Skinning Framework and Supported Render Kits

The ADF skinning framework supports the creation of ADF skins for the following render kits:

- `org.apache.myfaces.trinidad.desktop`
- `org.apache.myfaces.trinidad.pda`

You can use the visual editor in the ADF Skin Editor and in JDeveloper to create ADF skins for the following render kit:

`org.apache.myfaces.trinidad.desktop`

You can create an ADF skin for the following render kit using the source editor in the ADF Skin Editor or in JDeveloper:

`org.apache.myfaces.trinidad.pda`

ADF Faces components delegate the functionality of the component to a component class, and the display of the component to a renderer. By default, all tags for ADF Faces combine the associated component class with an HTML renderer, and are part of the HTML render kit. HTML render kits are included with ADF Faces for display on both desktop and PDA. You cannot customize ADF Faces renderers. However, you can customize how components display using ADF skins.

## 11.3 Configuration Files for an ADF Skin

The following list describes the configuration files associated with the project for an ADF skin. You modify values in these files while you develop your ADF skin or when you finish development and want to apply the finished ADF skin to an application.

- `trinidad-skins.xml`

  This file registers the ADF skins that you create, as described in Section 4.4, "Creating an ADF Skin File." For more information about this file, see the "Configuration in trinidad-skins.xml" section in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* (for the release that pertains to the application you are skinning).

- `trinidad-config.xml`

  You configure the `<skin-family>` element in this configuration file to tell the application what ADF skin to use, as described in Section 10.4, "Applying an ADF Skin to Your Web Application."

  For more information about this file, see the "Configuration in trinidad-config.xml" section in the *Oracle Fusion Middleware Web User Interface*

*Developer's Guide for Oracle Application Development Framework* (for the release that pertains to the application you are skinning).

- `web.xml`

  You can configure context initialization parameters in this file to facilitate the development and testing of your ADF skin, as described in Section 10.2, "Testing Changes in Your ADF Skin." You can also configure a context initialization parameter (`org.apache.myfaces.trinidad.skin.MAX_SKINS_CACHED`) to specify the maximum number of unique ADF skins (for example, `fusion` or `fusionFx-simple`) for which you store information in memory about the generated CSS files. Using this context initialization parameter can help maintain the performance of your application if you use many different skins.

  For more information about the `web.xml` file and context initialization parameters, see the "Configuration in web.xml" section in the *Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework* (for the release that pertains to the application you are skinning).

## 11.4 ADF Skins Provided by Oracle ADF

Oracle ADF provides a variety of ADF skins from which you can extend when you create a new ADF skin. It is recommended that you extend the latest version of the Fusion Simple family of ADF skins (`fusionFx-simple-v`*`N.N`*) when you create a new ADF skin, as described in Section 4.4, "Creating an ADF Skin File." In 11*g* Release 2 (11.1.2.1.0) of Oracle ADF, the Fusion Simple family of ADF skin is available in the following versions: `fusionFx-simple-v1`, and `fusionFx-simple-v1.1`, plus `fusionFx-simple-v2`.

If you target your ADF skin project at another release of Oracle ADF, as described in Section 4.2, "Creating ADF Skin Applications and ADF Skin Projects," all ADF skins described here will not be available for you to extend from.

The following list describes the differences between each ADF skin that Oracle ADF provides:

- `simple`: Contains only minimal formatting.
- `blafplus-medium`: Provides a modest amount of styling. This style extends the `simple` skin.
- `blafplus-rich`: This skin extends the `blafplus-medium` skin. Provides more styling than the `blafplus-medium` skin. For example, graphics in the `blafplus-rich` skin have rounded corners.

  **Note:** The `blafplus` skins are deprecated.

- `fusion`: Defines the default styles for ADF Faces components. This skin provides a significant amount of styling. This skin extends the `simple` skin.
- `fusion-11.1.1.3.0`: Modifies the `fusion` skin to make the hierarchy structure in certain components that render tabs clearer. These components are `panelTabbed`, `navigationPane` (attribute `hint="tabs"`), and `decorativeBox`. This skin also defines a more subtle background image for disclosed `panelAccordion` component panes to make text that appears in these panes easier to read.

- `fusionFx-v1`: This skin extends from the `fusion-11.1.1.3.0` skin. If you create an ADF skin that extends the `fusionFx-v1` skin, register it in the `trinidad-skins.xml` file using the following values:

```
<skin>
  <id>yourSkin.desktop</id>
  <family>yourSkinFamily</family>
  <extends>fusionFx-v1.desktop</extends>
  ...
</skin>
```

  Use the following value in the `trinidad-config.xml` file if you want your application to use the `fusionFx-v1` skin:

```
<skin-family>fusionFx</skin-family>
```

  The `fusionFx-v1` skin contains design improvements and changes to address a number of issues. Specifically, it adds:

  – A background color to the `.AFMaskingFrame` global style selector to prevent the display of content from an underlying frame when an inline popup displays in certain browsers.

  – A boolean ADF skin property, `-tr-stretch-dropdown-table`, for the `inputComboboxListOfValues` component. This property determines whether the table in the list stretches to show the content of the table columns or limits the width of the table to the width of the input field in the `inputComboboxListOfValues` component.

  – The `inlineFrame` component displays an image that serves as a loading indicator until the browser determines that the frame's contents have been loaded.

    You can implement this functionality in a ADF skin that you create. The `af|inlineFrame` selector has "`busy`" and "`flow`" pseudo-classes that enable you to do this. The `inlineFrame` component only generates an IFrame element when the parent component does not stretch the `inlineFrame` component (the `inlineFrame` component is flowing). Use `af|inlineFrame:busy:flow` to define a background-image style that references a loading indicator. When the parent component stretches the `inlineFrame` component, the generated content is more complex. This complexity allows you define a content image URL using the `af|inlineFrame::status-icon` and an optional additional background-image using the `af|inlineFrame::status-icon-style`. It also allows you to reuse images that other component selectors use. For example, the `carousel` component's `af|carousel::status-icon` and `af|carousel::status-icon-style` selectors. Use skinning aliases to reuse these images.

    The following global selectors have also been introduced that you can use if you implement this functionality in your ADF skin:

    * `.AFBackgroundImageStatus:alias`: use to reference the background image used in `af|inlineFrame::busy:flow`.

    * `.AFStatusIcon:alias` use to reference the `af|carousel::status-icon` and `af|inlineFrame::status-icon`.

    * `.AFStatusIconStyle:alias` use to reference the `af|carousel::status-icon-style` and `af|inlineFrame::status-icon-style`.

    A resource key (`af_inlineFrame.LABEL_FETCHING`) defines the string to display for the `inlineFrame` component's loading icon.

- `fusionFx-v1.1`: This skin extends from the `fusionFx-v1` skin. It adds support for the ability to clear Query-By-Example (QBE) filters in an `af:table` component.

  If you create an ADF skin that extends the `fusionFx-v1.1` skin, register it in the `trinidad-skins.xml` file. Use the following values in the `trinidad-skins.xml` file if you want to do this:

```
<skin>
  <id>yourSkin.desktop</id>
  <family>yourSkinFamily</family>
  <extends>fusionFx-v1.1.desktop</extends>
  …
</skin>
```

  Use the following value in the `trinidad-config.xml` file if you want your application to use the `fusionFx-v1.1` skin:

```
<skin-family>fusionFx</skin-family>
  <skin-version>v1.1<skin-version>
```

- `fusionFx-v2`: This skin extends from the `fusionFx-v1.1` skin. It makes the hierarchy structure in certain components that render tabs clearer. These components are `panelTabbed`, `navigationPane` (attribute `hint="tabs"`), and `decorativeBox`. This skin also defines a more subtle background image for disclosed `panelAccordion` component panes to make text that appears in these panes easier to read.

  If you create an ADF skin that extends the `fusionFx-v2` skin, register it in the `trinidad-skins.xml` file. Use the following values in the `trinidad-skins.xml` file to do this:

```
<skin>
  <id>yourSkin.desktop</id>
  <family>yourSkinFamily</family>
  <extends>fusionFx-v2.desktop</extends>
  ...
</skin>
```

  Use the following value in the `trinidad-config.xml` file if you want your application to use the `fusionFx-v2` skin:

```
<skin-family>fusionFx</skin-family>
<skin-version>v2</skin-version>
```

- `fusionFx-simple-v`*`N.N`*: The `fusionFx-simple` skin is the same as the `fusion` skin, but with a simplified color palette. This makes changing the color scheme for ADF skins that extend the `fusionFx-simple` skin easier than changing the color scheme for skins that extend the `fusion` skin. You can change a small number of color aliases in an ADF skin that extends the `fusionFx-simple` skin to make significant changes to the color scheme. In addition, you can use the Images window to change the color scheme of your ADF skin when you extend `fusionFx-simple` skin. For more information about the Images window, see Section 6.3, "Working with the Images Window."

- Projector skins: ADF Faces provides projector skins that you can download from the Oracle Technology Network (OTN) web site. These skins define styles for an application that you want to demonstrate to an audience using a projector. Each projector skin modifies a number of elements in a parent skin so that an application renders appropriately when displayed using table-top projectors (particularly older models of projector). These skins are useful if the audience is present at the same location as the projector. They may not be appropriate for an audience that views an application online through a web conference. ADF Faces provides the following projector skins:
  - `fusion-projector`: This skin modifies a number of elements in the `fusion` skin so that an application renders appropriately on a projector.
  - `fusionFx-v2-projector`: This skin modifies a number of elements in the `fusionFx-v2` skin so that an application renders appropriately on a projector.
  - `fusion-11.1.1.3.0-projector`: This skin modifies a number of elements in the `fusion-11.1.1.3.0` skin so that an application renders appropriately on a projector.

You can apply any of the previously listed ADF skins to your web application. For more information, see Section 10.4, "Applying an ADF Skin to Your Web Application." For a diagram that illustrates the inheritance relationship between the ADF skins, see Section 1.5, "Inheritance Relationship of the ADF Skins Provided by Oracle ADF."