Pagination.cfc Documentation

Pagination.cfc open source project
Nathan Strutz
http://www.dopefly.com/

Index

1. What Is Pagination?

Pagination is the practice of splitting your data into multiple pages and allowing navigation both to, and between those pages. It is sometimes called NextN Navigation.

Example 1, you may have 300 products to display on a page. Displaying them all would be unreasonable, however, displaying 10 or 20 at a time makes sense. This is where you would bring in pagination.

Example 2, you may be creating a mail application. Pagination could be easily used in 2 places here. First, to limit the number of message subjects displayed on the message list page. Second, to handle next / previous controls when viewing an individual message.

2. Why use Pagination.cfc?

Years ago, when creating my first pagination routine, I was surprised at how many lines of code it took, even for an instance specific, non-customizable, single-use solution. I was making a messaging system and needed to limit the number of records shown, so I created 2 "NextN" navigation systems to handle pagination. Years later, I used my hindsight to spot the problem: too many lines of code for a non-reusable solution. Something needed to be done.

Pagination.cfc offers a customizable, reusable pagination solution that won't make you write too many lines of untested code. Pagination.cfc offers an easy, friendly solution. Open

source means higher quality, and Pagination.cfc has undergone unit tests and peer reviews.

3. How to use Pagination.cfc

There are 6 steps to using Pagination.cfc

1. Copy Pagination.cfc into your application
2. Instantiate the Pagination component
3. Give Pagination.cfc the data to be paginated
4. Set your other output preferences
5. Output the generated HTML from Pagination.cfc
6. Use the calculated properties in Patination.cfc in other places

1. Copy Pagination.cfc into your application

I suggest putting it where your other components (CFCs) live, more appropriately, in a utilities package. If your components don't have a home, or if you don't use any, you can put them in a /components/ folder off of your application root. It really doesn't matter, though.

2. Instantiate the Pagination component

Much like you would do with any other component. I prefer using the createObject function.

```
<cfset pagination = createObject("component", "components.Pagination").init()
/>
```

3. Give Pagination.cfc the data to be paginated

Pagination.cfc can paginate queries, arrays or structures. In this way, nothing stops you from paginating over structs of arrays or arrays of structs, etc.

```
<cfset pagination.setQueryToPaginate(myQuery) />
```
or
```
<cfset pagination.setArrayToPaginate(myArray) />
```

This is an important step! Without it, Pagination.cfc will throw an error when you ask it to render the output HTML.

4. Set your other output preferences

It is recommended that you set the BaseLink property (the link to the current data view), especially if it could change, such as from a create/update/delete command on the data that then returns to the view with a different URI. This should be the URI without the pagination link. Pagination.cfc will discover the base link for itself if one is not provided.

```
<cfset pagination.setBaseLink("/app/photolist.cfm?year=2007") />
```

Another important property you may want to tweak is the ItemsPerPage property. A single message view will only want 1 item per page, while for a large list, you may want 100. The default value is 10.

```
<cfset pagination.setItemsPerPage(25) />
```

Another important property is UrlPageIndicator, which tells Pagination.cfc what URL variable to use for managing the paging index. The default value is "pagenumber" and you can change it.

```
<cfset pagination.setUrlPageIndicator("page") />
```

The final important property for this abbreviated list is ShowNumericLinks. This boolean property turns on and off the display of numbered pages. When ShowNumericLinks is *false*, which is the default, the rendered HTML will output previous and next controls but no numbers. When it is set to *true*, the numbering is displayed. The numbers that are displayed depends on the other options, such as the NumericDistanceFromCurrentPageVisible and NumericEndBufferCount properties.

```
<cfset pagination.setShowNumericLinks(true) />
```

Please see the reference in the next section of this document for a list of all the properties you can use to customize and tweak the pagination display.

5. Output the generated HTML from Pagination.cfc.

```
#pagination.getRenderedHTML()#
```

Pagination.cfc caches the rendered output, so subsequent calls will cause no performance decrease. The cache is reset if you change a property such as the number of items per page.

6. Use the calculated properties in Patination.cfc in other places

Pagination.cfc will calculate some fields you may need such as TotalNumberOfPages, CurrentPage, StartRow and EndRow. These are discussed in more detail in the reference section at the end of this document.

<cfoutput query="myQuery" startrow="#pagination.getStartRow()#"
maxrows="#pagination.getMaxRows()#">

The final block of code, using the above options, will look like this.

```
<cfset pagination = createObject("component", "components.Pagination").init()
/>
<cfset pagination.setQueryToPaginate(myQuery) />
<cfset pagination.setBaseLink("/app/photolist.cfm?year=2007") />
<cfset pagination.setItemsPerPage(25) />
<cfset pagination.setUrlPageIndicator("page") />
<cfset pagination.setShowNumericLinks(true) />
```

```
#pagination.getRenderedHTML()#
<cfoutput query="myQuery" startrow="#pagination.getStartRow()#"
maxrows="#pagination.getMaxRows()#">
   <li>#id# - #name#</li>
</cfoutput>
#pagination.getRenderedHTML()#
```

4. Pagination.cfc Property Reference

All of these properties are retrievable by use of getter methods. Almost all of them are transmutable (you can change them) via setter methods. As shown in the previous examples, you will call get*Property*() and set*Property*(*Value*) where *Property* is one of the below listed properties and *Value* is a valid type value for the given property.

Some types have aliases, which are sometimes friendlier alternatives to the given name. For example, you can set the ItemsPerPage value, but within a cfquery tag, it makes more sense to call it MaxRows (see the example in section 3). You can use these aliases interchangeably.

The CFC itself documents these properties with hints, but you can use this list as an offline reference.

Essential Properties

QueryToPaginate (query, alias:Query)
The query object to paginate. QueryToPaginate, ArrayToPaginate or StructToPaginate are the only required fields; one of these must be set.

ArrayToPaginate (array, alias:Array)
An array object to paginate instead of a query. QueryToPaginate, ArrayToPaginate or StructToPaginate are the only required fields; one of these must be set.

StructToPaginate (struct, alias:Struct)
A struct to paginate over instead of a query or an array. Will create its own ordered key list if one is not provided (as the 2nd argument). QueryToPaginate, ArrayToPaginate or StructToPaginate are the only required fields; one of these must be set.

ItemsPerPage (numeric; alias:MaxRows)
Number of items to display per page.

BaseLink (string, default:calculated)
The link to the current page. If one is not given, Pagination.cfc will guess based on the current URL. This can have bad side-effects if there is more than one URI that accesses the same data, such as an update or delete command to a list.

UrlPageIndicator (string, default:pagenumber)
The URL variable used to track the current page number.

CompressHTML (boolean, default:false)
Option to remove whitespace from the generated HTML and combine to one line. It is

possible, though unlikely, that this can cause formatting issues if you rely on whitespace for character spacing, etc.

Number Related Properties

ShowNumericLinks (boolean, default:false)
Option to display linked pagination numbers. Without, you will see something like:
< Previous Next >
With this option enabled, you will see something like:
<Previous **1** 2 3 Next >

NumericDistanceFromCurrentPageVisible (numeric, default:3)
The count of numeric links to display on either side of the currently selected page. A count of 1 may display:
1 2 ... 28 **29** 30 ... 49 50
A count of 4 may display:
1 2 ... 25 26 27 28 **29** 30 31 32 33 ... 49 50

NumericEndBufferCount (numeric, default:2)
The count of end "buffer" numbers for either side. A count of 1 may display:
1 ... 27 28 **29** 30 31 ... 50
A count of 5 may display:
1 2 3 4 5 ... 27 28 **29** 30 31 ... 46 47 48 49 50

ShowMissingNumbersHTML (boolean, default:true)
Option to display HTML between numbers when one or more numbers are skipped. This is the same as setting the MissingNumbersHTML value to "".

MissingNumbersHTML (string, default:"...")
HTML to place where numbers are skipped. The default value will display:
1 2 3 ... 10
Changing the value to "_{skip a few}" will display:
1 2 3 skip a few 10

BeforeNumericLinksHTML (string)
HTML to place between any Previous link and the numeric links (if numeric links are enabled).

BeforeNextLinkHTML (string)
HTML to place between numeric links and any Next link

Next/Previous Control Related Properties

ShowPrevNextHTML (boolean, default:true)
Option to display Previous and Next controls. If you turn this off, please turn on ShowNumericLinks so the numeric navigation is displayed.

PreviousLinkHTML (string, default:"< Previous")
HTML to display for the previous page link. The link to the previous page will automatically be placed around this HTML. Remember to use HTML entities to escape illegal characters such as < and >.

NextLinkHTML (string, default:"Next >")

HTML to display for the next page link. The link to the next page will automatically be placed around this HTML. Remember to use HTML entities to escape illegal characters such as < and >.

ShowPrevNextDisabledHTML (boolean, default:true)
Option to display previous and next links while on the first and last pages, respectively.

PreviousLinkDisabledHTML (string, default:"< Previous")
HTML to display for the previous page link when the link is not active because you are on the first page. Remember to use HTML entities to escape illegal characters such as < and >.

NextLinkDisabledHTML (string, default:"Next >")
HTML to display for the next page link when the link is not active because you are on the last page. Remember to use HTML entities to escape illegal characters such as < and >.

ShowFirstLastHTML (boolean, default:false)
Option to display link controls to jump to the first and last pages.

FirstLinkHTML (string, default:"<< First")
HTML to display for the first page link. The link to the first page will automatically be placed around this HTML. Remember to use HTML entities to escape illegal characters such as < and >.

LastLinkHTML (string, default:"Last >>")
HTML to display for the last page link. The link to the last page will automatically be placed around this HTML. Remember to use HTML entities to escape illegal characters such as < and >.

ShowFirstLastDisabledHTML (boolean, default:false)
Option to display first and last link controls while on the first and last pages, respectively. If enabled, the FirstLinkDisabledHTML and LastLinkDisabledHTML will be used

FirstLinkDisabledHTML (string, default:"<< First")
HTML to display for the first page link placeholder when the link is not active because you are on the first page. Remember to use HTML entities to escape illegal characters such as < and >.

LastLinkDisabledHTML (string, default:"Last >>")
HTML to display for the last page link placeholder when the link is not active because you are on the last page. Remember to use HTML entities to escape illegal characters such as < and >.

Calculated Properties

These properties do not have public setters. Do not attempt to set these properties, but please use them.

TotalNumberOfPages (numeric, default:calculated)
The total number of pages to be paginated over. This is calculated based upon the total number of items to display and the number of items to display per page.

CurrentPage (numeric, default:calculated)
The page the user is currently viewing. This is taken from the URL page number variable.

Please call this instead of #url.pagenumber#, as the URL variable name can be changed.

StartRow (numeric, default:calculated)
The starting row for the current page. Use this in your output loop to set the first row to display on the page. This should match up with you cfoutput startrow attribute.

EndRow (numeric, default:calculated)
The last row displayed on the current page. This property is for your convenience.

TotalNumberOfItems (numeric, default:calculated)
Number of items in your data that will be paginated over.

FirstPageLink (string)
Link to the first page.

PreviousPageLink (string)
Link to the previous sequential page.

NextPageLink (string)
Link to the next sequential page.

LastPageLink (string)
Link to the last page.

RenderedHTML (string)
The calculated and rendered HTML output. Call for this to retrieve your pagination four outputting directly into your page.

5. Pagination.cfc Styling Reference

Use stylesheets is the way to make Pagination.cfc's output look the way you want it to. As such, I have created this quick reference.

Here is an outline of the HTML for CSS reference:

```
<div class="pagination custom"><!-- can include your own custom class -->
    <a class="first">First Page</a> OR <span class="first">Disabled First
page Link</span>
    <a class="previous">Previous Page</a> OR <span class="first">Disabled
First page Link</span>
        <a>1</a>
        <span class="current">2</span>
        <a>3</a>
    <a class="next">Next Page</a> OR <span class="next">Disabled First page
Link</span>
    <a class="last">Last Page</a> OR <span class="last">Disabled First page
Link</span>
</div>
```

Now, here is some basic CSS I recommend to get you started styling your pagination output:

```
div.pagination a { border:1px solid blue; text-decoration:none; padding:4px; marign:0 4px; }
div.pagination span { marign:0 4px; }
```

This will surround linked elements with a nice box and space them out a bit.

Also, please note the numerous styled demo examples included with the download where you can see both execution and styling examples in action.

6. Extending Pagination.cfc

In order to provide deeper customization to your application, you may want to extend pagination.cfc. As a large number of ColdFusion developers don't know what this involves, I will provide an example.

Let's say that you want your pagination to output in this format:

<< < Page 7 of 9 > >>

There are two ways to do it. First, you can write the HTML on your .cfm page like this:

```
<cfset pagination = createObject("component", "components.Pagination").init()
/>
<cfset pagination.setQueryToPaginate(myQuery) />

<a href="#pagination.getFirstPageLink()#">&lt;&lt;</a>
<a href="#pagination.getPreviousPageLink()#">&lt;&lt;</a>
Page #pagination.getCurrentPage()# of #pagination.getTotalNumberOfPages()#
<a href="#pagination.getNextPageLink()#">&lt;&lt;</a>
<a href="#pagination.getLastPageLink()#">&lt;&lt;</a>
```

This solution is not reusable and not very well separated from the rest of your application. It adds a lot of messy code to the page. Instead, consider extending Pagination.cfc with a reusable solution. Here is Pagination_Custom.cfc:

```
<cfcomponent extends="Pagination">
  <cffunction name="renderHTML">
    <cfset var renderedOutput="" />
    <cfsavecontent variable="renderedOutput">
      <cfoutput>
        <a href="#getFirstPageLink()#">&lt;&lt;</a>
        <a href="#getPreviousPageLink()#">&lt;</a>
        Page #getCurrentPage()# of #getTotalNumberOfPages()#
        <a href="#getNextPageLink()#">&gt;</a>
        <a href="#getLastPageLink()#">&gt;&gt;</a>
```

```
      </cfoutput>
    </cfsavecontent>
    <cfreturn renderedOutput />
  </cffunction>
</cfcomponent>
```

This new solution overrides the renderHTML() private method in Pagination.cfc. When you call getRenderedHTML(), it will run this method to render the HTML instead of the default method. This cleans up the code required on your cfm page as well. The following example shows how to use this new solution:

```
<cfset pagination = createObject("component",
"components.Pagination_Custom").init() />
<cfset pagination.setQueryToPaginate(myQuery) />
#pagination.getRenderedHTML()#
```