Developer Guide

Develop applications using RichFaces 4

by Brian Leathem (Red Hat), Lukas Fryc (Red Hat), and Sean Rogers (Red Hat)

Introduction

The RichFaces framework is a rich component library for JavaServer Faces (JSF). The framework extends the Ajax capabilities of JSF with advanced features for the development of enterprise web applications.

RichFaces leverages several parts of the JSF 2 framework including the lifecycle, validation, conversion facilities, and management of static and dynamic resources. The RichFaces framework includes components with built-in Ajax support and a customizable look-and-feel that can be incorporated into JSF applications.

RichFaces provides a number of advantages for enterprise web application development:

- Create complex application views using out-of-the-box components. The RichFaces user interface (UI) library contains components for adding rich interactive features to JSF applications. It extends the RichFaces framework to include a large set of Ajax-enabled components that come with extensive skinning support. Additionally, the RichFaces framework is designed to be used seamlessly with other 3d-party libraries on the same page, so you have more options for developing applications.
- Write your own customized rich components with built-in Ajax support. The Component Development Kit (CDK), used for the RichFaces UI library creation, includes a code-generation facility and a templating facility using XHTML (extended hyper-text markup language) syntax.
- Generate binary resources on the fly. Extensions to JSF 2 resource-handling facilities can generate images, sounds, **Microsoft Excel** spreadsheets, and more during run-time.
- Create a modern rich user-interface with skinning technology. RichFaces provides a skinning feature that allows you to define and manage different color schemes and other parameters of the look and feel. It is possible to access the skin parameters from page code during run-time. RichFaces comes packaged with a number of skins to get you started, but you can also easily create your own customized skins too.

Getting started with RichFaces

Follow the instructions in this chapter to configure the RichFaces framework and get started with application development.

If you have existing projects that use a previous version of RichFaces, refer to the **RichFaces Migration Guide** .

2.1. Technical Requirements

The minimum technical requirements needed to get started with RichFaces are outlined below.

- Java Development Kit (JDK) 6 or higher
- An application server compliant with Java Platform, Enterprise Edition 6 (JEE6), such as JBoss EAP 6™, WildFly™ or a servlet container coupled with a JSF implementation, such as Apache Tomcat + Mojarra 2.x.
- A compliant web browser, (see the section on *Section 3.1.2, "Client requirements"* for further details)

2.1.1. Project libraries and dependencies

The RichFaces library is distributed across three jars providing all the components and services of the RichFaces framework.

RichFaces Library

- `richfaces.jar`
- `richfaces-a4j.jar`
- `richfaces-core.jar`

The framework depends on both mandatory and optional third-party dependencies. Some of the framework services are only enabled when the optional libraries are present.

Note that these dependencies may further depend on their own runtime dependencies.

Mandatory third-party dependencies

- Java Server Faces 2.x implementation

  - `javax.faces.jar` (version `2.1.28` or higher)
  - or `myfaces-impl.jar` (version `2.1.10` or higher)
- Google Guava
  - `guava.jar` (version `16.0.1`)
- CSS Parser
  - `cssparser.jar` (version `0.9.14`)
- Simple API for CSS
  - `sac.jar` (version `1.3`)

Optional third-party dependencies

- Bean validation (JSR-303) integration for client-side validation (JSR-303 API and Implementation)
  - `validation-api.jar` (version `1.0.0.GA`)
  - `hibernate-validator.jar` (version `4.2.0.Final` or higher)
- Push transport library - Atmosphere (without dependencies)
  - `atmosphere-runtime.jar` (version `1.0.18`)

    (selected compatibility modules `atmosphere-compat-*.jar` may be necessary)
- Push JMS integration (JMS API and Implementation)
  - `jms.jar` (version `1.1`)
  - `hornetq-jms.jar` (version `2.2.7.Final` or higher)
- Push CDI integration (CDI API and Implementation)
  - `cdi-api.jar` (version `1.0-SP4`)
  - `javax.inject.jar` (version `1`)
  - `jsr-250-api.jar` (version `1.0`)
  - `weld-servlet.jar` (version `1.1.18.Final`)
- Extended caching (EhCache)
  - `ehcache.jar` (version `1.6.0`)

Dependencies for servlet containers

Some of the dependencies are part of the Java EE 6 specification and thus it is not necessary to include them in projects running on Java EE applications servers. It is still necessary to include them when using servlet containers.

This does not apply to dependencies on the Servlet API: the JSP API and the EL API. These APIs are integral parts of both application servers and servlet containers.

2.2. Development environments

RichFaces applications can be developed using a range of tools, including integrated development environments (`IDEs`). This chapter covers only two such environments in detail:

- JBoss Developer Studio™, as described in *Section 2.4, "Creating a project with JBoss Developer Studio"*.
- Maven™, as described in *Section 2.5, "Creating a project with Maven"*.

Other development environments such as Idea™ or NetBeans™ could also be used for RichFaces development, but such usage is not detailed in this book.

2.3. Setting up RichFaces

Follow the instructions in this section to set up a project with the RichFaces framework and begin building applications.

1. **Download the RichFaces archive**

   Download RichFaces from the JBoss RichFaces Downloads area at *http://www.jboss.org/richfaces/download.html*. The binary files (available as a `.zip` archive) contain the following:

   - compiled, ready-to-use Java Archives (JAR files) of the RichFaces library
   - library source JAR files
   - documentation, including Java documentation and JavaScript documentation
   - archetypes
   - example source code

2. **Unzip the archive**

   Create a new directory named `RichFaces`, then unzip the archive that contains the binaries into this new directory.

2.4. Creating a project with JBoss Developer Studio™

Follow the procedure in this section to create a new RichFaces application with JBoss Developer Studio™. Ensure you are using the latest version of JBoss Developer Studio™ to take advantage of the latest feautures and stability improvements.

1. **Create a new project**

   Create a new project based on the JSF 2 environment using the RichFaces 4 template. In JBoss Developer Studio™, select **File** → **New JSF Project** from the menu. Name the project, select **JSF 2** from the **JSF Environment** drop-down box, and click the **Finish** button to create the project.

   If necessary, update the JSF 2 JAR files to the latest versions.

2. **Add the RichFaces libraries to the project**

   Add the *RichFaces libraries and their mandatory dependencies* to the project. Copy them from the location where you unzipped the RichFaces archive to the `WebContent/WEB-INF/lib/` directory of your project in **JBoss Developer Studio**.

3. **Reference the tag library**

   The RichFaces tag libraries must be referenced on each XHTML page in your project:

```
<ui:composition xmlns:rich="http://richfaces.org/rich">
<ui:composition xmlns:a4j="http://richfaces.org/a4j">
   ...
</ui:composition>
```

You are now ready to begin developing your RichFaces application. RichFaces components can be dragged and dropped from the JBoss Developer Studio™ RichFaces palette into your application's XHTML pages.

2.5. Creating a project with Maven™

Apache Maven™ is a build automation and project management tool for Java projects. Follow the instructions in this section to create a Maven™ project for RichFaces™.

2.5.1. Setting up Maven™

Maven™ can be downloaded and installed from Apache's website at *http://maven.apache.org/download.html*. Due to the use of dependency importing, Maven™ version 3.0.4 or above is required.

Once Maven™ has been installed, no further configuration is required to begin building Maven projects.

2.5.2. Using the RichFaces™ project archetype

A Maven archetype is a template for creating projects. Maven™ uses an archetype to generate a directory structure and files for a particular project, as well as creating `pom.xml` files that contain build instructions.

The RichFaces distribution includes a Maven archetype named `richfaces-archetype-simpleapp` for generating the basic structure and requirements of a RichFaces application project. Maven can obtain the archetype from maven central at *http://search.maven.org*. The archetype is also included with the RichFaces distribution in the `archetypes` directory. Follow the procedure in this section to generate a new Maven-based RichFaces project using the archetype.

1. Generate the project from the archetype

   The project can be generated with the `richfaces-archetype-simpleapp` archetype. Create a new directory for your project, then run the following Maven command in the directory:

   ```
   mvn archetype:generate -DarchetypeGroupId=org.richfaces.archetypes
    -DarchetypeArtifactId=richfaces-archetype-simpleapp -
   DarchetypeVersion=4.5.0.Beta2 -DgroupId=org.docs.richfaces -
   DartifactId=new_project
   ```

   The following parameters can be used to customize your project:

   -DgroupId
   : Defines the package for the Managed Beans

   -DartifactId
   : Defines the name of the project

   The command generates a new RichFaces project with the following structure:

   ```
   new_project
   ### pom.xml
   ### readme.txt
   ### src
       ### main
           ### java
           #   ### org
           #       ### docs
           #           ### richfaces
           #               ### RichBean.java
           ### webapp
               ### index.xhtml
               ### templates
               #   ### template.xhtml
   ```

```
            ### WEB-INF
                ### faces-config.xml
                ### web.xml
```

2. **Add test dependencies (optional)**

   Your root directory of your project contains a project descriptor file: `pom.xml`. If you wish to include modules for test-driven JSF development, add any dependencies for the tests to the `pom.xml` file.

   For testing the server-side part of your application, check out *JBoss Arquillian project* [http://www.jboss.org/arquillian]™.

   If you want to test JSF from client's perspective with ability to access state of JSF internals, use *Arquillian Warp* [https://github.com/arquillian/arquillian-extension-warp/blob/master/README.md]™.

   For automation of client-side tests in real-browser, you may want to employ *Arquillian Graphene* [http://community.jboss.org/wiki/ArquillianGraphene]™ and *Arquillian Drone* [https://docs.jboss.org/author/display/ARQ/Drone]™ extensions.

3. **Build the project**

   Build the project from the command line by entering the *mvn install* command.

   The **BUILD SUCCESSFUL** message indicates the project has been assembled and is ready to import into an IDE (integrated development environment), such as JBoss Developer Studio™.

4. **Import the project into an IDE**

   To import the project into Eclipse™ and JBoss Developer Studio™, open the importing wizard by choosing **File** → **Import** from the menu.

   a. **Select the project**

      Select **Maven** → **Existing Maven Projects** as the import source and choose the directory with the `pom.xml` file for your project.

Exporting from Maven

The ability to prepare the project for Eclipse and export it using Maven is deprecated in RichFaces 4.5.0.Beta2. The process does not support JBoss integration-specific features, such as JSF Facets.

Your project is now ready to use. Once components and functionality have been added, you can run the application on a server and access it through a web browser at the address *http://localhost:8080/jsf-app/* (where jsf-app is the name of your project).

2.6. Using RichFaces in existing JSF 2 projects

RichFaces can be added to existing JSF 2 projects by adding the new RichFaces libraries. Refer to *Step 2* and *Step 3* in *Section 2.4, "Creating a project with JBoss Developer Studio"* for details.

Application-level settings

In RichFaces 4, it is not necessary to add any extra settings to the `web.xml` or `config.xml` settings files to use the framework.

RichFaces overview

Read this chapter for technical details on the RichFaces framework.

3.1. Full technical requirements

RichFaces has been developed with an open architecture to be compatible with a wide variety of environments.

3.1.1. Server requirements

RichFaces 4 requires either of the following server technologies:

- An application server compliant with Java Platform, Enterprise Edition 6 (JEE6 or JEE6), such as JBoss EAP 6.2.3+™ or WildFly 8.0.0.Final+™.
- A major servlet container, such as Jetty 8™ or Apache Tomcat 7™.

3.1.2. Client requirements

Clients accessing RichFaces applications require a web browser. For a list of supported web browsers, refer to the *browser compatibility matrix* [https://community.jboss.org/wiki/PrioritizedRichFacesBrowsersCompatibilityMatrix] in the RichFaces wiki.

3.1.3. Development requirements

Developing applications with the RichFaces framework requires the Java Development Kit (JDK), an implementation of JavaServer Faces (JSF), and a development environment.

Java Development Kit (JDK)
: RichFaces supports the following JDK versions:

  - JDK 1.6 and higher

JavaServer Faces (JSF)
: RichFaces supports the following JSF implementations and frameworks:

  - MyFaces 2.x™
  - Mojara 2.x™

Development environment
: RichFaces can be developed using most Java development environments. The following are recommended, and used for examples in this guide:

  - JBoss Developer Studio 6.x™ and higher

- Maven 3.0.4™ and higher

3.2. Architecture

The important elements of the RichFaces framework are as follows:

- Ajax Action Components
- Ajax Containers
- Ajax Output
- Skins and Theming
- RichFaces Ajax Extensions

Read this section for details on each element.

3.2.1. Ajax Action Components

The RichFaces framework includes several Ajax Action Components and Submitting Bahaviors: `<a4j:commandButton>`, `<a4j:commandLink>`, `<a4j:poll>`, `<a4j:ajax>`, and more. Use Ajax Action Components to send Ajax requests from the client side.

3.2.2. Ajax Containers

`AjaxContainer` is an interface that marks part of the JSF tree that is decoded during an Ajax request. It only marks the JSF tree if the component or behavior sending the request does not explicitly specify an alternative. `AjaxRegion` is an implementation of this interface.

3.2.3. Ajax Output

`AjaxContainer` is an interface that marks part of the JSF tree that will be updated and rendered on the client for every Ajax request. It only marks the JSF tree if the component or behavior sending the request does not explicitly turn off automatic updates.

3.2.4. Skins and theming

RichFaces includes extensive support for application skinning. Skinning is a high-level extension to traditional CSS (Cascading Style Sheets) which allows the color scheme and appearance of an application to be easily managed. The skins simplify look-and-feel design by allowing multiple elements of the interface to be handled as manageable features, which have associated color palettes and styling. Application skins can additionally be changed on the fly during run-time, allowing user experiences to be personalized and customized.

For full details on skinning and how to create skins for the components in your application, refer to *Chapter 6, Skinning and theming*.

3.2.5. RichFaces Ajax Extensions

The RichFaces Ajax Extensions plug in to the standard JSF 2 Ajax script facility. They extend the script facility with new features and options.

3.3. Technologies

RichFaces 4 features full JSF 2 integration and uses standard web application technologies such as JavaScript, XML (Extensible Markup Language), and XHTML (Extensible Hypertext Markup Language).

3.4. Differences between JSF and RichFaces mechanisms

JavaServer Faces 2 evaluates Ajax options, such as `execute` and `render`, while rendering a page. This allows any parameters to be sent directly from the client side.

RichFaces evaluates the options when the current request is sent. This increases both the security of the data and the convenience for evaluating parameters.

For example, binding Ajax options to Java Bean properties in RichFaces allows you to evaluate the options dynamically for the current request, such as defining additional zones to render. Parameters changed manually on the client side will not influence the request processing. With JSF 2, the options have evaluated during the previous page rendering would need to be used.

3.5. Restrictions

The following restrictions apply to applications implementing the RichFaces framework:

- As with most Ajax frameworks, you should not attempt to append or delete elements on a page using RichFaces Ajax, but should instead replace them. As such, elements that are rendered conditionally should not be targeted in the `render` attributes for Ajax controls. For successful updates, an element with the same identifier as in the response must exist on the page. If it is necessary to append code to a page, include a placeholder for it (an empty element).
- JSF 2 does not allow resources such as JavaScript or Cascading Style Sheets (CSS) to be added if the element requiring the resource is not initially present in the JSF tree. As such, components added to the tree via Ajax must have any required resources already loaded. In RichFaces, any components added to the JSF tree should have components with corresponding resources included on the main page initially. To facilitate this, components can use the `rendered="false"` setting to not be rendered on the page.
- JSF does render resource links (stylesheets, scripts) in order of occurence, thus if you add `<h:outputStylesheet>` to the `<h:head>` section, JSF will render it before the RichFaces resource links (dependencies of RichFaces components). To be able to overwrite

RichFaces stylesheets and re-use RichFaces JavaScript implementation, you need to render `<h:outputStylesheet target="head">` to the `<h:body>` section (safe solution is to place it on the end of the section; however to keep readability, you can use start of the section).

- Switching RichFaces skins via Ajax during runtime should be avoided, as this requires all the stylesheets to be reloaded.

Basic concepts

Read this chapter for the basic concepts of using RichFaces in conjunction with Ajax and JavaServer Faces.

4.1. Sending an Ajax request

Many of the tags in the `r` tag library are capable of sending Ajax requests from a JavaServer Faces (JSF) page.

- The `<a4j:commandButton>` and `<a4j:commandLink>` tags are used to send an Ajax request on the `click` JavaScript event.
- The `<a4j:poll>` tag is used to send an Ajax request periodically using a timer.
- The `<a4j:ajax>` tag allows you to add Ajax functionality to standard JSF components and send Ajax request on a chosen JavaScript event, such as `keyup` or `mouseover`, for example.
- Most components in the `r` tag library have built-in Ajax support. Refer to the **RichFaces Component Reference** for details on the use of each component.

4.2. Partial tree processing

Use the `execute` attribute to specify which parts of the JSF tree to process during an Ajax request. The `execute` attribute can point to an `id` identifier of a specific component to process. Components can also be identified through the use of Expression Language (EL).

Alternatively, the `execute` attribute accepts the following keywords:

`@all`
: Every component is processed.

`@none`
: No components are processed.

`@this`
: The requesting component with the `execute` attribute is processed.

`@form`
: The form that contains the requesting component is processed.

`@region`
: The region that contains the requesting component is processed. Use the `<a4j:region>` component as a wrapper element to specify regions. Some components make use of additional keywords. These are detailed under the relevant component entry in the **RichFaces Component Reference** .

4.3. Partial view updates

Use the `render` attribute to specify which components to render for an Ajax update. The `render` attribute can point to an `id` identifier of a specific component to update. Components can also be identified through the use of Expression Language (EL).

Alternatively, the `render` attribute accepts the following keywords:

`@all`
: Every component is updated.

`@none`
: No components are updated.

`@this`
: The requesting component with the `execute` attribute is updated.

`@form`
: The form that contains the requesting component is updated.

`@region`
: The region that contains the requesting component is updated. Use the `<a4j:region>` component as a wrapper element to specify regions.

Some components make use of additional keywords. These are detailed under the relevant component entry in the **RichFaces Component Reference** .

Use the `<a4j:outputPanel>` component with the `ajaxRendered="true"` setting to always update a section irrespective of the requesting component's `render` attribute. The `<rich:message>` and `<rich:messages>` components are based on the `<a4j:outputPanel>` component, and as such will also always be updated. To override this behavior, use the `limitRender="true"` setting on the requesting component.

4.4. Component overview

The RichFaces framework is made up of two tag libraries: the `rich` and `a4j` libraries. The `a4j` tag library includes both the low-level ajax functionality, while the `rich` tag library includes the high-level components for building web applications. This allows developers to make use of custom Ajax behavior with existing componentsas well as leverage the many ready-made, self-contained components. These components don't require additional configuration in order to send requests or update.

For details on the use of the various components, refer to **RichFaces Component Reference** .

Advanced features

Read this chapter for details on some of the advanced features and configuration possibilities for the RichFaces framework.

5.1. JSF 2 integration

JavaServer Faces (JSF) is the Java-based web application framework upon which the RichFaces framework has been built. RichFaces is now integrated with JSF 2, which features several improvements to the framework.

- The standard display technology used by JSF 1 was JavaServer Pages (JSP). With JSF 2, the standard display technology has been changed to Facelets, which is a more powerful and more efficient View Declaration Language (VLD) than JSP.

5.2. Error handling

RichFaces allows standard handlers to be defined for processing different application exceptions. Custom JavaScript can be executed when these exceptions occur.

5.2.1. Client-side errors

JSF provides a global `onError` handler on the client. The handler provides the relevant error code and other associated data. The RichFaces Ajax components provide the `error` attribute if extra functionality needs to be defined.

Additional processing is available through a number of components, such as the following:

- The `<a4j:status>` component has an additional error state.
- The `<a4j:queue>` component can be used to process errors.

5.2.2. Server-side errors

Use the JSF 2 `ExceptionHandler` class to handle server-side errors such as session expiration.

5.3. Other functions

RichFaces provides a number of advanced functions, such as managing user roles and identifying elements. Refer to the **Functions** chapter in the **RichFaces Component Reference** for further details.

5.4. Resource loading

The RichFaces improves a standard JSF resource handling in order to achieve following features:

- resource optimization - serves optimized component resource dependencies (JavaScript, CSS)
- resource mapping - re-routes resource requests (maps an one resource to an another resource)

5.4.1. Configuring ResourceServlet

For leveraging RichFaces resource loading improvements, the `ResourceServlet` needs to be registered.

`ResourceServlet` is automatically registered in the Servlet 3.0 and higher environments.

In the Servlet 2.5 and lower environments, it is necessary to register the `ResourceServlet` manually in the `WEB-INF/web.xml` configuration file:

```
<servlet>
    <servlet-name>Resource Servlet</servlet-name>
    <servlet-class>org.richfaces.webapp.ResourceServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>Resource Servlet</servlet-name>
    <url-pattern>/org.richfaces.resources/*</url-pattern>
</servlet-mapping>
```

5.4.2. Resource optimization

The resource optimization feature provides optimized component dependencies - JavaScript, CSS - which are compressed and aggregated to resource packages.

The loading of compressed resource packages may lead into significant client performance boost, since many small files are aggregated into one big file - the number of HTTP connections necessary to download application resources is significantly decreased.

Example 5.1. Enabling resource optimization

To enable the resource optimization, add a following configuration to `web.xml`:

```
<context-param>
    <param-name>org.richfaces.resourceOptimization.enabled</param-name>
    <param-value>true</param-value>
</context-param>
```

Example 5.2. Resource optimization in development JSF project stage

Resource optimization is influenced by the project stage:

- resources are not compressed in the development stage and during unit-testing to enable client-side debugging
- resources are compressed in the production stage and during a system-testing to minimize network bandwidth

Switch to the development project stage during a development:

```
<context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
</context-param>
```

5.4.3. Resource mapping

The resource mapping feature maps an existing JSF resource (determined by library and name) to a another resource.

This feature can help to solve the following cases:

- providing alternative versions of JSF resources
- map several JSF resources to one
- using external resources
- moving resources to servers serving static content

5.4.3.1. Resource mapping configuration file

Configuring the resource mapping means adding new records to the class-path file `META-INF/richfaces/static-resource-mappings.properties`.

Each line in the configuration file represents one relocation.

A following sample shows a JSF resource with the name `resourceLibrary:resourceName` relocated to a resource `anotherResourceLibrary:anotherResourceName`:

```
resourceLibrary\:resourceName=anotherResourceLibrary/anotherResourceName
```

Mapping resource name to relative URL

The definition above contains a JSF resource name on the left side of the expression and a relative path on the right side.

The expression on the right side represents a path relative to a JSF resource root, thus resource path `anotherResourceLibrary/`

`anotherResourceName` actually maps to a JSF resource with name `anotherResourceLibrary:anotherResourceName`.

Additional mapping files

It is possible to define additional resource mapping configuration files by using a contextual parameter identifying the class-path locations where the files reside: `org.richfaces.resourceMapping.mappingFile` (a comma-separated list of the class-path files).

5.4.3.2. Examples of resource mapping

Example 5.3. Providing alternative file

All requests for `jquery.js` are served as requests for `jquery-alternative-version.js`:

```
jquery.js=jquery-alternative-version.js
```

Example 5.4. Mapping several resources to one

Both `some:jquery.js` and `another:jquery.js` are mapped to `final:jquery.js`:

```
some\:jquery.js=final/jquery.js
another\:jquery.js=final/jquery.js
```

Example 5.5. Using external resources

Mappings with a resource path starting with `http://` or `https://` are served as absolute resource locations:

A following sample instructs to load `jquery.js` from CDN:

```
jquery.js=http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js
```

Skinning and theming

Read this chapter for a guide to skinning and theming RichFaces applications, including how to implement themes, and details on customizing and extending skins.

6.1. What are skins?

Application skins are used with the RichFaces framework to change the appearance of an application through setting the colors and decoration of controls and components. Typically the appearance of web applications is handled through the CSS (Cascading Style Sheet) files associated with the application, but skinning allows the settings in a CSS file to be abstracted and easily edited. Skins consist of a small, generalized set of font and color parameters that can be applied to multiple different styles. This avoids repetitive coding and duplication in CSS files. CSS files are not completely replaced: skins work as a high-level extension to standard CSS.

Each skin has a set of `skin-parameters`, which are used to define the theme palette and other elements of the user interface. These parameters work together with regular CSS declarations, and can be referred to from within CSS using JavaServer Faces Expression Language (EL).

The skinning feature of RichFaces also allows skins to be changed at runtime, so users can personalize an application's appearance on the fly.

6.2. Using skins

RichFaces includes a number of predefined skins. These skins can be used in RichFaces web applications by specifying the skin name in the `org.richfaces.skin` context parameter in the `web.xml` settings file. The predefined skins are as follows:

- `DEFAULT`
- `plain`, which contains no skin parameters and is intended for embedding RichFaces components into existing projects with their own styles.
- `emeraldTown`
- `blueSky`
- `wine`
- `japanCherry`
- `ruby`
- `classic`
- `deepMarine`

To add one of these skins to your application, add the `org.richfaces.SKIN` context parameter to the `web.xml` configuration file:

```
<context-param>
    <param-name>org.richfaces.skin</param-name>
    <param-value>skin_name</param-value>
</context-param>
```

6.3. Skinning overview

RichFaces skins are implemented using the following three-level scheme:

Component stylesheets
: Stylesheets are provided for each component. CSS style parameters map to skin parameters defined in the skin property file. This mapping is accomplished through the use of ECSS files. Refer to *Section 6.3.3, "ECSS files"* for details on ECSS files.

Skin property files
: Skin property files map skin parameters to constant styles. Skin properties are defined in `skin.properties` files. Refer to *Section 6.3.1, "Skin parameter tables"* for a listing of the skin parameters used in a typical skin.

Custom style classes
: Individual components can use the `styleClass` attribute to redefine specific elements. These components then use the styles defined in a CSS file instead of the standard look for components as defined by the ECSS stylesheets.

6.3.1. Skin parameter tables

Table 6.1, "Parameter settings for the blueSky skin" lists the default values for the parameter settings in the `blueSky` skin. These values are all listed in the `blueSky.skin.properties` file, which can be customized and extended as described in *Section 6.4, "Customizing skins"*.

Table 6.1. Parameter settings for the `blueSky` skin

Parameter name	Default value
`headerBackgroundColor`	`#BED6F8`
`headerGradientColor`	`#F2F7FF`
`headTextColor`	`#000000`
`headerWeightFont`	`bold`
`generalBackgroundColor`	`#FFFFFF`
`generalTextColor`	`#000000`
`generalSizeFont`	`11px`
`generalFamilyFont`	`Arial, Verdana, sans-serif`
`controlTextColor`	`#000000`
`controlBackgroundColor`	`#FFFFFF`

Parameter name	Default value
additionalBackgroundColor	#ECF4FE
shadowBackgroundColor	#000000
shadowOpacity	1
panelBorderColor	#BED6F8
subBorderColor	#FFFFFF
calendarWeekBackgroundColor	#F5F5F5
calendarHolidaysBackgroundColor	#FFEBDA
calendarHolidaysTextColor	#FF7800
calendarCurrentBackgroundColor	#FF7800
calendarCurrentTextColor	#FFEBDA
calendarSpecBackgroundColor	#E4F5E2
calendarSpecTextColor	#000000
editorBackgroundColor	#F1F1F1
editBackgroundColor	#FEFFDA
errorColor	#FF0000
gradientType	plain
tabBackgroundColor	#C6DEFF
tabDisabledTextColor	#8DB7F3
tableHeaderBackgroundColor	#D6E6FB
tableSubHeaderBackgroundColor	#ECF4FE
tableBorderWidth	1px
tableHeaderTextColor	#0B356C
trimColor	#D6E6FB
tipBackgroundColor	#FAE6B0
tipBorderColor	#E5973E
selectControlColor	#E79A00
generalLinkColor	#0078D0
hoverLinkColor	#0090FF
visitedLinkColor	#0090FF
headerSizeFont	11px
headerFamilyFont	Arial, Verdana, sans-serif
tabSizeFont	11px
tabFamilyFont	Arial, Verdana, sans-serif
buttonSizeFont	11px

Parameter name	Default value
buttonFamilyFont	Arial, Verdana, sans-serif
tableBackgroundColor	#FFFFFF
tableFooterBackgroundColor	#CCCCCC
tableSubfooterBackgroundColor	#F1F1F1
tableBorderColor	#C0C0C0
warningColor	#FFE6E6
warningBackgroundColor	#FF0000

6.3.2. Support for round corners

Support for round borders in your skins is available via the `panelBorderRadius` skin parameter. The value of this parameter maps to the CSS 3 `border-radius` property. This CSS 3 property is ignored in older browsers, and the skin gracefully degrades to square corners.

Units of the `panelBorderRadius` skin parameter must be either `px` (pixels). or `%` (a percentage).

6.3.3. ECSS files

RichFaces uses ECSS files to add extra functionality to the skinning process. ECSS files are CSS files which use Expression Language (EL) to connect styles with skin properties.

Example 6.1. ECSS style mappings

The ECSS code for the `<rich:panel>` component contains styles for the panel and its body:

```
.rf-p{
   background-color:'#{richSkin.generalBackgroundColor}';
   color:'#{richSkin.panelBorderColor}';
   border-width:1px;
   border-style:solid;
   padding:1px;
}

.rf-p-b{
   font-size:'#{richSkin.generalSizeFont}';
   color:'#{richSkin.generalTextColor}';
   font-family:'#{richSkin.generalFamilyFont}';
   padding:10px;
}
```

`.rf-p` defines the panel styles

- The `background-color` CSS property maps to the `generalBackgroundColor` skin parameter.

- The `color` CSS property maps to the `panelBorderColor` skin parameter.

`.rf-p-b` defines the panel body styles

- The `font-family` CSS property maps to the `generalFamilyFont` skin parameter.
- The `font-size` CSS property maps to the `generalSizeFont` skin parameter.
- The `color` CSS property maps to the `generalTextColor` skin parameter.

6.4. Customizing skins

Skins in RichFaces can be customized on each of the three levels:

Skin property files
: Application interfaces can be modified by altering the values of skin parameters in the skin itself. Edit the constant values defined in the `skin.properties` file to change the style of every component mapped to that skin property.

Component stylesheets
: Mappings and other style attributes listed in a component's ECSS file can be edited. Edit the ECSS file to change the styles of all components of that type.

Custom components style classes
: Individual components can use the `styleClass` attribute to use a unique style class. Add the new style class to the application CSS and reference it from an individual component with the `styleClass` attribute.

Overwriting stylesheets in application
: You can load custom stylesheets using `<h:outputStylesheet>` which rewrites of extends styles defined for style classes of components.

Customizing skins by rewriting/extending component style classes

If you want to extend/overwrite style sheet definitions with own stylesheets, make sure you place definitions to be rendered in right order of occurence (see *Restrictions* section for details).

Example 6.2. Simple skinning example

Using any component, such as a panel, without specifying a `styleClass` will use the default skin parameters for that component.

```
<rich:panel>This is a panel without a header</rich:panel>
```

When rendered for display, the panel consists of two HTML elements: a wrapper `<div>` element and a `<div>` element for the body of the panel. The wrapper element for a panel without a specified `styleClass` is rendered as follows:

```
<div id="..." class="rf-p">
   <div id="..." class="rf-p-b">
      This is a panel without a header
   </div>
</div>
```

To customize the panel appearance according to the three-level scheme, adjust the styles according to the following approach:

1. Change the definitions for the `generalBackgroundColor` or `panelBorderColor` parameters in the skin. This will cause all panels in the application to change to the new settings.
2. Redefine the `rf-p` class in the application CSS. This will also cause all panels in the application to change to the new settings, though the skin itself has not been altered. Any properties not mapped to skin parameters should be redefined in this way.
3. Specify a different `styleClass` attribute to style the individual component. If a `styleClass` attribute is used, the specified style class is applied to the component, which could extend or override the default styles.

```
<rich:panel styleClass="customClass">...</rich:panel>
```

   The `customClass` style is added to the CSS, and is applied to the component when it is rendered for display:

```
<div class="rf-p customClass">
   ...
</div>
```

6.4.1. Creating a new skin

1. **Create the skin file**

   The name of the skin file should follow the format `new_skin_name.skin.properties` and is placed in either the `META-INF/skins/` directory or the classpath directory of your project.

2. **Define the skin constants**

   - **Define all the skin constants**

Add skin parameter constants and values to the file. All the skin parameters listed in *Table 6.1, "Parameter settings for the blueSky skin"* should be included in the skin file, with settings relevant to your new skin.

Example 6.3. `blueSky.skin.properties` file

Open the `blueSky.skin.properties` file from the `/META-INF/skins` directory in the `richfaces-core-{version}.jar` package. The file lists all the skin parameter constants shown in *Table 6.1, "Parameter settings for the blueSky skin"*.

You can use the `blueSky.skin.properties` file as a template for your new skin.

- **Extend a base skin**

  Instead of redefining an entire new skin, your skin can use an existing skin as a base on which to build new parameters. Specify a base skin by using the `baseSkin` parameter in the skin file, as shown in *Example 6.4, "Using a base skin"*.

Example 6.4. Using a base skin

This example takes the `blueSky` skin as a base and only changes the `generalSizeFont` parameter.

```
baseSkin=blueSky
generalSizeFont=12pt
```

3. **Reference the skin definition**

   Add a skin definition `<context-param>` to the `web.xml` settings file of your application:

```
<context-param>
   <param-name>org.richfaces.skin</param-name>
   <param-value>new_skin_name</param-value>
</context-param>
```

6.5. Changing skins at runtime

To allow users to change skins at runtime, use a managed bean to access the skin.

1. **Create the skin bean**

   The skin bean is a simple interface to manage the skin:

```
public class SkinBean {
```

```
    private String skin;

    public String getSkin() {
        return skin;
    }
    public void setSkin(String skin) {
        this.skin = skin;
    }
}
```

2. **Reference the skin bean**

   Add the `@ManagedBean` and `@SessionScoped` references to the class.

   - Alternatively, use EL (Expression Language) to reference the skin bean from the `web.xml` settings file.

```
<context-param>
    <param-name>org.richfaces.skin</param-name>
    <param-value>#{skinBean.skin}</param-value>
</context-param>
```

3. **Set initial skin**

   The application needs an initial skin to display before the user chooses an alternative skin. Specify the skin in your class with `@ManagedProperty`.

```
@ManagedProperty(value="blueSky")
private String skin;
```

   - Alternatively, specify the initial skin in the `web.xml` configuration file.

```
<managed-bean>
    <managed-bean-name>skinBean</managed-bean-name>
    <managed-bean-class>SkinBean</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
    <managed-property>
        <property-name>skin</property-name>
        <value>blueSky</value>
    </managed-property>
</managed-bean>
```

6.6. Skinning standard controls

Standard HTML controls used alongside RichFaces components are also themed to create a cohesive user interface.

6.6.1. Automatic skinning

The skinning style properties are automatically applied to controls based on their element names and attribute types. If the HTML elements are referenced in the standard skin stylesheets, the controls will be styled according to the mapped skin properties.

Standard HTML controls are skinned in this way by default. To override this behavior and prevent the RichFaces skins from being applied to the standard HTML controls, set the `org.richfaces.enableControlSkinning` context parameter in the `web.xml` configuration file to `false`:

```
<context-param>
   <param-name>org.richfaces.enableControlSkinning</param-name>
   <param-value>false</param-value>
</context-param>
```

6.6.2. Skinning with the `rfs-ctn` class

The skinning style properties can be determined through a separate CSS class. This method is not available by default, but is enabled through the `org.richfaces.enableControlSkinningClasses` context parameter in the `web.xml` configuration file:

```
<context-param>
   <param-name>org.richfaces.enableControlSkinningClasses</param-name>
   <param-value>true</param-value>
</context-param>
```

When enabled, a stylesheet with predefined classes offers a special CSS class named `rfs-ctn`. Reference the `rfs-ctn` class from any container element (such as a `<div>` element) to skin all the standard HTML controls in the container.

Standard HTML controls can also be specifically defined in the CSS. Refer to the `/core/impl/src/main/resources/META-INF/resources/skinning_both.ecss` file in the richfaces-ui.jarpackage for examples of specially-defined CSS classes with skin parameters for HTML controls.

Appendix A. Style classes and skin parameters

Each of the RichFaces™ components are listed below, along with their style classes and skin parameters. For further details on each component, refer to the relevant section in the **RichFaces Component Reference** .

A.1. Processing management

A.1.1. `<a4j:log>`

Table A.1. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-log` This class defines styles for the log.	`generalTextColor`	`color`
`.rf-log-popup` This class defines styles for the log when it appears as a pop-up.	No skin parameters.	
`.rf-log-popup-cnt` This class defines styles for the content of the log pop-up.	No skin parameters.	
`.rf-log-inline` This class defines styles for the log when it appears in-line.	No skin parameters.	
`.rf-log-contents` This class defines styles for the log contents.	No skin parameters.	
`.rf-log-entry-lbl` This class defines styles for a label in the log.	No skin parameters.	
`.rf-log-entry-lbl-debug` This class defines styles for the **debug** label in the log.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-log-entry-lbl-info` This class defines styles for the **information** label in the log.	No skin parameters.	
`.rf-log-entry-lbl-warn` This class defines styles for the **warning** label in the log.	No skin parameters.	
`.rf-log-entry-lbl-error` This class defines styles for the **error** label in the log.	No skin parameters.	
`.rf-log-entry-msg` This class defines styles for a message in the log.	No skin parameters.	
`.rf-log-entry-msg-debug` This class defines styles for the **debug** message in the log.	No skin parameters.	
`.rf-log-entry-msg-info` This class defines styles for the **information** message in the log.	No skin parameters.	
`.rf-log-entry-msg-warn` This class defines styles for the **warning** message in the log.	No skin parameters.	
`.rf-log-entry-msg-error` This class defines styles for the **error** message in the log.	No skin parameters.	
`.rf-log-entry-msg-xml` This class defines styles for an XML message in the log.	No skin parameters.	

A.2. Rich inputs

A.2.1. `<rich:autocomplete>`

Table A.2. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-au-fnt` This class defines styles for the auto-complete box font.	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-au-inp` This class defines styles for the auto-complete input box.	`controlBackgroundColor`	`background-color`
`.rf-au-fld` This class defines styles for the auto-complete field.	`panelBorderColor`	`border-color`
	`controlBackgroundColor`	`background-color`
`.rf-au-fld-btn` This class defines styles for a button in the auto-complete field.	No skin parameters.	
`.rf-au-btn` This class defines styles for the auto-complete box button.	`headerBackgroundColor`	`background-color`
	`panelBorderColor`	`border-left-color`
`.rf-au-btn-arrow` This class defines styles for the button arrow.	No skin parameters.	
`.rf-au-btn-arrow-dis` This class defines styles for the button arrow when it is disabled.	No skin parameters.	
`.rf-au-lst-scrl` This class defines styles for the scrollbar in the auto-complete list.	No skin parameters.	
`.rf-au-itm` This class defines styles for an item in the auto-complete list.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-au-itm-sel` This class defines styles for a selected item in the auto-complete list.	`headerBackgroundColor`	`background-color`
	`generalTextColor`	`border-color`
`.rf-au-shdw` This class defines styles for the auto-complete box shadow.	No skin parameters.	
`.rf-au-shdw-t`, `.rf-au-shdw-l`, `.rf-au-shdw-r`, `.rf-au-shdw-b` These classes define styles for the top, left, right, and bottom part of the auto-complete box shadow.	No skin parameters.	
`.rf-au-tbl` This class defines styles for a table in the auto-complete box.	No skin parameters.	

A.2.2. `<rich:calendar>`

Table A.3. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-cal-extr` This class defines the styles for a pop-up calendar exterior.	`panelBorderColor`	`border-color`
`.rf-cal-btn` This class defines styles for a calendar button.	No skin parameters.	
`.rf-cal-hdr` This class defines the styles for a calendar header.	`panelBorderColor`	`border-bottom-color`
	`additionalBackgroundColor`	`background-color`
	`generalSizeFont`	`font-size`
	`generalFamilyFont`	`font-family`
`.rf-cal-hdr-optnl` This class defines the styles for an optional header.	`panelBorderColor`	`border-bottom-color`
	`additionalBackgroundColor`	`background-color`
	`generalSizeFont`	`font-size`

Class (selector)	Skin Parameters	Mapped CSS properties
	generalFamilyFont	font-family
.rf-cal-hdr-month This class defines the styles for the month header.	headerBackgroundColor	background-color
	headerSizeFont	font-size
	headerFamilyFont	font-family
	headerWeightFont	font-weight
	headerTextColor	color
.rf-cal-ftr This class defines the styles for a calendar footer.	panelBorderColor	border-right-color, border-bottom-color
	additionalBackgroundColor	background
	generalSizeFont	font-size
	generalFamilyFont	font-family
.rf-cal-ftr-optnl This class defines the styles for an optional footer.	panelBorderColor	border-right-color, border-bottom-color
	additionalBackgroundColor	background
	generalSizeFont	font-size
	generalFamilyFont	font-family
.rf-cal-tl This class defines the styles for calendar toolbars.	headerBackgroundColor	background-color
	headerSizeFont	font-size
	headerFamilyFont	font-family
	headerWeightFont	font-weight
	headerTextColor	color
.rf-cal-tl-ftr This class defines the styles for a toolbar item in the calendar footer.	additionalBackgroundColor	background
	generalSizeFont	font-size
	generalFamilyFont	font-family
.rf-cal-tl-btn This class defines styles for a toolbar button.	No skin parameters.	
.rf-cal-tl-btn-dis This class defines styles for a disabled toolbar button.	No skin parameters.	
.rf-cal-tl-btn-hov This class defines the styles for toolbar items when it is hovered over with the mouse cursor.	calendarWeekBackgroundColor	background-color
	generalTextColor	color
	tableBackgroundColor	border-color

Class (selector)	Skin Parameters	Mapped CSS properties
	`panelBorderColor`	`border-right-color`, `border-bottom-color`
`.rf-cal-tl-btn-press` This class defines the styles for toolbar items when it is pressed.	`panelBorderColor`	`border-color`
	`panelBorderColor`	`border-right-color`, `border-bottom-color`
`.rf-cal-tl-close` This class defines styles for a **Close** button in a toolbar.	No skin parameters.	
`.rf-cal-c` This class defines the styles for regular calendar cells.	`panelBorderColor`	`border-bottom-color`, `border-right-color`
	`tableBackgroundColor`	`background-color`
	`generalSizeFont`	`font-size`
	`generalFamilyFont`	`font-family`
`.rf-cal-c-cnt` This class defines styles for the content of a cell.	No skin parameters.	
`.rf-cal-today` This class defines the styles for the cell representing today's date.	`calendarCurrentBackgroundColor`	`background-color`
	`calendarCurrentTextColor`	`color`
`.rf-cal-sel` This class defines the styles for the selected day.	`headerBackgroundColor`	`background-color`
	`headerTextColor`	`color`
`.rf-cal-hov` This class defines the styles for a cell when it is hovered over with the mouse cursor.	`calendarSpecBackgroundColor`	`background-color`
	`calendarSpecTextColor`	`color`
`.rf-cal-week` This class defines the styles for week numbers.	`panelBorderColor`	`border-bottom-color`, `border-right-color`
	`calendarWeekBackgroundColor`	`background-color`
	`generalSizeFont`	`font-size`
	`generalFamilyFont`	`font-family`

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-cal-holiday` This class defines the styles for weekends and holidays.	`calendarHolidaysBackgroundColor`	`background-color`
	`calendarHolidaysTextColor`	`color`
`.rf-cal-boundary-day` This class defines styles for an active boundary button.	No skin parameters.	
`.rf-cal-sp-inp` This class defines the styles for a spinner input field in the pop-up element for time selection.	`buttonSizeFont`	`font-size`
	`buttonFamilyFont`	`font-family`
`.rf-cal-sp-inp-cntr` This class defines the styles for a wrapper `<td>` element for a spinner input field in the pop-up element for time selection.	`controlBackgroundColor`	`background-color`
	`panelBorderColor`	`border-color`
	`subBorderColor`	`border-right-color`, `border-bottom-color`
`.rf-cal-sp-btn` This class defines the styles for a wrapper `<td>` element for spinner buttons in the pop-up element for time selection.	`headerBackgroundColor`	`background-color`, `border-color`
`.rf-cal-sp-up` This class defines styles for the **Up** spinner button.	No skin parameters.	
`.rf-cal-sp-down` This class defines styles for the **Down** spinner button.	No skin parameters.	
`.rf-cal-sp-press` This class defines styles for a spinner button when it is pressed.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-cal-edtr-shdw` This class defines the styles for the calendar editor shadow.	`tableBackgroundColor`	`background`
`.rf-cal-edtr-layout-shdw` This class defines the styles for the layout shadow of a calendar editor.	`shadowBackgroundColor`	`background-color`
`.rf-cal-edtr-btn` This class defines styles for a button in the calendar editor.	No skin parameters.	
`.rf-cal-edtr-btn-over` This class defines the styles for the calendar editor button when it is hovered over with the mouse cursor.	`panelBorderColor`	`border-color`
	`calendarSpecBackgroundColor`	`background`
`.rf-cal-edtr-btn-sel` This class defines the styles for the calendar editor button when it is selected.	`calendarCurrentBackgroundColor`	`background-color`
	`calendarCurrentTextColor`	`color`
`.rf-cal-edtr-tl-over` This class defines the styles for a toolbar item in the calendar editor when it is hovered over with the mouse cursor.	`additionalBackgroundColor`	`background`
	`tableBackgroundColor`	`border-color`
	`panelBorderColor`	`border-right-color`, `border-bottom-color`
`.rf-cal-edtr-tl-press` This class defines the styles for a toolbar item in the calendar editor when it is pressed.	`additionalBackgroundColor`	`background`
	`panelBorderColor`	`border-color`
	`tableBackgroundColor`	`border-right-color`, `border-bottom-color`
`.rf-cal-time-inp` This class defines styles for the time input field.	No skin parameters.	
`.rf-cal-time-btn` This class defines the styles for a button in the	`tableBackgroundColor`	`border-color`

Class (selector)	Skin Parameters	Mapped CSS properties
pop-up element for the calendar's time section.	panelBorderColor	border-right-color, border-bottom-color
.rf-cal-time-btn-press This class defines the styles for a pressed button in the pop-up element for the calendar's time section.	tableBackgroundColor	border-right-color, border-bottom-color
	panelBorderColor	border-color
	calendarWeekBackgroundColor	background-color
.rf-cal-timepicker-cnt This class defines the styles for the content of the pop-up element during time selection.	panelBorderColor	border-color
	additionalBackgroundColor	background
	generalSizeFont	font-size
	generalFamilyFont	font-family
.rf-cal-timepicker-inp This class defines the styles for an input field in the time picker.	generalSizeFont	font-size
	generalFamilyFont	font-family
.rf-cal-timepicker-ok This class defines styles for the **OK** button in the time picker.	No skin parameters.	
.rf-cal-timepicker-cancel This class defines styles for the **Cancel** button in the time picker.	No skin parameters.	
.rf-cal-monthpicker-cnt This class defines the styles for the content of the pop-up element during month or year selection.	panelBorderColor	border-color
	tableBackgroundColor	background
	generalSizeFont	font-size
	generalFamilyFont	font-family
.rf-cal-monthpicker-ok This class defines the styles for the **OK** button for the month picker.	additionalBackgroundColor	background
	panelBorderColor	border-top-color

Class (selector)	Skin Parameters	Mapped CSS properties
.rf-cal-monthpicker-cancel This class defines the styles for the **Cancel** button for the month picker.	additionalBackgroundColor	background
	panelBorderColor	border-top-color
.rf-cal-monthpicker-split This class defines the styles for the splitter in the month picker.	panelBorderColor	border-right-color

A.2.3. <rich:editor>

Table A.4. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
.cke_skin_richfaces	panelBorderColor	border-color
.cke_skin_richfaces .cke_wrapper	editorMainBackgroundColor	background-color
.cke_skin_richfaces .cke_dialog_body	panelBorderColor	border-color
	generalBackgroundColor	background
.cke_skin_richfaces .cke_dialog_tab	headerBackgroundColor	repeat-x
	headerWeightFont	font-weight
	headerTextColor	color
	headerFamilyFont	font-family
	headerSizeFont	font-size
.cke_skin_richfaces .cke_path a, .cke_skin_richfaces .cke_path .cke_empty	editorMainTextColor	color
.cke_skin_richfaces .cke_button a.cke_on	additionalBackgroundColor	background-color
	panelBorderColor	border-color
.cke_skin_richfaces .cke_button a:hover, .cke_skin_richfaces .cke_button a:focus, .cke_skin_richfaces .cke_button a:active	panelBorderColor	border-color
	tabBackgroundColor	background-color

Class (selector)	Skin Parameters	Mapped CSS properties
.cke_skin_richfaces .cke_rcombo a, .cke_skin_richfaces .cke_rcombo a:active, .cke_skin_richfaces .cke_rcombo a:hover	panelBorderColor	border-color
	generalSizeFont	font-size
	generalFamilyFont	font-family
	controlTextColor	color
	controlBackgroundColor	background-color
.cke_skin_richfaces .cke_rcombo .cke_combo_inlinelabel	headerBackgroundColor	background-color
	panelBorderColor	border-left-color

A.2.4. <rich:fileUpload>

Table A.5. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
.rf-fu This class defines styles for the file upload control.	generalBackgroundColor	background-color
	panelBorderColor	border-color
.rf-fu-hdr This class defines styles for the header of the file upload control.	headerBackgroundColor	background-color, border-color
.rf-fu-lst This class defines styles for lists in the file upload control.	No skin parameters.	
.rf-fu-cntr-hdn This class defines styles for the file upload container when it is hidden.	No skin parameters.	
.rf-fu-btns-lft, .rf-fu-btns-rgh These classes define styles for buttons on the left and right of the file upload control.	No skin parameters.	
.rf-fu-btn-add This class defines styles for the **Add** button in the file upload control.	trimColor	background-color
	panelBorderColor	border-color

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-fu-btn-cnt-add` This class defines styles for the content of the **Add** button in the file upload control.	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-fu-btn-add-dis` This class defines styles for the **Add** button in the file upload control when it is disabled.	`tableFooterBackgroundColor`	`background-color`
	`tableFooterBackgroundColor`	`border-color`
`.rf-fu-btn-cnt-add-dis` This class defines styles for the content of the **Add** button in the file upload control when it is disabled.	`tabDisabledTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-fu-btn-upl` This class defines styles for the **Upload** button in the file upload control.	`trimColor`	`background-color`
	`panelBorderColor`	`border-color`
`.rf-fu-btn-cnt-upl` This class defines styles for the content of the **Upload** button in the file upload control.	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-fu-btn-clr` This class defines styles for the **Clear** button in the file upload control.	`trimColor`	`background-color`
	`panelBorderColor`	`border-color`
`.rf-fu-btn-cnt-clr` This class defines styles for the content of the **Clear** button in the file upload control.	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-fu-itm` This class defines styles for an item in the file upload control.	`panelBorderColor`	`border-bottom-color`

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-fu-itm-lft`, `.rf-fu-itm-rgh` These classes define styles for items on the left and right of the file upload control.	No skin parameters.	
`.rf-fu-itm-lbl` This class defines styles for the label of an item in the file upload control.	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-fu-itm-st` This class defines styles for the status of an item in the file upload control.	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-fu-itm-lnk` This class defines styles for a link item in the file upload control.	`generalLinkColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-fu-inp` This class defines styles for the input field in the file upload control.	No skin parameters.	
`.rf-fu-inp-cntr` This class defines styles for the input field container in the file upload control.	No skin parameters.	

A.2.5. `<rich:inplaceInput>`

Table A.6. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-ii` This class defines styles for the in-place input when it is in the default state.	`editorBackgroundColor`	`background-color`
	`generalTextColor`	`border-bottom-color`
`.rf-ii-act` This class defines styles for the in-place input	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
when it is in the editing state.		
`.rf-ii-chng` This class defines styles for the in-place input when it is in the changed state.	No skin parameters.	
`.rf-ii-dis` This class defines styles for the in-place input when it is in the disabled state.	No skin parameters.	
`.rf-ii-fld` This class defines styles for the in-place input field.	`editBackgroundColor`	`background-color`, `border-bottom-color`
	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-ii-lbl` This class defines styles for the label of the in-place input.	`generalTextColor`	`color`
	`generalSizeFont`	`font-size`
`.rf-ii-dflt-lbl` This class defines styles for the default label of the in-place input.	No skin parameters.	
`.rf-ii-btn` This class defines styles for the buttons for the in-place input.	`tabBackgroundColor`	`background-color`
	`panelBorderColor`	`border-color`
`.rf-ii-btn-p` This class defines styles for the buttons for the in-place input when they are pressed.	`tabBackgroundColor`	`background-color`
	`panelBorderColor`	`border-color`
`.rf-ii-btn-set`, `.rf-ii-btn-prepos`, `.rf-ii-btn-pos` These classes define the positioning of the buttons.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-ii-btn-shdw` This class defines styles for the button shadows for the in-place input.	No skin parameters.	
`.rf-ii-btn-shdw-t`, `.rf-ii-btn-shdw-b`, `.rf-ii-btn-shdw-l`, `.rf-ii-btn-shdw-r` These classes define the top, bottom, left, and right edge of the button shadows.	No skin parameters.	
`.rf-ii-none` This class defines styles for the in-place input when it cannot be edited.	No skin parameters.	

A.2.6. `<rich:inputNumberSlider>`

Table A.7. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-insl` This class defines styles for the number slider itself.	No skin parameters.	
`.rf-insl-trc` This class defines styles for the number slider track.	`controlBackgroundColor`	`background-color`
	`panelBorderColor`	`border-bottom-color`
`.rf-insl-trc-cntr` This class defines styles for the container of the number slider track.	No skin parameters.	
`.rf-insl-mn` This class defines styles for the **minimum** label on the number slider.	`generalSizeFont`	`font-size`
	`generalFamilyFont`	`font-family`
	`generalTextColor`	`color`
	`panelBorderColor`	`border-left-color`

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-insl-mx` This class defines styles for the **maximum** label on the number slider.	`generalSizeFont`	`font-size`
	`generalFamilyFont`	`font-family`
	`generalTextColor`	`color`
	`panelBorderColor`	`border-right-color`
`.rf-insl-inp` This class defines styles for the input field on the number slider.	`generalSizeFont`	`font-size`
	`generalFamilyFont`	`font-family`
	`generalTextColor`	`color`
`.rf-insl-inp-cntr` This class defines styles for the container of the input field.	No skin parameters.	
`.rf-insl-hnd` This class defines styles for the handle on the number slider.	No skin parameters.	
`.rf-insl-hnd-cntr` This class defines styles for the container of the handle.	No skin parameters.	
`.rf-insl-hnd-sel` This class defines styles for the handle when it is selected.	No skin parameters.	
`.rf-insl-hnd-dis` This class defines styles for the handle when it is selected.	No skin parameters.	
`.rf-insl-dec`, `.rf-insl-inc` These classes define styles for the step controls to decrease and increase the number.	No skin parameters.	
`.rf-insl-dec-sel`, `.rf-insl-inc-sel` These classes define styles for the step controls when they are selected.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-insl-dec-dis`, `.rf-insl-inc-dis` These classes define styles for the step controls when they are disabled.	No skin parameters.	
`.rf-insl-tt` This class defines styles for the tool-tip on the number slider.	`generalSizeFont`	`font-size`
	`generalFamilyFont`	`font-family`
	`generalTextColor`	`color`
	`tipBorderColor`	`border`
	`tipBackgroundColor`	`background-color`

A.2.7. `<rich:inputNumberSpinner>`

Table A.8. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-insp` This class defines styles for the number spinner itself.	`panelBorderColor`	`border-color`
`.rf-insp-inp` This class defines styles for the input field on the number spinner.	`generalSizeFont`	`font-size`
	`generalFamilyFont`	`font-family`
	`generalTextColor`	`color`
	`controlBackgroundColor`	`background-color`
`.rf-insp-btns` This class defines styles for the buttons on the number spinner.	`headerBackgroundColor`	`background-color`
	`panelBorderColor`	`border-left-color`
`.rf-insp-dec`, `.rf-insp-inc` These classes define styles for the step controls to decrease and increase the number.	No skin parameters.	
`.rf-insp-dec-dis`, `.rf-insp-inc-dis` These classes define styles for the step	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
controls when they are disabled.		

A.3. Rich selects

A.3.1. `<rich:inplaceSelect>`

Table A.9. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-is` This class defines styles for the in-place select when it is in the default state.	`editorBackgroundColor`	`background-color`
	`generalTextColor`	`border-bottom-color`
`.rf-is-act` This class defines styles for the in-place select when it is in the editing state.	No skin parameters.	
`.rf-is-chng` This class defines styles for the in-place select when it is in the changed state.	No skin parameters.	
`.rf-is-dis` This class defines styles for the in-place select when it is in the disabled state.	No skin parameters.	
`.rf-is-fld` This class defines styles for the in-place select field.	`editBackgroundColor`	`background`
	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-is-opt` This class defines styles for an option for the in-place select.	`generalTextColor`	`border-color`

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-is-sel` This class defines styles for the selected option of the in-place select.	`generalTextColor`	`border-color`
`.rf-is-lbl` This class defines styles for the label of the in-place select.	No skin parameters.	
`.rf-is-dflt-lbl` This class defines styles for the default label of the in-place select.	No skin parameters.	
`.rf-is-edit` This class defines styles for the in-place select when it is being edited.	No skin parameters.	
`.rf-is-btn` This class defines styles for the buttons for the in-place select.	`tabBackgroundColor`	`background-color`
	`panelBorderColor`	`border-color`
`.rf-is-btn-p` This class defines styles for the buttons for the in-place select when they are pressed.	`tabBackgroundColor`	`background-color`
	`panelBorderColor`	`border-color`
`.rf-is-btn-set`, `.rf-is-btn-prepos`, `.rf-is-btn-pos` These classes define the positioning of the buttons.	No skin parameters.	
`.rf-is-lst-pos` This class defines the positioning of the list.	No skin parameters.	
`.rf-is-lst-dec` This class defines styles for a decreasing list for the in-place select.	`editBackgroundColor`	`background-color`
	`panelBorderColor`	`border-color`
`.rf-is-lst-scrl` This class defines styles for the list scrollbar.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-is-shdw` This class defines styles for the in-place select shadow.	No skin parameters.	
`.rf-is-shdw-t`, `.rf-is-shdw-b`, `.rf-is-shdw-l`, `.rf-is-shdw-r` These classes define the top, bottom, left, and right edge of the in-place select shadows.	No skin parameters.	
`.rf-is-btn-shdw` This class defines styles for the button shadows for the in-place select.	No skin parameters.	
`.rf-is-none` This class defines styles for the in-place select when it cannot be edited.	No skin parameters.	

A.3.2. `<rich:select>`

Table A.10. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-sel` This class defines styles for the select control itself.	No skin parameters.	
`.rf-sel-cntr` This class defines styles for the container of the select control.	`panelBorderColor`	`border-color`
`.rf-sel-inp` This class defines styles for the select control input field.	`controlBackgroundColor`	`background-color`
`.rf-sel-fld-err` This class defines styles for the input field when an error occurs.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-sel-opt` This class defines styles for an option in the select control.	`generalTextColor`	`color`
	`generalSizeFont`	`font-size`
	`generalFamilyFont`	`font-family`
`.rf-sel-sel` This class defines styles for the selected option of the select control.	`generalTextColor`	`border-color`
`.rf-sel-dflt-lbl` This class defines styles for the default label of the select control.	No skin parameters.	
`.rf-sel-btn` This class defines styles for the button of the select control.	`headerBackgroundColor`	`background-color`
	`panelBorderColor`	`border-left-color`
`.rf-sel-btn-arrow` This class defines styles for the arrow on the button.	No skin parameters.	
`.rf-sel-btn-dis` This class defines styles for the button of the select control when it is disabled.	No skin parameters.	
`.rf-sel-lst-scrl` This class defines styles for the list scrollbar.	No skin parameters.	
`.rf-sel-shdw` This class defines styles for the select control shadow.	No skin parameters.	
`.rf-sel-shdw-t`, `.rf-sel-shdw-b`, `.rf-sel-shdw-l`, `.rf-sel-shdw-r` These classes define the top, bottom, left, and right edge of the select control shadows.	No skin parameters.	

A.3.3. `<rich:orderingList>`

Table A.11. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-ord` This class defines styles for the orderingList control itself.	No skin parameters.	
`.rf-ord-cntr` This class defines styles for the container of the orderingList control.	No skin parameters.	
`.rf-ord-cptn` This class defines styles for the caption of the orderingList control.	`headerTextColor`	`color`
	`headerSizeFont`	`font-size`
	`headerFamilyFont`	`font-family`
	`headerWeightFont`	`font-weight`
`.rf-ord-lst` This class defines styles for the items list of the orderingList control.	No skin parameters.	
`.rf-ord-hdr` This class defines styles for the header of the items list.	`headerBackgroundColor`	`background-color`
	`headerTextColor`	`color`
	`headerSizeFont`	`font-size`
	`headerFamilyFont`	`font-family`
	`headerWeightFont`	`font-weight`
`.rf-ord-opt` This class defines styles for an option in the orderingList control.	`generalTextColor`	`color`
	`generalSizeFont`	`font-size`
	`generalFamilyFont`	`font-family`
`.rf-ord-sel` This class defines styles for the selected option of the orderingList control.	`generalTextColor`	`border-color`
`.rf-ord-dflt-lbl` This class defines styles for the default label of the orderingList control.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-ord-btn` This class defines styles for the button of the orderingList control.	`headerBackgroundColor`	`background-color`
	`panelBorderColor`	`border-left-color`
`.rf-ord-btn-dis` This class defines styles for the button of the orderingList control when it is disabled.	No skin parameters.	
`.rf-ord-lst-scrl` This class defines styles for the list scrollbar.	No skin parameters.	

A.3.4. `<rich:pickList>`

Table A.12. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-pick` This class defines styles for the pickList control itself.	No skin parameters.	
`.rf-pick-src-cptn, .rf-pick-tgt-cptn` These classes define styles for the source and target captions of the pickList control.	`headerTextColor`	`color`
	`headerSizeFont`	`font-size`
	`headerFamilyFont`	`font-family`
	`headerWeightFont`	`font-weight`
`.rf-pick-lst` This class defines styles for the items list of the pickList control.	No skin parameters.	
`.rf-pick-hdr` This class defines styles for the header of the items list.	`headerBackgroundColor`	`background-color`
	`headerTextColor`	`color`
	`headerSizeFont`	`font-size`
	`headerFamilyFont`	`font-family`
	`headerWeightFont`	`font-weight`

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-pick-opt` This class defines styles for an option in the pickList control.	`generalTextColor`	`color`
	`generalSizeFont`	`font-size`
	`generalFamilyFont`	`font-family`
`.rf-pick-sel` This class defines styles for the selected option of the pickList control.	`generalTextColor`	`border-color`
`.rf-pick-dflt-lbl` This class defines styles for the default label of the pickList control.	No skin parameters.	
`.rf-pick-btn` This class defines styles for the button of the pickList control.	`headerBackgroundColor`	`background-color`
	`panelBorderColor`	`border-left-color`
`.rf-pick-btn-dis` This class defines styles for the button of the pickList control when it is disabled.	No skin parameters.	
`.rf-pick-lst-scrl` This class defines styles for the list scrollbar.	No skin parameters.	

A.4. Panels and containers

A.4.1. `<rich:panel>`

Table A.13. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-p` This class defines styles for the panel itself.	`generalBackgroundColor`	`background-color`
	`panelBorderColor`	`color`
`.rf-p-hdr` This class defines styles for the header of a panel.	`headerBackgroundColor`	`background-color`, `border-color`
	`headerTextColor`	`color`
	`headerSizeFont`	`font-size`
	`headerWeightFont`	`font-weight`

Class (selector)	Skin Parameters	Mapped CSS properties
	headerFamilyFont	font-family
.rf-p-b This class defines styles for the body of a panel.	generalTextColor	color
	generalSizeFont	font-size
	generalFamilyFont	font-family

A.4.2. <rich:accordion>

Table A.14. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
.rf-ac This class defines styles for the accordion control itself.	panelBorderColor	border-color
	generalBackgroundColor	background
.rf-ac-itm-hdr This class defines styles for the header of an accordion item.	panelBorderColor	border-bottom-color
	headerBackgroundColor	background-color
	headerTextColor	color
	headerWeightFont	font-weight
	headerFamilyFont	font-family
	headerSizeFont	font-size
.rf-ac-itm-hdr-act, .rf-ac-itm-hdr-inact These classes define styles for the header when the item is either active (expanded) or inactive (collapsed).	No skin parameters.	
.rf-ac-itm-hdr-dis This class defines styles for the header when it is disabled.	tabDisabledTextColor	color
.rf-ac-itm-gr This class defines styles for an item group.	No skin parameters.	
.rf-ac-itm-cnt This class defines styles for the content of an accordion item.	panelBorderColor	border-bottom-color
	generalTextColor	color
	generalFamilyFont	font-family
	generalSizeFont	font-size

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-ac-itm-ico` This class defines styles for the item icon.	No skin parameters.	
`.rf-ac-itm-exp-ico` This class defines styles for the expanded icon for an item.	No skin parameters.	
`.rf-ac-itm-ico-act`, `.rf-ac-itm-ico-inact` These classes define styles for the icon when the item is either active (expanded) or inactive (collapsed).	No skin parameters.	
`.rf-ac-itm-lbl` This class defines styles for the item label.	No skin parameters.	
`.rf-ac-itm-lbl-act`, `.rf-ac-itm-lbl-inact` These classes define styles for the label when the item is either active (expanded) or inactive (collapsed).	No skin parameters.	

A.4.3. `<rich:collapsiblePanel>`

Table A.15. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-cp` This class defines styles for the collapsible panel itself.	`panelBorderColor`	`color`
	`generalBackgroundColor`	`background`
`.rf-cp-hdr` This class defines styles for the header of a collapsible panel.	`headerBackgroundColor`	`background-color`, `border-color`
	`headerTextColor`	`color`
	`headerWeightFont`	`font-weight`
	`headerFamilyFont`	`font-family`
	`headerSizeFont`	`font-size`

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-cp-hdr-exp`, `.rf-cp-hdr-colps` These classes define styles for the header when the item is either expanded or collapsed.	No skin parameters.	
`.rf-cp-gr` This class defines styles for a collapsible panel group.	No skin parameters.	
`.rf-cp-b` This class defines styles for the body of a collapsible panel.	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-cp-ico` This class defines styles for the panel icon.	No skin parameters.	
`.rf-cp-exp-ico` This class defines styles for the expanded icon for a panel.	No skin parameters.	
`.rf-cp-ico-exp`, `.rf-cp-ico-colps` These classes define styles for the icon when the panel is either expanded or collapsed.	No skin parameters.	
`.rf-cp-lbl` This class defines styles for the panel label.	No skin parameters.	
`.rf-cp-lbl-exp`, `.rf-cp-lbl-colps` These classes define styles for the label when the panel is either expanded or collapsed.	No skin parameters.	

A.4.4. `<rich:popupPanel>`

Table A.16. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-pp-btn` This class defines styles for the pop-up panel button.	No skin parameters.	
`.rf-pp-shade` This class defines styles for the shading that covers the page when presenting a modal pop-up panel.	No skin parameters.	
`.rf-pp-cntr` This class defines styles for the container for the pop-up panel.	`panelBorderColor`	`border`
	`generalBackgroundColor`	`background`
`.rf-pp-hdr` This class defines styles for the header of the pop-up panel.	`headerBackgroundColor`	`background`
`.rf-pp-hdr-cnt` This class defines styles for the content of the header.	`headerTextColor`	`color`
	`headerWeightFont`	`font-weight`
	`headerFamilyFont`	`font-family`
	`headerSizeFont`	`font-size`
`.rf-pp-cnt` This class defines styles for the content of the pop-up panel.	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-pp-cnt-scrlr` This class defines styles for the scroll bars of the pop-up panel.	`generalBackgroundColor`	`background`
`.rf-pp-hndlr` This class defines styles for borders of the pop-up panel. The border handler is used to re-size the panel.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-pp-hndlr-t`, `.rf-pp-hndlr-b`, `.rf-pp-hndlr-l`, `.rf-pp-hndlr-r`, `.rf-pp-hndlr-tl`, `.rf-pp-hndlr-tr`, `.rf-pp-hndlr-bl`, `.rf-pp-hndlr-br` These classes define styles for the top, bottom, left, right, top-left, top-right, bottom-left, and bottom-right edges of the border handler.	No skin parameters.	

A.4.5. `<rich:tabPanel>`

Table A.17. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-tab-hdr` This class defines styles for a tab header.	`panelBorderColor`	`border`
	`tabBackgroundColor`	`background-color`
	`generalTextColor`	`color`
`.rf-tab-hdr-act` This class defines styles for a tab header when it is active.	`additionalBackgroundColor`	`background-color`
`.rf-tab-hdr-inact` This class defines styles for a tab header when it is inactive.	No skin parameters.	
`.rf-tab-hdr-dis` This class defines styles for a tab header when it is disabled.	`tabDisabledTextColor`	`color`
`.rf-tab-hdr-tabline-vis` This class defines styles for the header tab line when it is visible.	`additionalBackgroundColor`	`background-color`
	`panelBorderColor`	`border-color`
`.rf-tab-hdr-tabs` This class defines styles for the tabs in the header.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-tab-hdr-spcr` This class defines styles for the tab header spacer.	`panelBorderColor`	`border-bottom`
`.rf-tab-lbl` This class defines styles for the tab label.	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-tab-hdn` This class defines styles for the tab when it is hidden.	No skin parameters.	
`.rf-tab-hdr-scrl-lft`, `.rf-tab-hdr-scrl-rgh` These classes define styles for the left and right controls for the tab header scroller.	`additionalBackgroundColor`	`background`
	`panelBorderColor`	`border`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-tab-hdr-tablst` This class define styles for the tab header list.	`additionalBackgroundColor`	`background`
	`panelBorderColor`	`border`
	`generalFamilyFont`	`font-family`
`.rf-tab-hdr-brd` This class define styles for the tab header border.	`tabBackgroundColor`	`background`
	`panelBorderColor`	`border`
`.rf-tab-cnt` This class define styles for the content of the tab panel.	`generalBackgroundColor`	`background`
	`panelBorderColor`	`border`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`

A.5. Tables and grids

A.5.1. `<rich:dataTable>`

Table A.18. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-dt` This class defines styles for the table.	`tableBackgroundColor`	`background-color`
	`tableBorderWidth`	`border-left-width`, `border-top-width`
	`tableBorderColor`	`border-left-color`, `border-top-color`

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-dt-cap` This class defines styles for the table caption.	No skin parameters.	
`.rf-dt-r` This class defines styles for a table row.	No skin parameters.	
`.rf-dt-fst-r` This class defines styles for the first row in a table.	No skin parameters.	
`.rf-dt-c` This class defines styles for a table cell.	`tableBackgroundColor`	`background-color`
	`tableBorderWidth`	`border-bottom-width`, `border-right-width`
	`tableBorderColor`	`border-bottom-color`, `border-right-color`
	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-dt-nd` This class defines styles for a node.	`tableBorderWidth`	`border-bottom-width`, `border-right-width`
	`tableBorderColor`	`border-bottom-color`, `border-right-color`
	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-dt-hdr` This class defines styles for a table header.	No skin parameters.	
`.rf-dt-hdr-fst` This class defines styles for the first header.	No skin parameters.	
`.rf-dt-hdr-c` This class defines styles for a header cell.	`tableHeaderBackgroundColor`	`background-color`
	`tableBorderWidth`	`border-bottom-width`, `border-right-width`
	`tableBorderColor`	`border-bottom-color`, `border-right-color`
	`tableHeaderTextColor`	`color`
	`generalFamilyFont`	`font-family`

Class (selector)	Skin Parameters	Mapped CSS properties
	generalSizeFont	font-size
.rf-dt-shdr This class defines styles for a table sub-header.	No skin parameters.	
.rf-dt-shdr-fst This class defines styles for the first sub-header.	No skin parameters.	
.rf-dt-shdr-c This class defines styles for a sub-header cell.	tableHeaderBackgroundColor	background-color
	tableBorderWidth	border-bottom-width, border-right-width
	tableBorderColor	border-bottom-color, border-right-color
	tableHeaderTextColor	color
	generalFamilyFont	font-family
	generalSizeFont	font-size
.rf-dt-ftr This class defines styles for a table footer.	No skin parameters.	
.rf-dt-ftr-fst This class defines styles for the first footer.	No skin parameters.	
.rf-dt-ftr-c This class defines styles for a footer cell.	tableFooterBackgroundColor	background-color
	tableBorderWidth	border-bottom-width, border-right-width
	tableBorderColor	border-bottom-color, border-right-color
	generalTextColor	color
	generalFamilyFont	font-family
	generalSizeFont	font-size
.rf-dt-sftr This class defines styles for a table sub-footer.	No skin parameters.	
.rf-dt-sftr-fst This class defines styles for the first sub-footer.	No skin parameters.	
.rf-dt-sftr-c This class defines styles for a sub-footer cell.	tableFooterBackgroundColor	background-color

Class (selector)	Skin Parameters	Mapped CSS properties
	tableBorderWidth	border-bottom-width, border-right-width
	tableBorderColor	border-bottom-color, border-right-color
	generalTextColor	color
	generalFamilyFont	font-family
	generalSizeFont	font-size

A.5.2. <rich:collapsibleSubTable>

Table A.19. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
.rf-cst This class defines styles for the table.	No skin parameters.	
.rf-cst-r This class defines styles for a table row.	No skin parameters.	
.rf-cst-fst-r This class defines styles for the first row in a table.	No skin parameters.	
.rf-cst-c This class defines styles for a table cell.	tableBackgroundColor	background-color
	tableBorderWidth	border-bottom-width, border-right-width
	tableBorderColor	border-bottom-color, border-right-color
	generalTextColor	color
	generalFamilyFont	font-family
	generalSizeFont	font-size
.rf-cst-hdr This class defines styles for a table header.	No skin parameters.	
.rf-cst-hdr-fst This class defines styles for the first header.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-cst-hdr-fst-r` This class defines styles for the first row in the header.	No skin parameters.	
`.rf-cst-hdr-c` This class defines styles for a header cell.	`tableSubHeaderBackgroundColor`	`background-color`
	`tableBorderWidth`	`border-bottom-width`, `border-right-width`
	`tableBorderColor`	`border-bottom-color`, `border-right-color`
	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-cst-shdr` This class defines styles for a table sub-header.	No skin parameters.	
`.rf-cst-shdr-fst` This class defines styles for the first sub-header.	No skin parameters.	
`.rf-cst-shdr-c` This class defines styles for a sub-header cell.	`tableSubHeaderBackgroundColor`	`background-color`
	`tableBorderWidth`	`border-bottom-width`, `border-right-width`
	`tableBorderColor`	`border-bottom-color`, `border-right-color`
	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-cst-ftr` This class defines styles for a table footer.	No skin parameters.	
`.rf-cst-ftr-fst` This class defines styles for the first footer.	No skin parameters.	
`.rf-cst-ftr-c` This class defines styles for a footer cell.	`tableSubFooterBackgroundColor`	`background-color`
	`tableBorderWidth`	`border-bottom-width`, `border-right-width`
	`tableBorderColor`	`border-bottom-color`, `border-right-color`

Class (selector)	Skin Parameters	Mapped CSS properties
	generalTextColor	color
	generalFamilyFont	font-family
	generalSizeFont	font-size
.rf-cst-sftr This class defines styles for a table sub-footer.	No skin parameters.	
.rf-cst-sftr-fst This class defines styles for the first sub-footer.	No skin parameters.	
.rf-cst-sftr-c This class defines styles for a sub-footer cell.	tableSubFooterBackgroundColor	background-color
	tableBorderWidth	border-bottom-width, border-right-width
	tableBorderColor	border-bottom-color, border-right-color
	generalTextColor	color
	generalFamilyFont	font-family
	generalSizeFont	font-size

A.5.3. <rich:collapsibleSubTableToggler>

Style classes (selectors)

.rf-csttg
This class defines styles for a toggle control.

.rf-csttg-exp
This class defines styles for a toggle control which expands the sub-table.

.rf-csttg-colps
This class defines styles for a toggle control which collapses the sub-table.

A.5.4. <rich:extendedDataTable>

Table A.20. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
.rf-edt This class defines styles for the table.	tableBorderWidth, tableBorderColor	border
	tableBackgroundColor	background-color

Class (selector)	Skin Parameters	Mapped CSS properties
`.rich-edt-cnt` This class defines styles for the table content.	No skin parameters.	
`.rf-edt-c` This class defines styles for a table cell.	`tableBorderWidth`, `tableBorderColor`	`border-bottom`
	`tableBorderWidth`, `tableBorderColor`	`border-right`
`.rf-edt-c-cnt` This class defines styles for the contents of a cell.	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-edt-tbl-hdr` This class defines styles for the table header.	`tableBorderWidth`, `tableBorderColor`	`border-bottom`
	`tableHeaderTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
	`tableHeaderTextColor`	`color`
`.rich-edt-hdr` This class defines styles for a header.	No skin parameters.	
`.rf-edt-hdr-c` This class defines styles for a table header cell.	`tableBorderWidth`, `tableBorderColor`	`border-bottom`
	`tableBorderWidth`, `tableBorderColor`	`border-right`
`.rf-edt-hdr-c-cnt` This class defines styles for the contents of a header cell.	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
	`tableHeaderTextColor`	`color`
`.rf-edt-tbl-ftr` This class defines styles for the table footer.	`tableBorderWidth`, `tableBorderColor`	`border-top`
	`tableFooterBackgroundColor`	`background-color`
`.rich-edt-ftr` This class defines styles for a footer.	`tableBorderWidth`, `tableBorderColor`	`border-top`
	`tableFooterBackgroundColor`	`background-color`
`.rich-edt-ftr-cnt` This class defines styles for the content of a footer.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-edt-ftr-c` This class defines styles for a table footer cell.	`tableBorderWidth`, `tableBorderColor`	`border-bottom`
	`tableBorderWidth`, `tableBorderColor`	`border-right`
`.rf-edt-ftr-c-cnt` This class defines styles for the contents of a footer cell.	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
	`generalTextColor`	`color`
`.rf-edt-ftr-emp` This class defines styles for an empty footer cell.	`tableBorderWidth`, `tableBorderColor`	`border-right`
`.rich-edt-ftr-fzn` This class defines styles for a frozen footer.	No skin parameters.	
`.rich-edt-b` This class defines styles for the body of the table.	No skin parameters.	
`.rf-edt-r-sel` This class defines styles for the selected row.	`tableBorderWidth`, `tableBorderColor`	`border-right`
`.rich-edt-r-act` This class defines styles for the active row.	No skin parameters.	
`.rich-edt-rsz` This class defines styles for the table resizer.	No skin parameters.	
`.rich-edt-rsz-cntr` This class defines styles for the resize container.	No skin parameters.	
`.rich-edt-rsz-mkr` This class defines styles for the resize marker.	`generalTextColor`	`border-left`
`.rf-edt-rord` This class defines styles for the re-order functionality.	`tableBorderWidth`, `tableBorderColor`	`border`
	`tableHeaderBackgroundColor` / `tableBackgroundColor`	`background-color`
`.rich-edt-rord-mkr` This class defines styles for the re-order marker.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rich-edt-spcr` This class defines a spacer for Internet Explorer 7compatibility.	No skin parameters.	

A.5.5. `<rich:dataGrid>`

Table A.21. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-dg` This class defines styles for the grid.	`tableBackgroundColor`	`background-color`
	`tableBorderWidth`	`border-left-width`, `border-top-width`
	`tableBorderColor`	`border-left-color`, `border-top-color`
`.rf-dg-cap` This class defines styles for the grid caption.	No skin parameters.	
`.rf-dg-r` This class defines styles for a grid row.	No skin parameters.	
`.rf-dg-c` This class defines styles for a grid cell.	`tableBorderWidth`	`border-bottom-width`, `border-right-width`
	`tableBorderColor`	`border-bottom-color`, `border-right-color`
	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-dg-nd-c` This class defines styles for a node cell.	`tableBorderWidth`	`border-bottom-width`, `border-right-width`
	`tableBorderColor`	`border-bottom-color`, `border-right-color`
	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-dg-th` This class defines styles for the grid header section.	`tableBorderWidth`	`border-bottom-width`
	`tableBorderColor`	`border-bottom-color`

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-dg-h` This class defines styles for a grid header.	No skin parameters.	
`.rf-dg-h-f` This class defines styles for the first header.	No skin parameters.	
`.rf-dg-h-r` This class defines styles for a header row.	No skin parameters.	
`.rf-dg-h-c` This class defines styles for a header cell.	`tableHeaderBackgroundColor`	`background-color`
	`tableBorderWidth`	`border-bottom-width`, `border-right-width`
	`tableBorderColor`	`border-bottom-color`, `border-right-color`
	`tableHeaderTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-dg-f` This class defines styles for a grid footer.	No skin parameters.	
`.rf-dg-f-f` This class defines styles for the first footer.	No skin parameters.	
`.rf-dg-f-c` This class defines styles for a footer cell.	`tableFooterBackgroundColor`	`background-color`
	`tableBorderWidth`	`border-bottom-width`, `border-right-width`
	`tableBorderColor`	`border-bottom-color`, `border-right-color`
	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`

A.5.6. <rich:list>

Table A.22. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-ulst-itm` This class defines styles for an item in an unordered list.	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-olst-itm` This class defines styles for an item in an unordered list.	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-dlst-trm` This class defines styles for the term of an item in a definition list.	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-dlst-dfn` This class defines styles for the definition of an item in a definition list.	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`

A.5.7. <rich:dataScroller>

Table A.23. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-ds` This class defines styles for the data scroller.	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
	`tableBackgroundColor`	`background`
`.rf-ds-btn` This class defines styles for buttons in the data scroller.	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
	`tableBorderColor`	`border-color`
	`headerBackgroundColor`	`background-color`
`.rf-ds-btn-first` This class defines styles for the **first** button.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-ds-btn-fastrwd` This class defines styles for the **fast rewind** button.	No skin parameters.	
`.rf-ds-btn-prev` This class defines styles for the **previous** button.	No skin parameters.	
`.rf-ds-btn-next` This class defines styles for the **next** button.	No skin parameters.	
`.rf-ds-btn-fastfwd` This class defines styles for the **fast forward** button.	No skin parameters.	
`.rf-ds-btn-last` This class defines styles for the **last** button.	No skin parameters.	
`.rf-ds-nmb-btn` This class defines styles for page number buttons in the data scroller.	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
	`tableBorderColor`	`border-color`
	`tableBackgroundColor`	`background-color`
`.rf-ds-press` This class defines styles for a data scroller when a control is pressed.	`tableBorderColor`	`border-color`
	`tableBackgroundColor`	`background`
`.rf-ds-act` This class defines styles for an active data scroller.	`tableBorderColor`	`color`
`.rf-ds-dis` This class defines styles for a disabled data scroller.	`tableBorderColor`	`color`

A.6. Trees

A.6.1. `<rich:tree>`

Style classes (selectors)

`.rf-tr-nd`
This class defines styles for the nodes in a tree.

`.rf-tr-nd-last`
This class defines styles for last node in a tree.

`.rf-tr-nd-colps`
This class defines styles for a collapsed tree node.

`.rf-tr-nd-exp`
This class defines styles for an expanded tree node.

A.6.2. `<rich:treeNode>`

Table A.24. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-trn` This class defines styles for a tree node.	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-trn-lbl` This class defines styles for a tree node label.	No skin parameters.	
`.rf-trn-cnt` This class defines styles for tree node content.	No skin parameters.	
`.rf-trn-sel` This class defines styles for a selected tree node.	`additionalBackgroundColor`	`background`
`.rf-trn-ldn` This class defines styles for a tree node when it is loading.	`additionalBackgroundColor`	`background`
`.rf-trn-hnd` This class defines styles for a tree node handle.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-trn-hnd-lf` This class defines styles for the handle of a leaf node.	No skin parameters.	
`.rf-trn-hnd-colps` This class defines styles for the handle of a collapsed node.	No skin parameters.	
`.rf-trn-hnd-exp` This class defines styles for the handle of an expanded node.	No skin parameters.	
`.rf-trn-hnd-ldn-fct` This class defines styles for the loading facet of a tree node handle.	No skin parameters.	
`.rf-trn-ico` This class defines styles for tree node icon.	No skin parameters.	
`.rf-trn-ico-lf` This class defines styles for the icon of a leaf node.	No skin parameters.	
`.rf-trn-ico-colps` This class defines styles for the icon of a collapsed node.	No skin parameters.	
`.rf-trn-ico-exp` This class defines styles for the icon of an expanded node.	No skin parameters.	
`.rf-trn-ico-cst` This class defines styles for a custom node icon.	No skin parameters.	

A.7. Menus and toolbars

A.7.1. `<rich:dropDownMenu>`

Table A.25. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-ddm-lbl` This class defines styles for the label of the drop-down menu.	`headerFamilyFont`	`font-family`
`.rf-ddm-dis` This class defines styles for the drop-down menu when it is disabled.	`tabDisabledTextColor`	`color`
`.rf-ddm-lbl-dis` This class defines styles for the label of the drop-down menu when it is disabled.	`headerFamilyFont`	`font-family`
`.rf-ddm-pos` This class defines the positioning of the drop-down menu.	No skin parameters.	
`.rf-ddm-lbl-unsel` This class defines styles for the label of the drop-down menu when it is unselected.	No skin parameters.	
`.rf-ddm-lst` This class defines styles for the drop-down list.	`panelBorderColor`	`border-color`
	`additionalBackgroundColor`	`background-color`
`.rf-ddm-lst-bg` This class defines styles for the background of the drop-down list.	`additionalBackgroundColor`	`border-color`
`.rf-ddm-sublst` This class defines the positioning of the menu when used as a sub-menu.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-ddm-itm` This class defines styles for a menu item.	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-ddm-itm-sel` This class defines styles for a menu item when it is selected.	`headerBackgroundColor`	`border-color`
	`tabBackgroundColor`	`background-color`
`.rf-ddm-itm-unsel` This class defines styles for a menu item when it is unselected.	No skin parameters.	
`.rf-ddm-itm-dis` This class defines styles for a menu item when it is disabled.	`tabDisabledTextColor`	`color`
`.rf-ddm-itm-lbl` This class defines styles for the label in a menu item.	`generalTextColor`	`color`
`.rf-ddm-itm-ic` This class defines styles for the icon in a menu item.	No skin parameters.	
`.rf-ddm-emptyIcon` This class defines styles for an empty icon in a menu item.	No skin parameters.	
`.rf-ddm-sep` This class defines styles for a menu separator.	`panelBorderColor`	`border-top-color`
`.rf-ddm-nd` This class defines styles for a menu node.	No skin parameters.	

A.7.2. <rich:contextMenu>

Table A.26. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-ctx-lbl` This class defines styles for the top level container of the context menu.	`headerFamilyFont`	`font-family`
`.rf-ctx-dis` This class defines styles for the context menu when it is disabled.	`tabDisabledTextColor`	`color`
`.rf-ctx-lbl-dis` This class defines styles for the top level of the context menu when it is disabled.	`headerFamilyFont`	`font-family`
`.rf-ctx-pos` This class defines the positioning of the context menu.	No skin parameters.	
`.rf-ctx-lbl-unsel` This class defines styles for the top level of the context menu when it is unselected.	No skin parameters.	
`.rf-ctx-lst` This class defines styles for the context list.	`panelBorderColor`	`border-color`
	`additionalBackgroundColor`	`background-color`
`.rf-ctx-lst-bg` This class defines styles for the background of the context list.	`additionalBackgroundColor`	`border-color`
`.rf-ctx-sublst` This class defines the positioning of the menu when used as a sub-menu.	No skin parameters.	
`.rf-ctx-itm` This class defines styles for a menu item.	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-ctx-itm-sel` This class defines styles for a menu item when it is selected.	`headerBackgroundColor`	`border-color`
	`tabBackgroundColor`	`background-color`
`.rf-ctx-itm-unsel` This class defines styles for a menu item when it is unselected.	No skin parameters.	
`.rf-ctx-itm-dis` This class defines styles for a menu item when it is disabled.	`tabDisabledTextColor`	`color`
`.rf-ctx-itm-lbl` This class defines styles for the label in a menu item.	`generalTextColor`	`color`
`.rf-ctx-itm-ic` This class defines styles for the icon in a menu item.	No skin parameters.	
`.rf-ctx-emptyIcon` This class defines styles for an empty icon in a menu item.	No skin parameters.	
`.rf-ctx-sep` This class defines styles for a menu separator.	`panelBorderColor`	`border-top-color`
`.rf-ctx-nd` This class defines styles for a menu node.	No skin parameters.	

A.7.3. `<rich:panelMenu>`

Table A.27. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-pm` This class defines styles for the panel menu itself.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-pm-gr` This class defines styles for a panel menu group.	`panelBorderColor`	`border-top-color`
`.rf-pm-exp`, `.rf-pm-colps` These classes define styles for the panel menu when it is expanded or collapsed.	No skin parameters.	
`.rf-pm-ico` This class defines styles for the panel menu icons.	No skin parameters.	
`.rf-pm-ico-exp`, `.rf-pm-ico-colps` These classes define styles for the panel menu icons when they are expanded or collapsed.	No skin parameters.	
`.rf-pm-hdr-exp`, `.rf-pm-hdr-colps` These classes define styles for the panel menu headers when they are expanded or collapsed.	No skin parameters.	
`.rf-pm-itm` This class defines styles for a panel menu item.	`panelBorderColor`	`border-top-color`
	`generalTextColor`	`color`
`.rf-pm-itm-gr` This class defines styles for a panel menu item as part of a panel menu group.	No skin parameters.	
`.rf-pm-itm:hover` This class defines styles for a panel menu item when the mouse hovers over it.	`additionalBackgroundColor`	`background-color`
`.rf-pm-itm-sel` This class defines styles for a panel menu item when it is selected.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-pm-itm-dis` This class defines styles for a panel menu item when it is disabled.	`tabDisabledTextColor`	`color`
`.rf-pm-itm-ico` This class defines styles for the icon in a panel menu item.	No skin parameters.	
`.rf-pm-itm-exp-ico` This class defines styles for the icon in a panel menu item when it is expanded.	No skin parameters.	
`.rf-pm-itm-lbl` This class defines styles for the label in a panel menu item.	`generalSizeFont`	`font-size`
	`generalFamilyFont`	`font-family`
`.rf-pm-gr` This class defines styles for a panel menu group.	`panelBorderColor`	`border-top-color`
`.rf-pm-gr-gr` This class defines styles for a panel menu group as part of another panel menu group.	No skin parameters.	
`.rf-pm-gr-sel` This class defines styles for a panel menu group when it is selected.	No skin parameters.	
`.rf-pm-gr-hdr` This class defines styles for the header of a panel menu group.	`generalTextColor`	`color`
`.rf-pm-gr-hdr:hover` This class defines styles for the header of a panel menu group when the mouse hovers over it.	`additionalBackgroundColor`	`background`

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-pm-gr-hdr-dis` This class defines styles for the header of a panel menu group when it is disabled.	`tabDisabledTextColor`	`color`
`.rf-pm-gr-ico` This class defines styles for the icon in a panel menu group.	No skin parameters.	
`.rf-pm-gr-exp-ico` This class defines styles for the icon in a panel menu group when it is expanded.	No skin parameters.	
`.rf-pm-gr-lbl` This class defines styles for the label in a panel menu group.	`generalSizeFont`	`font-size`
	`generalFamilyFont`	`font-family`
`.rf-pm-gr-cnt` This class defines styles for the content of a panel menu group.	No skin parameters.	
`.rf-pm-top-itm` This class defines styles for the top panel menu item.	`panelBorderColor`	`border-color`
	`generalTextColor`	`color`
`.rf-pm-top-itm-gr` This class defines styles for the top panel menu item as part of a panel menu group.	No skin parameters.	
`.rf-pm-top-itm:hover` This class defines styles for the top panel menu item when the mouse hovers over it.	`headerTextColor`	`color`
`.rf-pm-top-itm-sel` This class defines styles for the top panel menu item when it is selected.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-pm-top-itm-dis` This class defines styles for the top panel menu item when it is disabled.	`tabDisabledTextColor`	`color`
`.rf-pm-top-itm-ico` This class defines styles for the icon in the top panel menu item.	No skin parameters.	
`.rf-pm-top-itm-exp-ico` This class defines styles for the icon in the top panel menu item when it is expanded.	No skin parameters.	
`.rf-pm-top-itm-lbl` This class defines styles for the label in the top panel menu item.	`generalSizeFont`	`font-size`
	`generalFamilyFont`	`font-family`
`.rf-pm-top-gr` This class defines styles for the top panel menu group.	`panelBorderColor`	`border-color`
`.rf-pm-top-gr-gr` This class defines styles for the top panel menu group as part of another panel menu group.	No skin parameters.	
`.rf-pm-top-gr-sel` This class defines styles for the top panel menu group when it is selected.	No skin parameters.	
`.rf-pm-top-gr-hdr` This class defines styles for the header of the top panel menu group.	`headerTextColor`	`color`
	`headerBackgroundColor`	`background-color`
`.rf-pm-top-gr-hdr-dis` This class defines styles for the header of the top panel menu group when it is disabled.	`tabDisabledTextColor`	`color`
	`additionalBackgroundColor`	`background-color`

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-pm-top-gr-ico` This class defines styles for the icon in the top panel menu group.	No skin parameters.	
`.rf-pm-top-gr-exp-ico` This class defines styles for the icon in the top panel menu group when it is expanded.	No skin parameters.	
`.rf-pm-top-gr-lbl` This class defines styles for the label in the top panel menu group.	`generalSizeFont`	`font-size`
	`generalFamilyFont`	`font-family`
`.rf-pm-top-gr-cnt` This class defines styles for the content of the top panel menu group.	No skin parameters.	

A.7.4. `<rich:toolbar>`

Table A.28. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-tb` This class defines styles for the toolbar itself.	`panelBorderColor`	`border-color`
	`headerTextColor`	`color`
	`headerBackgroundColor`	`background-color`
	`headerFamilyFont`	`font-family`
	`headerSizeFont`	`font-size`
	`headerWeightFont`	`font-weight`
`.rf-tb-itm` This class defines styles for an item in the toolbar.	No skin parameters.	
`.rf-tb-sep` This class defines styles for a separator in the toolbar.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-tb-sep-grid`, `.rf-tb-sep-line`, `.rf-tb-sep-disc`, `.rf-tb-sep-square` These classes define styles for grid, line, disc, and square separators.	No skin parameters.	
`.rf-tb-cntr` This class defines styles for the container of the toolbar.	No skin parameters.	

A.8. Output and messages

A.8.1. `<rich:message>`

Table A.29. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-msg` This class defines styles for the message itself.	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-msg-err` This class defines styles for an error message.	`errorColor`	`color`
`.rf-msg-ftl` This class defines styles for a fatal message.	`errorColor`	`color`
`.rf-msg-inf` This class defines styles for an information message.	`generalTextColor`	`color`
`.rf-msg-wrn` This class defines styles for a warning message.	`warningTextColor`	`color`
`.rf-msg-ok` This class defines styles for a basic **OK** message.	`generalTextColor`	`color`
`.rf-msg-sum`, `.rf-msg-det` These classes define styles for the summary or details of a message.	No skin parameters.	

A.8.2. `<rich:messages>`

Table A.30. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-msgs` This class defines styles for the message itself.	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`
`.rf-msgs-err` This class defines styles for an error message.	`errorColor`	`color`
`.rf-msgs-ftl` This class defines styles for a fatal message.	`errorColor`	`color`
`.rf-msgs-inf` This class defines styles for an information message.	`generalTextColor`	`color`
`.rf-msgs-wrn` This class defines styles for a warning message.	`warningTextColor`	`color`
`.rf-msgs-ok` This class defines styles for a basic **OK** message.	`generalTextColor`	`color`
`.rf-msgs-sum`, `.rf-msgs-det` These classes define styles for the summary or details of a message.	No skin parameters.	

A.8.3. `<rich:notify>`

Table A.31. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-ntf` This class defines styles for notification	No skin parameters.	
`.rf-ntf-shdw` This class defines style of the shadow under notification box.	`headerBackgroundColor`	`background-color`
	`headerTextColor`	`color`

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-ntf-cnt` This class defines style of the content of notification box.	`panelBorderColor`	`border-color`
	`generalBackgroundColor`	`background-color`
	`panelTextColor`	`color`
`.rf-ntf-ico` This class defines style for notification icon.	No skin parameters.	
`.rf-ntf-sum` This class defines style for notification message summary.	No skin parameters.	
`.rf-ntf-det` This class defines style for notification message detail.	No skin parameters.	
`.rf-ntf-cls` This class defines style for element wrapping close button.	No skin parameters.	
`.rf-ntf-cls-ico` This class defines style for close button icon.	No skin parameters.	

A.8.4. `<rich:notifyMessage>`

Table A.32. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-ntf-inf` This class defines styles for an informative message.	`generalTextColor`	`color`
`.rf-ntf-wrn` This class defines styles for a warning notifications.	No skin parameters.	
`.rf-ntf-err` This class defines styles for a error notifications.	No skin parameters.	

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-ntf-ftl` This class defines styles for a fatal notifications.	No skin parameters.	
`.rf-ntf-inf .rf-ntf-ico`, `.rf-ntf-wrn .rf-ntf-ico`, `.rf-ntf-err .rf-ntf-ico`, `.rf-ntf-ftl .rf-ntf-ico` These classes define style for notification icon based on severity of notification message.	No skin parameters.	

A.8.5. `<rich:notifyStack>`

Table A.33. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-ntf-pos-tl` This class defines where top-left stack of notification will be positioned	No skin parameters.	
`.rf-ntf-pos-tr` This class defines where top-right stack of notification will be positioned	No skin parameters.	
`.rf-ntf-pos-bl` This class defines where bottom-left stack of notification will be positioned	No skin parameters.	
`.rf-ntf-pos-br` This class defines where bottom-right stack of notification will be positioned	No skin parameters.	

A.8.6. `<rich:progressBar>`

Table A.34. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-pb-lbl` This class defines styles for labels on the progress bar.	No skin parameters.	
`.rf-pb-prgs` This class defines styles for the progressed portion of the progress bar.	`panelBorderColor`	`border-color`
	`selectControlColor`	`background-color`
`.rf-pb-init`, `.rf-pb-fin` These classes define styles for the initial state and finished state.	`generalTextColor`	`color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`

A.8.7. `<rich:tooltip>`

Table A.35. Style classes (selectors) and corresponding skin parameters

Class (selector)	Skin Parameters	Mapped CSS properties
`.rf-tt` This class defines styles for the tool-tip itself.	No skin parameters.	
`.rf-tt-loading` This class defines styles for the tool-tip when it is loading.	No skin parameters.	
`.rf-tt-cnt` This class defines styles for the tool-tip content.	No skin parameters.	
`.rf-tt-cntr` This class defines styles for the progressed portion of the progress bar.	`tipBorderColor`	`border-color`
	`generalFamilyFont`	`font-family`
	`generalSizeFont`	`font-size`

A.9. Drag and drop

A.9.1. `<rich:dropTarget>`

Style classes (selectors)

`.rf-drp-hvr`
: This class defines styles for the drop target when a dragged item is hovering over it.

`.rf-drp-hlight`
: This class defines styles for a highlighted drop target.

A.9.2. `<rich:dragIndicator>`

Style classes (selectors)

`.rf-ind`
: This class defines styles for the drag indicator.

`.rf-ind-drag.accept`
: This class defines styles for the indicator when it is over an acceptable drop target.

`.rf-ind-drag.reject`
: This class defines styles for the indicator when it is over an unacceptable drop target.

`.rf-ind-drag.default`
: This class defines styles for the indicator when it is being dragged, and is not over any drop targets.

Appendix B. Migration Notes

This section of the guide will track any breaking changes introduced in new releases, and identify any steps required to accommodate those changes in your application.

B.1. RichFaces 4.3.7.Final

B.1.1. Autosize changes for the popupPanel

The `<rich:popupPanel>` no longer ignores the min and max sizes for width and height when `autosize=true`.

B.2. RichFaces 4.3.0.Final

B.2.1. Built-in sorting and filtering controls

The `<rich:extendedDataTable>` now has built-in sorting and filtering controls. If you have existing `<rich:extendedDataTable>` with custom sort and/or filter controls, you will want to disable the built-in sort and/or filter controls. This can be done either on a column-by-column basis, or for all columns in your applications.

For details on disabling the built-in sort and filter controls, refer to sections "External filter controls" and "External sort controls" in the RichFaces Component Reference.

B.2.2. NotifyMessage string escaping

Prior to version 4.3.0.Final, the message summary and details of the `<rich:notifyMessage>` and `<rich:notifyMessages>` components were not escaped. In the 4.3.0.Final release, an attribute `escape` was added with a default value true.

B.2.3. Select input validation

The `<rich:select>` now validates that manually entered input values match one of the values of the provided list (including support for client-side validation).

Appendix C. Revision History

Revision History		
Revision 1.0 4.0.0.Final Release	Mon Apr 11 2011	Sean Rogers
Revision 1.1 4.1.0.Final Release	Wed Nov 16 2011	Brian Leathem , Lukas Fryc
Revision 1.2 4.2.0.Final Release	Wed Feb 22 2011	Brian Leathem , Lukas Fryc