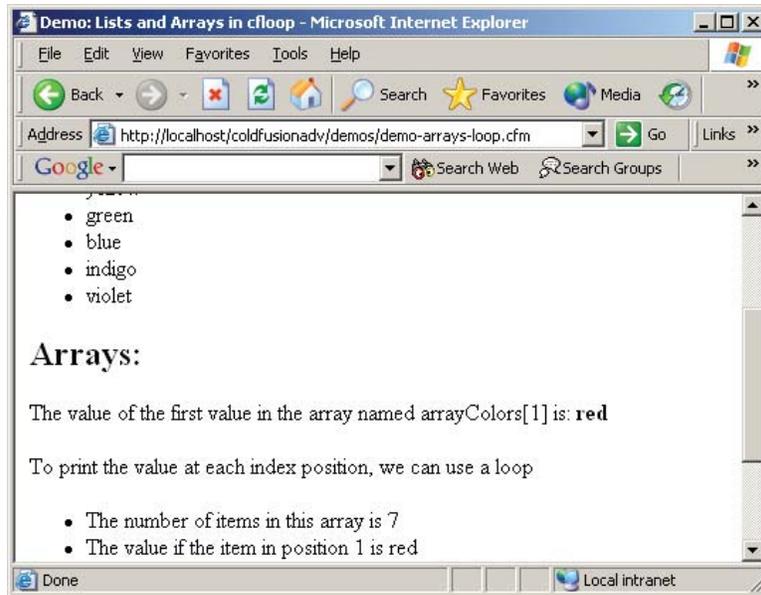


Looping with <cfloop>

In ASP and other server-side web technologies developers must loop through resulting records of a database query to display the results. Since ColdFusion provides the automatic looping of <cfoutput>, <cfloop> is less important, but is still it is a very useful feature. The <cfloop> tag can be used to loop many different ways. This chart defines the most common looping options:

ColdFusion Loop Type	Comparable to	Code Example	Explanation
Index Loop	For Loop	<pre><cfloop index="x" from="1" to="10"> code to execute </cfloop></pre>	An index loop will repeat once for each value of x. X will increment for each value between the value in the from and to attributes. (In our example, it will repeat 10 times.)
Condition Loop	While Loop	<pre><cfloop condition="x LTE 10"> code to execute </cfloop></pre>	A Condition loop will repeat as long as the listed condition is true. In our example, it will repeat as long as x is still less than 10. You must include some way to change the value x within the <cfloop> block or we could get stuck in an infinite loop!
Query Loop	For Each loop (looping though resulting records)	<pre><cfloop query="name"> code to execute </cfloop></pre> <p>Where "name" is the name of a valid <cfquery></p>	The Query loop is very similar to <cfoutput> tags – it loops once for each resulting record of a query. It is useful because it can be used when nested <cfoutput> tags are desired or to use tags that are not allowed in a <cfoutput> block. (For example, <cfinclude> is not allowed within <cfoutput> tags.)
List Loop	For Each loop (looping though a list)	<pre><cfloop index="x" list="110, 120,130,140"> code to execute </cfloop></pre>	The List loop loops through a list that is (1) defined in the list attribute or (2) defined in a variable (see demo below).
Collection Loop	For Each loop (looping though a Collection)	<pre><cfloop collection="URL " item="x"> code to execute </cfloop></pre>	The Collection loop will repeat once for each member of a collection such as URL, FORM, or CGI variables. It can also be used with COM objects.

The `<cfloop>` tag can provide an easy way to loop through each value of a list or an array. In the following demo, we will loop through a list and an array that we saw in the previous section. This demo is saved as **ComplexObjects/demos/demo-arrays-loop.cfm**.



```
<html>
<head>
  <title>Demo: Lists and Arrays in cfloop</title>
</head>

<body>
<cfoutput>

<h2>Lists:</h2>
<cfset listColors = "red,orange,yellow,green,blue,indigo,violet">

<p>The following unordered list shows each value in the list named
listColors</p>
<ul>
<cfloop list="#listColors#" index="current_one">
  <li>#current_one#</li>
</cfloop>
</ul>

<h2>Arrays:</h2>
<cfset arrayColors = ArrayNew(1)>
<cfset arrayColors[1] = "red">
<cfset arrayColors[2] = "orange">
<cfset arrayColors[3] = "yellow">
<cfset arrayColors[4] = "green">
<cfset arrayColors[5] = "blue">
<cfset arrayColors[6] = "indigo">
```

```

<cfset arrayColors[7] = "violet">

<p>The value of the first value in the array named arrayColors[1] is:
<b>#arrayColors[1]#</b></p>
<p>To print the value at each index position, we can use a loop</p>
<ul>
<cfset max=ArrayLen(arrayColors)>
  <li>The number of items in this array is #max#</li>
<cfloop index="x" from="1" to="#max#">
  <li>The value if the item in position #x# is #arrayColors[x]#</li>
</cfloop>
</ul>

</cfoutput>

</body>
</html>

```

The <cfloop> tag can be used to manually produce the same sort of tables that are automatically produced with the <cfdump> tag. For many debugging purposes <cfdump> will be sufficient, however, if you need to control the spacing or the color of the table, <cfloop> gives you ultimate control of the output since you will build it!

The following code shows how to loop through collections. Note the syntax when displaying a FORM, URL or CGI variable (i.e. **FORM[current_one]**). In the previous section we took a quick look at structures. The FORM, URL and CGI Collections are really just structures. You will find the code saved in **ComplexObjects/demos/demo-cfloop-external.cfm**:

```

<html>
<head>
  <title>Demo: Quick List of External Variables</title>
</head>

<body>

<h2>Quick List of External Variables</h2>

<cfoutput>

  <p>Here are the FORM variables (submitted by POST method):</p>
  <table border="1" cellspacing="0" cellpadding="1">
  <cfloop collection="#FORM#" item="current_one">
    <tr>
      <td bgcolor="##aaaaee">#current_one#</td>
      <td>#FORM[current_one]#</td>
    </tr>
  </cfloop>
  </table>

  <hr>

  <p>And the URL variables (links or forms submitted by GET method):</p>

```

```

<table border="1" cellspacing="0" cellpadding="1">
<cfloop collection="#URL#" item="current_one">
  <tr>
    <td bgcolor="##aaaaee">#current_one#</td>
    <td>#URL[current_one]#</td>
  </tr>
</cfloop>
</table>

<hr>

<p>And the CGI variables:</p>
<table border="1" cellspacing="0" cellpadding="1">
<cfloop collection="#CGI#" item="current_one">
  <tr>
    <td bgcolor="##aaaaee">#current_one#</td>
    <td>#CGI[current_one]#</td>
  </tr>
</cfloop>
</table>

</cfoutput>

</body>
</html>

```

To test this page, start with **ComplexObjects/demos/demo-cfloop-form.cfm**, and fill out the form. .

Syntax of <cfloop>

The <cfloop> tag can be used for various types of loops, but here we are looping through a **collection** (as evidenced by the **collection** attribute). The **item** attribute gives us a reusable variable for the **name** in each name/value pair. Here, we arbitrarily chose the variable name **current_one**. This variable name will be used later to refer to the current record.

The other types of <cfloop> loops include **index loops** (like **for** loops in other programming languages), **conditional loops** (like **while** loops in other languages), **list loops** (like **foreach** loops in other languages), and **query loops** (similar to <cfoutput> with a **query** attribute). For documentation about these other types of loops, please consult ColdFusion Studio's built-in help (right-click any <cfloop> tag and choose **Edit Tag...**).

Weeding Out the FIELDNAMES Variable

With both <cfdump> and <cfloop>, you may have noticed that a form submitted via POST method will have one extra field beyond the expected ones: a separate field

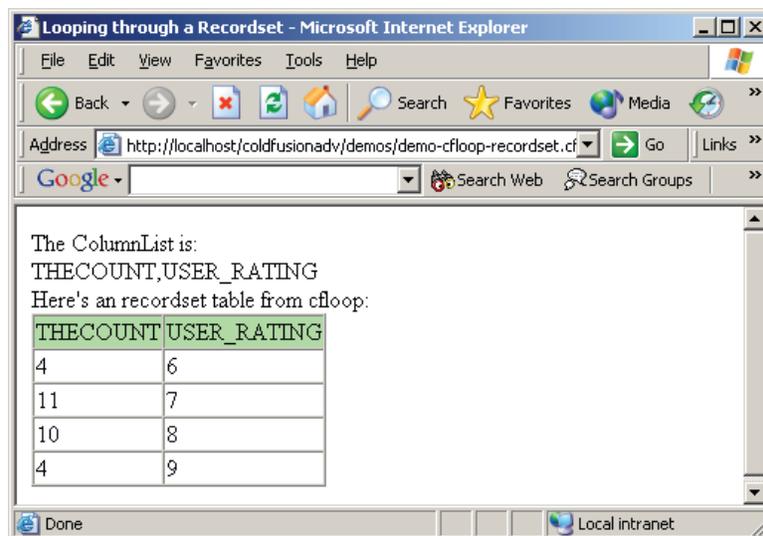
named **FIELDNAMES** is provided as another means of creating loops. If you want to weed out this extra field, simply add a `<cfif>` conditional inside your loop:

```
<cfloop collection="#FORM#" item="current_one">
  <cfif current_one is not "fieldnames">
    <tr>
      <td bgcolor="##aaaaee">#current_one#</td>
      <td>#FORM[current_one]#</td>
    </tr>
  </cfif>
</cfloop>
```

Using `<cfloop>` for automatic query output

If you want to duplicate the same effect we had with `<cfdump>` and a query name, the looping syntax is a little bit different from the collection loops above.

To see this in action, please open **ComplexObjects/demos/demo-cfloop-recordset.cfm** in your browser:



As you may have noticed, it looks almost identical to a `<cfdump>`. Only difference is, since we built it by hand, we have more control over display features (such as color, spacing, fonts, etc.).

Here's the code for that file:

```
<cfquery name="getdata" datasource="movieList">
  SELECT user_rating, count(user_rating) as thecount
  FROM movies
  GROUP BY user_rating
</cfquery>

<html>
```

```

<head>
  <title>Looping through a Recordset</title>
</head>

<body>
The ColumnList is:<br>
<cfoutput>#getdata.ColumnList#</cfoutput><br>

Here's an recordset table from cfloop:
<table border="1" cellspacing="0" cellpadding="1">
  <cfoutput>
    <tr bgcolor="##aaeaaa">
      <cfloop index="current_item" list="#getdata.ColumnList#">
        <td>#current_item#</td>
      </cfloop>
    </tr>
  </cfoutput>

  <cfoutput query="getdata">
    <tr bgcolor="##ffffff">
      <cfloop index="current_item" list="#getdata.ColumnList#">
        <td>#Evaluate(current_item)#</td>
      </cfloop>
    </tr>
  </cfoutput>
</table>

</body>
</html>

```

The first thing to notice here is that our initial `<cfloop>` tag is a **list** loop, not a **collection** loop. In something like a form (using the FORM or URL collection), each name had exactly one value; here we have to look at one table row at a time, and each individual row has its own list of fields & values.

In this table, we want to display two sections (1) a single row with column headings and (2) the resulting data that will repeat once for each record. For the column headings, we can loop through the list named **ColumnList** that we saw in the last section. Remember, it contains a comma-separated list of database column names (similar to what the **FORM.FieldNames** variable does for forms). If we were to print it out on screen for this example, it would contain **THECOUNT,USER_RATING**. Thus, when we plug that into a CFLOOP tag, and pull out the index item at any step along the way, we get the word **THECOUNT** or the word **USER_RATING**. That's exactly what happens in our heading row:

```

<cfloop index="current_item" list="#getdata.ColumnList#">
  <td>#current_item#</td>
</cfloop>

```

Note that the `#current_item#` variable here represents the *word* **USER_RATING**, not the *value* of the **USER_RATING** field. If we want to get the **USER_RATING** interpreted, and it's being handed to us just as a piece of text, we can use the **Evaluate()** function, as seen in the repeating value rows:

```
<cfloop index="current_item" list="#getdata.ColumnList#">
  <td>#Evaluate(current_item)#</td>
</cfloop>
```

This tells ColdFusion to do two things:

1. Figure out the value of the **current_item** variable → we get the word **USER_RATING**.
2. Evaluate **USER_RATING** as a variable → we get a value, such as **8**.

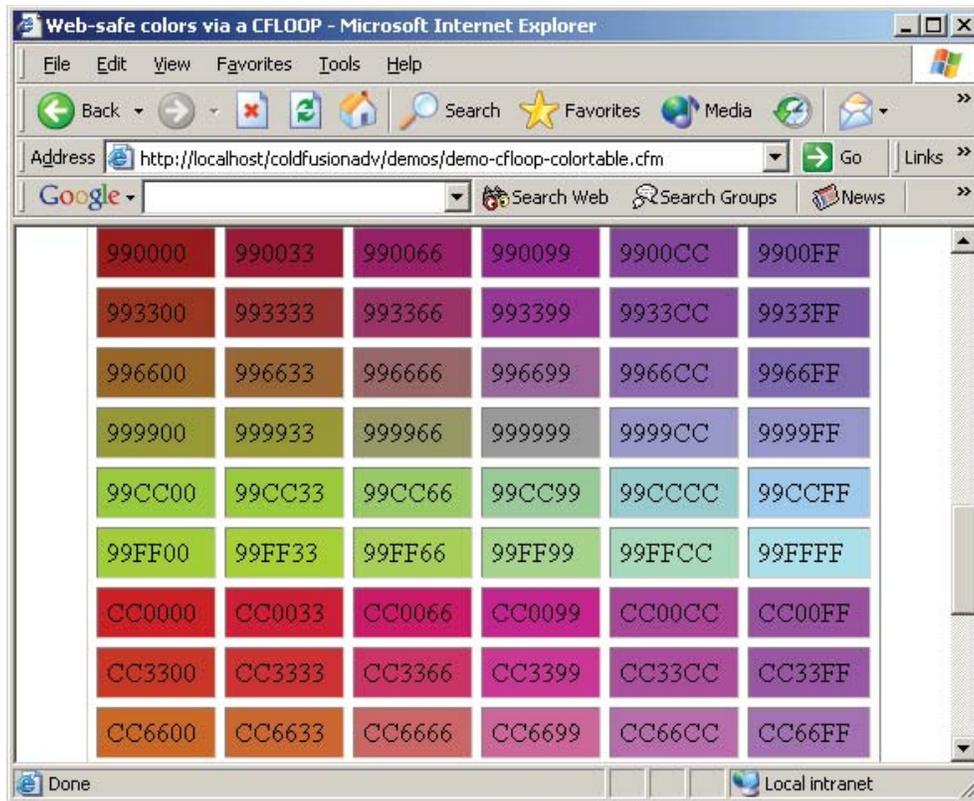
It's also important to note the different structure of the two `<cfoutput>` tags in that example. The first one, for the header rows, does not have a query name. It only needs to run once (one row of headings), and it doesn't have any ambiguous variable names inside. The second one, for the value rows, *does* have a query name. It needs that query name for two reasons:

- These rows have to repeat for all the records.
- The expression `#Evaluate(current_item)#`, which is equivalent to an expression such as `#USER_RATING#`, only makes sense as a column name within the `getdata` query.

Display Web-safe Colors using CFLOOP

While `<cfloop>` can be used for many different purposes, this web-safe color table is created with the use of three nested loops. We have created a list called `colors` that holds the values `00,33,66,99,CC` and `FF`. We will loop through these values using three different loops. For each table cell, we will concatenate together the values of each of the loop's index values.

This demo is saved as **ComplexObjects/demos/demo-cfloop-colortable.cfm** and comes from Ben Forta's *ColdFusion Web Application Construction Kit*.



```

<cfset colors="00,33,66,99,CC,FF">

<h2 align="center">This is a list of the Web-safe colors</h2>

<cfoutput>
<table border="1" cellspacing="5" cellpadding="5">

<cfloop list="#colors#" index="red">
  <cfloop list="#colors#" index="green">
    <tr>
      <cfloop list="#colors#" index="blue">
        <cfset rgb=#red#&#green#&#blue#>
        <td bgcolor="#rgb#">#rgb#</td>
      </cfloop>
    </tr>
  </cfloop>
</cfloop>
</table>
</cfoutput>

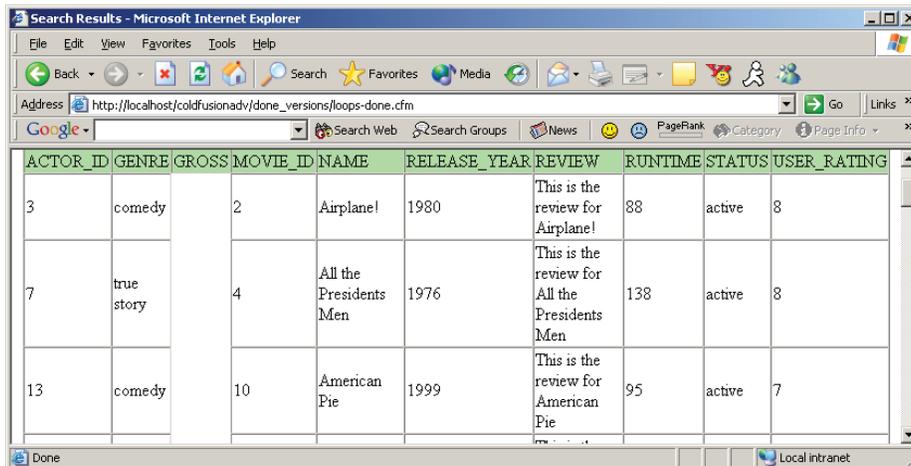
```

Exercise 4: Display results with <cfloop>

20 to 25 minutes

In this exercise you will query the database and then dynamically loop through the results. This will be done without using any of the field names! You will not specify which field is placed in which column. You will simply ask ColdFusion to display all of the fields in the order it decides to use. This could be called a much more complicated version of the <cfdump>.

When completed, it might look something like this:



The screenshot shows a Microsoft Internet Explorer browser window displaying a table of search results. The table has the following columns: ACTOR_ID, GENRE, GROSS, MOVIE_ID, NAME, RELEASE_YEAR, REVIEW, RUNTIME, STATUS, and USER_RATING. The data rows are as follows:

ACTOR_ID	GENRE	GROSS	MOVIE_ID	NAME	RELEASE_YEAR	REVIEW	RUNTIME	STATUS	USER_RATING
3	comedy		2	Airplane!	1980	This is the review for Airplane!	88	active	8
7	true story		4	All the Presidents Men	1976	This is the review for All the Presidents Men	138	active	8
13	comedy		10	American Pie	1999	This is the review for American Pie	95	active	7

1. Open "**ComplexObjects/exercises/loops-temp.cfm**". The code appears below:

```
<cfquery name="getmovies" datasource="movieList" >
SELECT * FROM movies
ORDER BY name
</cfquery>

<html>
<head>
  <title>Search Results</title>
</head>

<body>

<h2 align="center">Search Results</h2>

<p>List of movies:</p>
<table border="1" cellspacing="0" cellpadding="1">
  <!--
  Add a loop here that will display the field names as
  column headings.
  -->
```

```
<!---
Add a second loop here that will display the actual values
of the fields for each record. You will need the evaluate()
function
--->
</table>

</body>
</html>
```

2. Add two loops in place of the two comments
 - a. One will create the column headings
 - b. The second will display the actual value of the fields. *(Remember that you will need to use the Evaluate() function!)*
3. Test your page!

Possible Solution to Exercise 4

Saved as **ComplexObjects/solutions/loops-done.cfm**:

```
<cfquery name="getmovies" datasource="movieList" >
SELECT * FROM movies
ORDER BY name
</cfquery>

<html>
<head>
  <title>Search Results</title>
</head>

<body>

<h2 align="center">Search Results</h2>

<p>List of movies:</p>
<table border="1" cellspacing="0" cellpadding="1">
  <cfoutput>
    <tr bgcolor="##aeeaa">
      <cfloop index="current_item" list="#getmovies.ColumnList#">
        <td>#current_item#</td>
      </cfloop>
    </tr>
  </cfoutput>

  <cfoutput query="getmovies">
    <tr bgcolor="##ffffff">
      <cfloop index="current_item" list="#getmovies.ColumnList#">
        <td>#Evaluate(current_item) #</td>
      </cfloop>
    </tr>
  </cfoutput>
</table>

</body>
</html>
```