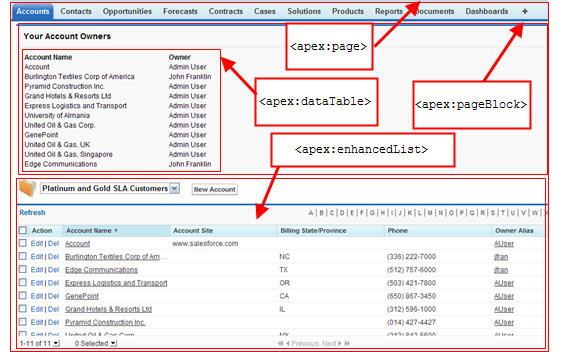
**Example of Visualforce Components and their Corresponding Tags**

## What is a Visualforce Page?

Developers can use Visualforce to create a Visualforce page definition. A page definition consists of two primary elements:

* Visualforce markup
* A Visualforce controller

## Visualforce Markup

Visualforce markup consists of Visualforce tags, HTML, JavaScript, or any other Web-enabled code embedded within a single <apex:page> tag. The markup defines the user interface components that should be included on the page, and the way they should appear.

## Visualforce Controllers

A Visualforce controller is a set of instructions that specify what happens when a user interacts with the components specified in associated Visualforce markup, such as when a user clicks a button or link. Controllers also provide access to the data that should be displayed in a page, and can modify component behavior.

A developer can either use a standard controller provided by the Force.com platform, or add custom controller logic with a class written in Apex:

* A standard controller consists of the same functionality and logic that is used for a standard Salesforce page. For example, if you use the standard Accounts controller, clicking a **Save** button in a Visualforce page results in the same behavior as clicking **Save** on a standard Account edit page.

If you use a standard controller on a page and the user doesn't have access to the object, the page will display an insufficient privileges error message. You can avoid this by checking the user's accessibility for an object and displaying components appropriately.

* A standard list controller enables you to create Visualforce pages that can display or act on a set of records. Examples of existing Salesforce pages that work with a set of records include list pages, related lists, and mass action pages.
* A custom controller is a class written in Apex that implements all of a page's logic, without leveraging a standard controller. If you use a custom controller, you can define new navigation elements or behaviors, but you must also reimplement any functionality that was already provided in a standard controller.

Like other Apex classes, custom controllers execute entirely in system mode, in which the object and field-level permissions of the current user are ignored. You can specify whether a user can execute methods in a custom controller based on the user's profile.

* A controller extension is a class written in Apex that adds to or overrides behavior in a standard or custom controller. Extensions allow you to leverage the functionality of another controller while adding your own custom logic.

Because standard controllers execute in user mode, in which the permissions, field-level security, and sharing rules of the current user are enforced, extending a standard controller allows you to build a Visualforce page that respects user permissions. Although the extension class executes in system mode, the standard controller executes in user mode. As with custom controllers, you can specify whether a user can execute methods in a controller extension based on the user's profile.

# Documentation Typographical Conventions

Apex and Visualforce​​ documentation uses the following typographical conventions.

| **Convention** | **Description** |
| --- | --- |
| Courier font | In descriptions of syntax, monospace font indicates items that you should type as shown, except for brackets. For example:   |  |  | | --- | --- | | 1 | Public class HelloWorld | |
| Italics | In descriptions of syntax, italics represent variables. You supply the actual value. In the following example, three values need to be supplied: datatype variable\_name [ = value];  If the syntax is bold and italic, the text represents a code element that needs a value supplied by you, such as a class name or variable value:   |  |  | | --- | --- | | 1 | public static class YourClassHere { ... } | |
| **Bold Courier font** | In code samples and syntax descriptions, bold courier font emphasizes a portion of the code or syntax. |
| < > | In descriptions of syntax, less-than and greater-than symbols (< >) are typed exactly as shown.   |  |  |  | | --- | --- | --- | | 1 | <apex:pageBlockTable value="{!account.Contacts}" var="contact"> | | | 2 | <apex:column value="{!contact.Name}"/> |  |  |  |  | | --- | --- | --- | | 3 | <apex:column value="{!contact.MailingCity}"/> | | | 4 | <apex:column value="{!contact.Phone}"/> |  |  |  | | --- | --- | | 5 | </apex:pageBlockTable> | |
| { } | In descriptions of syntax, braces ({ }) are typed exactly as shown.   |  |  | | --- | --- | | 1 | <apex:page> | | 2 | Hello {!$User.FirstName}! | |  |  |  | | --- | --- | | 3 | </apex:page> | |
| [ ] | In descriptions of syntax, anything included in brackets is optional. In the following example, specifying value is optional:   |  |  | | --- | --- | | 1 | data\_type variable\_name [ = value]; | |
| | | In descriptions of syntax, the pipe sign means “or”. You can do one of the following (not all). In the following example, you can create a new unpopulated set in one of two ways, or you can populate the set:   |  |  | | --- | --- | | 1 | Set<data\_type> set\_name | | 2 | [= new Set<data\_type>();] | | |  |  |  |  | | --- | --- | --- | | 3 | [= new Set<data\_type{value [, value2. . .] };] | | | | 4 | ; | |

**About the Visualforce Editor**

When editing Visualforce pages through the development mode footer or from Setup, an editor is available with the following functionality:

Syntax highlighting

The editor automatically applies syntax highlighting for keywords and all functions and operators.

Search (Search icon)

Search enables you to search for text within the current page, class, or trigger. To use search, enter a string in the Searchtextbox and click **Find Next**.

* To replace a found search string with another string, enter the new string in the Replace textbox and click **replace** to replace just that instance, or **Replace All** to replace that instance and all other instances of the search string that occur in the page, class, or trigger.
* To make the search operation case sensitive, select the **Match Case** option.
* To use a regular expression as your search string, select the **Regular Expressions** option. The regular expressions follow JavaScript's regular expression rules. A search using regular expressions can find strings that wrap over more than one line.

If you use the replace operation with a string found by a regular expression, the replace operation can also bind regular expression group variables ($1, $2, and so on) from the found search string. For example, to replace an <h1>tag with an <h2> tag and keep all the attributes on the original <h1> intact, search for <h1(\s+)(.\*)> and replace it with <h2$1$2>.

Go to line (Go To Line icon)

This button allows you to highlight a specified line number. If the line is not currently visible, the editor scrolls to that line.

Undo (Undo icon) and Redo (Redo icon)

Use undo to reverse an editing action and redo to recreate an editing action that was undone.

Font size

Select a font size from the drop-down list to control the size of the characters displayed in the editor.

Line and column position

The line and column position of the cursor is displayed in the status bar at the bottom of the editor. This can be used with go to line (Go To Line icon) to quickly navigate through the editor.

Line and character count

The total number of lines and characters is displayed in the status bar at the bottom of the editor.

The editor supports the following keyboard shortcuts:

Tab

Adds a tab at the cursor

SHIFT+Tab

Removes a tab

CTRL+f

Opens the search dialog or searches for the next occurrence of the current search

CTRL+r

Opens the search dialog or replaces the next occurrence of the current search with the specified replacement string

CTRL+g

Opens the go to line dialog

CTRL+s

Performs a quick save.

CTRL+z

Reverses the last editing action

CTRL+y

Recreates the last editing action that was undone

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Displaying Field Values with Visualforce**  Visualforce pages use the same expression language as formulas—that is, anything inside {! } is evaluated as an expression that can access values from records that are currently in context. For example, you can display the current user's first name by adding the {!$User.FirstName} expression to a page:   |  |  | | --- | --- | | 1 | <apex:page> | | 2 | Hello {!$User.FirstName}! | |  |  |  | | --- | --- | | 3 | </apex:page> |   $User is a global variable that always represents the current user record. All global variables are referenced with a $ symbol. For a list of global variables that you can use in Visualforce, see Global Variables.  To access fields from a record that is not globally available, like a specific account, contact, or custom object record, you need to associate your page with a *controller*. Controllers provide pages with the data and business logic that make your application run, including the logic that specifies how to access a particular object's records. While you can define a custom controller for any page with Apex, Salesforce includes standard controllers for every standard and custom object.  For example, to use the standard controller for accounts, add the standardController attribute to the <apex:page> tag, and assign it the name of the account object:   |  |  |  | | --- | --- | --- | | 1 | <apex:page standardController="Account"> | | | 2 | Hello {!$User.FirstName}! |  |  |  | | --- | --- | | 3 | </apex:page> |   After you save your page, the Accounts tab is highlighted for the page, and the look-and-feel for the components on the page match the Accounts tab. Additionally, you can now access fields on the account record currently in context by using {!account.*<fieldName>*} expression syntax.  For example, to display an account's name on a page, use {!account.name} in the page markup:   |  |  |  | | --- | --- | --- | | 1 | <apex:page standardController="Account"> | | | 2 | Hello {!$User.FirstName}! |  |  |  |  | | --- | --- | --- | | 3 | <p>You are viewing the {!account.name} account.</p> | | | 4 | </apex:page>  **Using the Visualforce Component Library**  Up to this point, the only Visualforce tag that has been used in the examples is the mandatory <apex:page> tag that must be placed at the start and end of all Visualforce markup. However, just as you can insert images or tables into an HTML document with the <img> or <table> tags, respectively, you can add user interface components to your Visualforce pages using tags that are defined in the Visualforce component library.  For example, to add a component that looks like a section on a detail page, use the <apex:pageBlock> component tag:   |  |  | | --- | --- | | 1 | <apex:page standardController="Account"> | | 2 | <apex:pageBlock title="Hello {!$User.FirstName}!"> | |  |  |  |  | | --- | --- | --- | | 3 | You are viewing the {!account.name} account. | | | 4 | </apex:pageBlock> |  |  |  | | --- | --- | | 5 | </apex:page> | | |

**apex:dataTable**

An HTML table that’s defined by iterating over a set of data, displaying information about one item of data per row. The body of the <apex:dataTable> contains one or more column components that specify what information should be displayed for each item of data. The data set can include up to 1,000 items, or 10,000 items when the page is executed in read-only mode.

For Visualforce pages running Salesforce.com API version 20.0 or higher, an <apex:repeat> tag can be contained within this component to generate columns.

See also: <apex:panelGrid>

This component supports HTML pass-through attributes using the "html-" prefix. Pass-through attributes are attached to the generated table's <tbody> tag.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | <!-- For this example to render fully, associate the page |  |  |  | | --- | --- | | 02 | with a valid account record in the URL. |  |  |  | | --- | --- | | 03 | For example: https://Salesforce\_instance/apex/myPage?id=001D000000IRt53 --> |  |  |  | | --- | --- | | 04 |  |  |  |  | | --- | --- | | 05 | <!-- Page: --> |  |  |  | | --- | --- | | 06 |  |  |  |  | | --- | --- | | 07 | <apex:page controller="dataTableCon" id="thePage"> |  |  |  | | --- | --- | | 08 | <apex:dataTable value="{!accounts}" var="account" id="theTable" |  |  |  | | --- | --- | | 09 | rowClasses="odd,even" styleClass="tableClass"> |  |  |  | | --- | --- | | 10 | <apex:facet name="caption">table caption</apex:facet> |  |  |  | | --- | --- | | 11 | <apex:facet name="header">table header</apex:facet> |  |  |  | | --- | --- | | 12 | <apex:facet name="footer">table footer</apex:facet> |  |  |  | | --- | --- | | 13 |  |  |  |  | | --- | --- | | 14 | <apex:column> |  |  |  | | --- | --- | | 15 | <apex:facet name="header">Name</apex:facet> |  |  |  | | --- | --- | | 16 | <apex:facet name="footer">column footer</apex:facet> |  |  |  | | --- | --- | | 17 | <apex:outputText value="{!account.name}"/> |  |  |  | | --- | --- | | 18 | </apex:column> |  |  |  | | --- | --- | | 19 |  |  |  |  | | --- | --- | | 20 | <apex:column> |  |  |  | | --- | --- | | 21 | <apex:facet name="header">Owner</apex:facet> |  |  |  | | --- | --- | | 22 | <apex:facet name="footer">column footer</apex:facet> |  |  |  | | --- | --- | | 23 | <apex:outputText value="{!account.owner.name}"/> |  |  |  | | --- | --- | | 24 | </apex:column> |  |  |  | | --- | --- | | 25 |  |  |  |  | | --- | --- | | 26 | </apex:dataTable> |  |  |  | | --- | --- | | 27 | </apex:page> |  |  |  | | --- | --- | | 28 |  |  |  |  | | --- | --- | | 29 | /\*\*\* Controller: \*\*\*/ |  |  |  | | --- | --- | | 30 |  |  |  |  | | --- | --- | | 31 | public class dataTableCon { |  |  |  | | --- | --- | | 32 |  |  |  |  | | --- | --- | | 33 | List<Account> accounts; |  |  |  | | --- | --- | | 34 |  |  |  |  | | --- | --- | | 35 | public List<Account> getAccounts() { |  |  |  | | --- | --- | | 36 | if(accounts == null) |  |  |  | | --- | --- | | 37 | accounts = [SELECT name, owner.name FROM account LIMIT 10]; |  |  |  | | --- | --- | | 38 | return accounts; |  |  |  | | --- | --- | | 39 | } |  |  |  | | --- | --- | | 40 |  |  |  |  | | --- | --- | | 41 | } | |

The example above renders the following HTML:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | <table class="tableClass" id="thePage:theTable" border="0" cellpadding="0" cellspacing="0"> |  |  |  | | --- | --- | | 02 |  |  |  |  | | --- | --- | | 03 | <colgroup span="2"></colgroup> |  |  |  | | --- | --- | | 04 | <caption>table caption</caption> |  |  |  | | --- | --- | | 05 | <thead> |  |  |  | | --- | --- | | 06 | <tr> |  |  |  | | --- | --- | | 07 | <td colspan="2" scope="colgroup">table header</td> |  |  |  | | --- | --- | | 08 | </tr> |  |  |  | | --- | --- | | 09 |  |  |  |  | | --- | --- | | 10 | <tr> |  |  |  | | --- | --- | | 11 | <td scope="col">Name</td> |  |  |  | | --- | --- | | 12 | <td scope="col">Owner</td> |  |  |  | | --- | --- | | 13 | </tr> |  |  |  | | --- | --- | | 14 | </thead> |  |  |  | | --- | --- | | 15 |  |  |  |  | | --- | --- | | 16 | <tfoot> |  |  |  | | --- | --- | | 17 | <tr> |  |  |  | | --- | --- | | 18 | <td scope="col">column footer</td> |  |  |  | | --- | --- | | 19 | <td scope="col">column footer</td> |  |  |  | | --- | --- | | 20 | </tr> |  |  |  | | --- | --- | | 21 | <tr> |  |  |  | | --- | --- | | 22 | <td colspan="2" scope="colgroup">table footer</td> |  |  |  | | --- | --- | | 23 | </tr> |  |  |  | | --- | --- | | 24 | </tfoot> |  |  |  | | --- | --- | | 25 |  |  |  |  | | --- | --- | | 26 | <tbody> |  |  |  | | --- | --- | | 27 | <tr class="odd"> |  |  |  | | --- | --- | | 28 | <td>Bass Manufacturing</td> |  |  |  | | --- | --- | | 29 | <td>Doug Chapman</td> |  |  |  | | --- | --- | | 30 | </tr> |  |  |  | | --- | --- | | 31 |  |  |  |  | | --- | --- | | 32 | <tr class="even"> |  |  |  | | --- | --- | | 33 | <td>Ball Corp</td> |  |  |  | | --- | --- | | 34 | <td>Alan Ball</td> |  |  |  | | --- | --- | | 35 | </tr> |  |  |  | | --- | --- | | 36 |  |  |  |  | | --- | --- | | 37 | <tr class="odd"> |  |  |  | | --- | --- | | 38 | <td>Wessler Co.</td> |  |  |  | | --- | --- | | 39 | <td>Jill Wessler</td> |  |  |  | | --- | --- | | 40 | </tr> |  |  |  | | --- | --- | | 41 | </tbody> |  |  |  | | --- | --- | | 42 |  |  |  |  | | --- | --- | | 43 | </table> | | Adding and Customizing Input Field Labels When used inside of a <apex:pageBlockSection> component, Visualforce input components and some output components automatically display a form label for the field. For components that map to standard or custom object fields, the displayed label is the object field label by default. To override the default value, and for components that aren’t mapped directly to object fields, you can set the label using the label attribute of the component. For example:   |  |  |  | | --- | --- | --- | | 01 | <apex:page standardController="Contact"> | | | 02 | <apex:form> |  |  |  | | --- | --- | | 03 | <apex:pageBlock title="Quick Edit: {!Contact.Name}"> | | 04 | <apex:pageBlockSection title="Contact Details" columns="1"> | |  |  |  | | --- | --- | | 05 | <apex:inputField value="{!Contact.Phone}"/> | | 06 | <apex:outputField value="{!Contact.MobilePhone}" | |  |  |  | | --- | --- | | 07 | label="Mobile #"/> | | 08 | <apex:inputText value="{!Contact.Email}" | |  |  |  |  | | --- | --- | --- | | 09 | label="{!Contact.FirstName + '’s Email'}"/> | | | 10 | </apex:pageBlockSection> |  |  |  | | --- | --- | | 11 | <apex:pageBlockButtons > | | 12 | <apex:commandButton action="{!save}" value="Save"/> | |  |  |  |  | | --- | --- | --- | | 13 | </apex:pageBlockButtons> | | | 14 | </apex:pageBlock> |  |  |  |  | | --- | --- | --- | | 15 | </apex:form> | | | 16 | </apex:page> |   **Using the label attribute to customize form field labels.**  Note  For this page to display contact data, the ID of a valid contact record must be specified as a query parameter in the URL for the page. For example,   |  |  | | --- | --- | | 1 | https://Salesforce\_instance/apex/myPage?id=003D000000Q513R |   Displaying Field Values with Visualforce has more information about retrieving the ID of a record.  The label attribute may be a string, or an expression that evaluates to a string. If you set label to an empty string, the form label for that field will be suppressed.  The label attribute can be set on the following Visualforce components:   * <apex:inputCheckbox> * <apex:inputField> * <apex:inputSecret> * <apex:inputText> * <apex:inputTextarea> * <apex:outputField> * <apex:outputText> * <apex:selectCheckboxes> * <apex:selectList> * <apex:selectRadio> |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Attributes  | **Attribute Name** | **Attribute Type** | **Description** | **Required?** | **API Version** | **Access** | | --- | --- | --- | --- | --- | --- | | align | String | The position of the rendered HTML table with respect to the page. Possible values include "left", "center", or "right". If left unspecified, this value defaults to "left". |  | 10.0 | global | | bgcolor | String | The background color of the rendered HTML table. |  | 10.0 | global | | border | String | The width of the frame around the rendered HTML table, in pixels. |  | 10.0 | global | | captionClass | String | The style class used to display the caption for the rendered HTML table, if a caption facet is specified. This attribute is used primarily to designate which CSS styles are applied when using an external CSS stylesheet. |  | 10.0 | global | | captionStyle | String | The style used to display the caption for the rendered HTML table, if a caption facet is specified. This attribute is used primarily for adding inline CSS styles. |  | 10.0 | global | | cellpadding | String | The amount of space between the border of each table cell and its contents. If the value of this attribute is a pixel length, all four margins are this distance from the contents. If the value of the attribute is a percentage length, the top and bottom margins are equally separated from the content based on a percentage of the available vertical space, and the left and right margins are equally separated from the content based on a percentage of the available horizontal space. |  | 10.0 | global | | cellspacing | String | The amount of space between the border of each table cell and the border of the other cells surrounding it and/or the table's edge. This value must be specified in pixels or percentage. |  | 10.0 | global | | columnClasses | String | A comma-separated list of one or more classes associated with the table's columns, used primarily to designate which CSS styles are applied when using an external CSS stylesheet.  If more than one class is specified, the classes are applied in a repeating fashion to all columns. For example, if you specify columnClasses="classA, classB", then the first column is styled with classA, the second column is styled with classB, the third column is styled with classA, the fourth column is styled with classB, and so on. |  | 10.0 | global | | columns | Integer | The number of columns in this table. |  | 10.0 | global | | columnsWidth | String | A comma-separated list of the widths applied to each table column. Values can be expressed as pixels (for example, columnsWidth="100px, 100px"). |  | 10.0 | global | | dir | String | The direction in which the generated HTML component should be read. Possible values include "RTL" (right to left) or "LTR" (left to right). |  | 10.0 | global | | first | Integer | The first element in the iteration visibly rendered in the table, where 0 is the index of the first element in the set of data specified by the value attribute. For example, if you did not want to display the first two elements in the set of records specified by the value attribute, set first="2". |  | 10.0 | global | | footerClass | String | The style class used to display the footer (bottom row) for the rendered HTML table, if a footer facet is specified. This attribute is used primarily to designate which CSS styles are applied when using an external CSS stylesheet. |  | 10.0 | global | | frame | String | The borders drawn for this table. Possible values include "none", "above", "below", "hsides", "vsides", "lhs", "rhs", "box", and "border". If not specified, this value defaults to "border". |  | 10.0 | global | | headerClass | String | The style class used to display the header for the rendered HTML table, if a header facet is specified. This attribute is used primarily to designate which CSS styles are applied when using an external CSS stylesheet. |  | 10.0 | global | | id | String | An identifier that allows the dataTable component to be referenced by other components in the page. |  | 10.0 | global | | lang | String | The base language for the generated HTML output, for example, "en" or "en-US". For more information on this attribute, see the W3C specifications. |  | 10.0 | global | | onclick | String | The JavaScript invoked if the onclick event occurs--that is, if the user clicks the data table. |  | 10.0 | global | | ondblclick | String | The JavaScript invoked if the ondblclick event occurs--that is, if the user clicks the data table twice. |  | 10.0 | global | | onkeydown | String | The JavaScript invoked if the onkeydown event occurs--that is, if the user presses a keyboard key. |  | 10.0 | global | | onkeypress | String | The JavaScript invoked if the onkeypress event occurs--that is, if the user presses or holds down a keyboard key. |  | 10.0 | global | | onkeyup | String | The JavaScript invoked if the onkeyup event occurs--that is, if the user releases a keyboard key. |  | 10.0 | global | | onmousedown | String | The JavaScript invoked if the onmousedown event occurs--that is, if the user clicks a mouse button. |  | 10.0 | global | | onmousemove | String | The JavaScript invoked if the onmousemove event occurs--that is, if the user moves the mouse pointer. |  | 10.0 | global | | onmouseout | String | The JavaScript invoked if the onmouseout event occurs--that is, if the user moves the mouse pointer away from the data table. |  | 10.0 | global | | onmouseover | String | The JavaScript invoked if the onmouseover event occurs--that is, if the user moves the mouse pointer over the data table. |  | 10.0 | global | | onmouseup | String | The JavaScript invoked if the onmouseup event occurs--that is, if the user releases the mouse button. |  | 10.0 | global | | onRowClick | String | The JavaScript invoked if the onRowClick event occurs--that is, if the user clicks a row in the data table. |  | 10.0 | global | | onRowDblClick | String | The JavaScript invoked if the onRowDblClick event occurs--that is, if the user clicks a row in the data table twice. |  | 10.0 | global | | onRowMouseDown | String | The JavaScript invoked if the onRowMouseDown event occurs--that is, if the user clicks a mouse button in a row of the data table. |  | 10.0 | global | | onRowMouseMove | String | The JavaScript invoked if the onRowMouseMove event occurs--that is, if the user moves the mouse pointer over a row of the data table. |  | 10.0 | global | | onRowMouseOut | String | The JavaScript invoked if the onRowMouseOut event occurs--that is, if the user moves the mouse pointer away from a row in the data table. |  | 10.0 | global | | onRowMouseOver | String | The JavaScript invoked if the onRowMouseOver event occurs--that is, if the user moves the mouse pointer over a row in the data table. |  | 10.0 | global | | onRowMouseUp | String | The JavaScript invoked if the onRowMouseUp event occurs--that is, if the user releases the mouse button over a row in the data table. |  | 10.0 | global | | rendered | Boolean | A Boolean value that specifies whether the component is rendered on the page. If not specified, this value defaults to true. |  | 10.0 | global | | rowClasses | String | A comma-separated list of one or more classes associated with the table's rows, used primarily to designate which CSS styles are applied when using an external CSS stylesheet.  If more than one class is specified, the classes are applied in a repeating fashion to all rows. For example, if you specify columnRows="classA, classB", then the first row is styled with classA, the second row is styled with classB, the third row is styled with classA, the fourth row is styled with classB, and so on. |  | 10.0 | global | | rows | Integer | The number of rows in this table. |  | 10.0 | global | | rules | String | The borders drawn between cells in the table. Possible values include "none", "groups", "rows", "cols", and "all". If not specified, this value defaults to "none". |  | 10.0 | global | | style | String | The style used to display the dataTable component, used primarily for adding inline CSS styles. |  | 10.0 | global | | styleClass | String | The style class used to display the dataTable component, used primarily to designate which CSS styles are applied when using an external CSS stylesheet. |  | 10.0 | global | | summary | String | A summary of the table's purpose and structure for Section 508 compliance. |  | 10.0 | global | | title | String | The text to display as a tooltip when the user's mouse pointer hovers over this component. |  | 10.0 | global | | value | Object | The collection of data displayed in the table. | Yes | 10.0 | global | | var | String | The name of the variable that represents one element in the collection of data specified by the value attribute. You can then use this variable to display the element itself in the body of the dataTable component tag. | Yes | 10.0 | global | | width | String | The width of the entire table, expressed either as a relative percentage to the total amount of available horizontal space (for example, width="80%"), or as the number of pixels (for example, width="800px"). |  | 10.0 | global |  Facets  | **Facet Name** | **Description** | **API Version** | | --- | --- | --- | | caption | The components that appear in the caption for the table. Note that the order in which a caption facet appears in the body of a dataTable component doesn’t matter, because any facet with name="caption" will control the appearance of the table's caption. | 10.0 | | footer | The components that appear in the footer row for the table. Note that the order in which a footer facet appears in the body of a dataTable component doesn’t matter, because any facet with name="footer" will control the appearance of the final row in the table. | 10.0 | | header | The components that appear in the header row for the table. Note that the order in which a header facet appears in the body of a dataTable component doesn’t matter, because any facet with name="header" will control the appearance of the first row in the table. | 10.0 | |