EBook Gratis

APRENDIZAJE coldfusion

Free unaffiliated eBook created from
Stack Overflow contributors.

#coldfusion

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: coldfusion

It is an unofficial and free coldfusion ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official coldfusion.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con la fusión en frío

Observaciones

Esta sección proporciona una descripción general de qué es Coldfusion y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema importante dentro de la fusión en frío, y vincular a los temas relacionados. Dado que la Documentación para coldfusion es nueva, es posible que deba crear versiones iniciales de los temas relacionados.

Versiones

Versión	Fecha de lanzamiento
Versión de Cold Fusion 1.0	1995-07-02
Cold Fusion version 1.5	1996-01-01
Versión de Cold Fusion 2.0	1996-10-01
Versión 3.0 de Cold Fusion	1997-06-01
Cold Fusion versión 3.1	1998-01-01
Versión de ColdFusion 4.0	1998-11-01
Versión de ColdFusion 4.5.1	1999-11-01
Versión de ColdFusion 5.0	2001-06-01
ColdFusion MX versión 6.0	2002-05-01
ColdFusion MX versión 6.1	2003-07-01
ColdFusion MX 7	2005-02-07
ColdFusion 8	2007-07-30
ColdFusion 9	2009-10-05
ColdFusion 10	2012-05-15
ColdFusion 11	2014-04-29
ColdFusion 2016	2016-02-16

Examples

Instalación o configuración

Instalación de Linux (Ubuntu)

Lucee (Open Source)

ColdFusion / CFML Interpretor

Descargue el archivo apropiado de su sitio (http://lucee.org/downloads.html) y ejecute su instalador

```
wget http://cdn.lucee.org/downloader.cfm/id/155/file/lucee-5.0.0.252-pl0-linux-x64-
installer.run
sudo chmod +x lucee-5.0.0.252-pl0-linux-x64-installer.run
sudo ./lucee-5.0.0.252-pl0-linux-x64-installer.run
```

Paso a través del instalador.

Nginx

Instala Nginx en tu servidor

```
sudo apt-get install nginx
```

Edite su / etc / nginx / sites-available / default

```
server {
    listen 80;
    server_name _;

    root /opt/lucee/tomcat/webapps/ROOT;
    index index.cfm index.html index.htm;

    #Lucee Admin should always proxy to Lucee
    location /lucee {
        include lucee.conf;
    }

    #Pretty URLs
    location / {
        try_files $uri /index.cfm$uri?$is_args$args;
        include lucee.conf;
    }

    location ~ \.cfm {
        include lucee.conf;
    }
```

```
    location ~ \.cfc {
        include lucee.conf;
    }
}
```

Editar /etc/nginx/lucee.conf

```
proxy_pass http://127.0.0.1:8888;
proxy_set_header Host $http_host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
```

Recargar nginx

```
sudo service nginx reload
```

Acceda al administrador del servidor Lucee aquí:

```
127.0.0.1/lucee/admin/server.cfm
```

o

```
127.0.0.1:8888/lucee/admin/server.cfm
```

Su directorio web raíz vive aquí:

```
/opt/lucee/tomcat/webapps/ROOT
```

Adobe (Fuente cerrada)

ColdFusion / CFML Interpretor

Descargue el archivo apropiado de su sitio (https://www.adobe.com/products/coldfusion/download-trial/try.html) y ejecute su instalador

```
wget <URL>/ColdFusion_2016_WWEJ_linux64.bin
sudo chmod +x ColdFusion_2016_WWEJ_linux64.bin
sudo ./ColdFusion_2016_WWEJ_linux64.bin
```

Paso a través del instalador. Asegúrese de seleccionar el servidor web interno (puerto 8500)

Nginx

Instala Nginx en tu servidor

```
sudo apt-get install nginx
```

Edite su / etc / nginx / sites-available / default

```
server {
    listen 80;
    server_name _;

    root /opt/coldfusion2016/cfusion/wwwroot;
    index index.cfm index.html index.htm;

    location / {
        try_files $uri $uri/ =404;
    }

    location ^~ /CFIDE/administrator {
        deny all;
    }

    location ~* \.(cfm|cfml|cfc|html)$ {
        include /etc/nginx/conf/dc_tomcat_connector.conf;
    }

    location ^~ /rest {
        include tomcatconf;
    }
}
```

Editar /etc/nginx/tomcat.conf

```
proxy_pass http://127.0.0.1:8500;
proxy_set_header Host $host;
proxy_set_header X-Forwarded-Host $host;
proxy_set_header X-Forwarded-Server $host;
proxy_set_header X-Forwarded-For $http_x_forwarded_for;
proxy_set_header X-Real-IP $remote_addr;
```

Recargar nginx

```
sudo service nginx reload
```

Acceda al administrador de Adobe ColdFusion Server aquí:

```
127.0.0.1:8500/CFIDE/administrator/index.cfm
```

Su directorio web raíz vive aquí:

```
/opt/coldfusion2016/cfusion/wwwroot
```

Hola Mundo

Archivo: test.cfm

Implementación de etiquetas

```
<cfoutput>Hello World!</cfoutput>
```

Implementación de CFScript

```
<cfscript>
writeOutput("Hello World!");
</cfscript>
```

Lea Empezando con la fusión en frío en línea:
https://riptutorial.com/es/coldfusion/topic/913/empezando-con-la-fusion-en-frio

Capítulo 2: Alcances en Coldfusion

Introducción

Un **ámbito** es "el rango en el que se puede hacer referencia a una variable". ColdFusion conoce, al igual que la mayoría de los otros lenguajes de programación y script, varios ámbitos. El siguiente texto trata sobre estos tipos y formas de aclararlas, sus diferencias y sus características.

Examples

Solicitar ámbitos

solicitud

variables

formar

url

cgi

Ámbitos globales

Servidor

Solicitud

Sesión

Componentes y funciones

variables

esta

local

argumentos

Etiquetas personalizadas

atributos

esta etiqueta

llamador

Alcances comunes

En la mayoría de los casos, probablemente estés trabajando con estos ámbitos:

- **El ámbito de variables** es el ámbito al que se asignan todas las variables cuando nada más se declara intencionalmente (como el ámbito de `window` en JavaScript).
- **Ámbito del formulario** Cuando envía un formulario a su servidor, todos los campos de formulario que pueden identificarse (mediante el establecimiento de la propiedad name / id) están accesibles en este ámbito para un procesamiento adicional del lado del servidor.
- **Alcance de URL** Todos los parámetros de consulta de url se almacenan en ese alcance
- **Este alcance** Dentro de un componente, `this` refiere al componente mismo.
- **ámbito local** Las variables declaradas dentro de una función que usa la declaración `local` están encapsuladas y solo son accesibles dentro de esa función específica (esto se hace para evitar la contaminación de otros sopes)
- **Ámbitos de argumentos Los** argumentos que se pasan a una función dentro de un componente declarado por la etiqueta `cfargument` son accesibles con ese ámbito

Visión general

- Componentes y funciones
    - variables
    - esta
    - local
    - argumentos
- Etiquetas personalizadas
    - atributos
    - esta etiqueta
    - llamador
- Ámbitos globales
    - Servidor
    - Solicitud
    - Sesión
- Solicitar ámbitos
    - solicitud
    - variables
    - formar
    - url
    - cgi

Lea Alcances en Coldfusion en línea: https://riptutorial.com/es/coldfusion/topic/7864/alcances-en-coldfusion

Capítulo 3: Arreglos ColdFusion

Sintaxis

- ArrayNew (dimension, isSynchronized)

Parámetros

Nombre	Descripción
Dimensión	Número de dimensiones en la nueva matriz: 1, 2 o 3
está sincronizado	Cuando es *falso* , crea una matriz no sincronizada. Cuando es *verdadera* , la función devuelve una matriz sincronizada.

Observaciones

En una matriz sincronizada, más de dos subprocesos no pueden acceder a la matriz simultáneamente. Otros subprocesos tienen que esperar hasta que el subproceso activo complete su trabajo, lo que resulta en un rendimiento significativo.

En la versión ColdFusion 2016, puede usar una matriz no sincronizada y permitir que múltiples hilos accedan al mismo objeto de la matriz simultáneamente.

Examples

Creando Arrays

Creando arrays explícitamente usando ArrayNew ()

Declare una matriz con la función ArrayNew. Especifique el número de dimensiones como un argumento.

- ArrayNew (*dimensión*) crea una matriz de 1–3 dimensiones.
- Las matrices de ColdFusion se expanden dinámicamente a medida que se agregan los datos.
- ArrayNew () devuelve una matriz.

Historia

Introducido en ColdFusion MX 6

Declaración

CFML

```
<!--- One Dimension Array--->
<cfset oneDimensionArray = ArrayNew(1)>
```

CFScript Tenga en cuenta que dentro de una función debe usar `var` scoping. Las versiones anteriores de CF requerían que el ámbito var sea lo primero en una función; Las versiones posteriores lo permiten en cualquier lugar en una función.

```
<cfscript>
    oneDimensionArray = ArrayNew(1);

    public void function myFunc() {
        var oneDimensionArray = ArrayNew(1);
    }
</cfscript>
```

Después de crear la matriz, agregue elementos utilizando los índices de elementos. El índice Coldfusion Array comienza desde 1:

CFML

```
<cfset oneDimensionArray[1] = 1>
<cfset oneDimensionArray[2] = 'one'>
<cfset oneDimensionArray[3] = '1'>
```

CFScript

```
<cfscript>
    oneDimensionArray[1] = 1;
    oneDimensionArray[2] = 'one';
    oneDimensionArray[3] = '1';
</cfscript>
```

Usando ArrayAppend ()

Puede agregar elementos a una matriz utilizando la función `ArrayAppend(array, value)` .

```
<cfscript>
    ArrayAppend(oneDimensionArray, 1);
    ArrayAppend(oneDimensionArray, 'one');
    ArrayAppend(oneDimensionArray, '1');
</cfscript>
```

`<cfdump>` contenido de la matriz usando `<cfdump>` :

```
<cfdump var="#oneDimensionArray#">
```

Resultados:

CFML

```
<!--- Two Dimension Array--->
<cfset twoDimensionArray = ArrayNew(2)>
<cfset twoDimensionArray[1][1] = 1>
<cfset twoDimensionArray[1][2] = 2>
<cfset twoDimensionArray[2][1] = 3>
<cfset twoDimensionArray[2][2] = 4>
```

CFScript

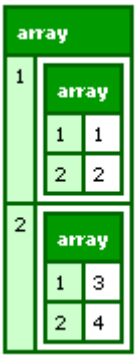
```
<cfscript>
    twoDimensionArray = ArrayNew(2);

    twoDimensionArray[1][1] = 1;
    twoDimensionArray[1][2] = 2;
    twoDimensionArray[2][1] = 3;
    twoDimensionArray[2][2] = 4;
</cfscript>
```

Salida del contenido de la matriz usando `<cfdump>`

```
<cfdump var="#twoDimensionArray#">
```

Resultado:



Cada elemento contiene otra matriz, que almacenará los valores.

Creando Array 1-D implícitamente

Al crear una matriz implícitamente, los corchetes ([]) rodean el contenido de la matriz con separadores de coma.

```
<cfset oneDimensionArrayImplicit = [ 1 ,'one','1' ]>
```

Esta declaración es equivalente a las cuatro declaraciones utilizadas para crear el oneDimensionArray anterior. El resultado es el mismo cuando se usa:

```
<cfdump var="#oneDimensionArrayImplicit#">
```

Crear 2-D Array implícitamente

```
<cfset twoDimensionArrayImplicit = [[ 1 , 2 ],[ 3 , 4 ],[ 5 , 6 ]]>
```

O:

```
<cfset firstElement = ["1", "2"]>
<cfset secondElement= ["3", "4"]>
<cfset thirdElement = ["5", "6"]>
<cfset twoDimensionArrayImplicit = [firstElement , secondElement, thirdElement]>
```

Salida del contenido usando

```
<cfdump var="#twoDimensionArrayImplicit#">
```

Declaración Explicita Alternativa

También puedes declarar 1 Dimension Array como

```
<cfset oneDimensionArray = []>

<cfscript>
    oneDimensionArray = [];
</cfscript>
```

Esta declaración es la misma que la de usar `ArrayNew(1)` .

Pero si intentas declarar 2 Dimension Array como

```
<cfset twoDimensionArray =[][]>    //Invalid CFML construct
```

se producirá un error al procesar esta solicitud.

La siguiente declaración procesará la solicitud:

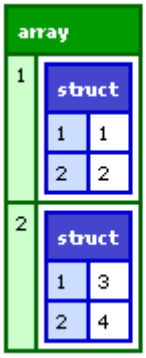
```
<cfset twoDimensionArray =[]>
```

pero la variable `twoDimensionArray` no será realmente una matriz dentro de la matriz (o matriz de 2 dimensiones). En realidad contiene estructura dentro de Array.

```
<cfset twoDimensionArray =[]>
<cfset twoDimensionArray[1][1] = 1>
<cfset twoDimensionArray[1][2] = 2>
<cfset twoDimensionArray[2][1] = 3>
<cfset twoDimensionArray[2][2] = 4>

<cfdump var="#twoDimensionArray#">
```

Resultado:



Array en CFScript

```
<cfscript>
    oneDimensionArray = ArrayNew(1);
    oneDimensionArray[1] = 1;
    oneDimensionArray[2] = 'one';
    oneDimensionArray[3] = '1';
</cfscript>

<cfif IsDefined("oneDimensionArray")>
    <cfdump var="#oneDimensionArray#">
</cfif>
```

Resultado:

Además, podemos declarar una matriz de una dimensión como:

```
oneDimensionArray = [];
```

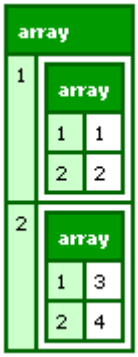
Alternativamente, CF introdujo `WriteDump()` de **CF9** como una función equivalente a la etiqueta `<cfdump>` que se puede usar en `<cfscript>` .

```
<cfscript>
    WriteDump(oneDimensionArray);
</cfscript>
```

Del mismo modo, para 2 Dimension Array:

```
<cfscript>
    twoDimensionArray = ArrayNew(2);
    twoDimensionArray[1][1] = 1;
    twoDimensionArray[1][2] = 2;
    twoDimensionArray[2][1] = 3;
    twoDimensionArray[2][2] = 4;
</cfscript>
<cfdump var="#twoDimensionArray#">
```

Resultado:



Información general

Primero, información general sobre cómo se comportan las matrices en Coldfusion en comparación con otros lenguajes de programación.

- Los arreglos solo pueden tener índices numéricos (si desea que un índice de cadena use `struct` su lugar)
- Las matrices comienzan en el índice [1]
- Las matrices pueden tener una o más *dimensiones.*

Lea Arreglos ColdFusion en línea: https://riptutorial.com/es/coldfusion/topic/6896/arreglos-coldfusion

Capítulo 4: CFLOOP Cómo hacer

Observaciones

Muchas gracias a

- Pete Freitag por su hoja de referencia de CFScript
- Adam Cameron para CF 11: CFLOOP en CFScript está muy roto (y aún lo está en CF 2016).

Examples

Recorrido a través de una colección utilizando etiquetas CFML.

```
<!--- Define collection --->
<cfset attributes = {
  "name": "Sales",
  "type": "text",
  "value": "Connection"
}>

<!---
  cfloop tag with attribute collection can be used to
  loop through the elements of a structure
--->
<cfloop collection=#attributes# item="attribute">
  <cfoutput>
    Key : #attribute#, Value : #attributes[attribute]#
  </cfoutput>
</cfloop>
```

Recorrido a través de una colección usando CFSCRIPT.

```
<cfscript>
  /*define collection*/
  attributes = {
    "name": "Sales",
    "type": "text",
    "value": "Connection"
  };
  for(attribute in attributes){
    /* attribute variable will contain the key name of each key value pair in loop */
    WriteOutput('Key : ' & attribute & ', Value : ' & attributes[attribute] & '<br/>');
  }
</cfscript>
```

Índice

Parámetros

Atributo	Necesario	Tipo	Defecto	Descripción
índice	cierto	cuerda		Nombre de variable para el índice del bucle. Por defecto al ámbito de las `variables` .
desde	cierto	numérico		Valor inicial para el índice.
a	cierto	numérico		Valor final para el índice.
paso	falso	numérico	1	Valor por el cual aumentar o disminuir el índice por iteración.

Bucle de índice básico

El valor final de `x` es 10.

```
<!--- Tags --->
<cfoutput>
    <cfloop index="x" from="1" to="10">
        <li>#x#</li>
    </cfloop>
</cfoutput>
<!--- cfscript --->
<cfscript>
    for (x = 1; x <= 10; x++) {
        writeOutput('<li>' & x & '</li>');
    }
</cfscript>
<!--- HTML Output --->
 - 1
 - 2
 - 3
 - 4
 - 5
 - 6
 - 7
 - 8
 - 9
 - 10
```

Incrementa el paso a 2.

El valor final de `x` es 11.

```
<!--- Tags --->
<cfoutput>
    <cfloop index="x" from="1" to="10" step="2">
        <li>#x#</li>
    </cfloop>
</cfoutput>
<!--- cfscript --->
<cfscript>
    for (x = 1; x <= 10; x += 2) {
```

```
        writeOutput('<li>' & x & '</li>');
    }
</cfscript>
<!--- HTML Output --->
 - 1
 - 3
 - 5
 - 7
 - 9
```

Decremento paso a paso por 1

El valor final de `x` es 0.

```
<!--- Tags --->
<cfoutput>
    <cfloop index="x" from="10" to="1" step="-1">
        <li>#x#</li>
    </cfloop>
</cfoutput>
<!--- cfscript --->
<cfscript>
    for (x = 10; x > 0; x--) {
        writeOutput('<li>' & x & '</li>');
    }
</cfscript>
<!--- HTML Output --->
 - 10
 - 9
 - 8
 - 7
 - 6
 - 5
 - 4
 - 3
 - 2
 - 1
```

CFLoop en una función

Asegúrese de que `var` o `local` alcance el índice dentro de una función. `Foo()` devuelve 11.

```
<!--- var scope --->
<cffunction name="foo" access="public" output="false" returntype="numeric">
    <cfset var x = 0 />
    <cfloop index="x" from="1" to="10" step="1">
        <cfset x++ />
    </cfloop>
    <cfreturn x />
</cffunction>

<!--- Local scope --->
<cffunction name="foo" access="public" output="false" returntype="numeric">
```

```
    <cfloop index="local.x" from="1" to="10" step="1">
        <cfset local.x++ />
    </cfloop>
    <cfreturn local.x />
</cffunction>
```

ColdFusion 11 a través de la corriente

La función `cfloop` no admite el `index` como un mecanismo de contador independiente.

Condición

Sintaxis de etiqueta

Parámetros

Atributo	Necesario	Tipo	Defecto	Descripción
condición	cierto	cuerda		Condición que gestiona el bucle. No puede contener símbolos matemáticos como `<` , `>` o `=` . Debe usar las implementaciones de texto de ColdFusion como `less than` , `lt` , `greater than` , `gt` , `equals` o `eq` .

El valor final de `x` es 5.

```
<cfset x = 0 />
<cfoutput>
    <cfloop condition="x LT 5">
        <cfset x++ />
        <li>#x#</li>
    </cfloop>
</cfoutput>
```

HTML generado

Esto también tendrá un salto de línea entre cada línea de HTML.

```
<li>1</li>
<li>2</li>
<li>3</li>
<li>4</li>
<li>5</li>
```

CFScript

Anterior a ColdFusion 8

```
<cfscript>
x = 0;
while (x LT 5) {
    x = x + 1;
    writeOutput('<li>' & x & '</li>');
}
</cfscript>
```

ColdFusion 8 a través de la corriente

```
<cfscript>
x = 0;
while (x LT 5) {
    x = x++;
    writeOutput('<li>' & x & '</li>');
}
</cfscript>
```

ColdFusion 11 a través de la corriente

> La función `cfloop` no tiene soporte para la `condition` .

HTML generado

Observe que la salida de cfscript está en una sola línea.

```
<li>one</li><li>two</li><li>three</li><li>four</li>
```

Rango de fecha u hora

Ejemplo para el rango de fecha u hora.

Consulta

Considere la mesa `dbo.state_zip` , que contiene las columnas `city` , `statecode` y `zipcode` y cuenta con más de 80.000 registros.

Parámetros

Atributo	Necesario	Tipo	Defecto	Descripción
consulta	cierto	cuerda		El nombre de la variable de un objeto de consulta.

Atributo	Necesario	Tipo	Defecto	Descripción
startrow	falso	numérico		El índice de la fila de inicio del objeto de consulta.
endrow	falso	numérico		El índice de la fila final del objeto de consulta.
grupo	falso	cuerda		El nombre de la columna de consulta en la que se agrupan los registros.

Consulta de ejemplo

```
<cfquery name="geo" datasource="reotrans-dev">
    SELECT city, stateCode, zipCode
    FROM dbo.state_zip
</cfquery>
```

Sintaxis de etiqueta

Usando el objeto de consulta `geo` como fuente para `cfloop` . Dado que la tabla `dbo.state_zip` tiene tantos registros, el HTML generado llevará bastante tiempo. Este ejemplo muestra solo el valor HTML de los primeros 20 registros.

```
<cfoutput>
    <ul>
    <cfloop query="geo">
        <!--- Scope the column names with the query name. --->
        <li>#geo.city# | #geo.stateCode# | #geo.zipCode#</li>
    </cfloop>
    </ul>
</cfoutput>
```

HTML generado

```
<ul>
    <li>100 PALMS | CA | 92274</li>
    <li>1000 PALMS | CA | 92276</li>
    <li>12 MILE | IN | 46988</li>
    <li>1ST NATIONAL BANK OF OMAHA | NE | 68197</li>
    <li>29 PALMS | CA | 92277</li>
    <li>29 PALMS | CA | 92278</li>
    <li>3 STATE FARM PLAZA | IL | 61710</li>
    <li>3 STATE FARM PLAZA | IL | 61791</li>
    <li>30TH STREET | PA | 19104</li>
    <li>3M CORP | MN | 55144</li>
    <li>65TH INFANTRY | PR | 00923</li>
    <li>65TH INFANTRY | PR | 00924</li>
    <li>65TH INFANTRY | PR | 00929</li>
    <li>65TH INFANTRY | PR | 00936</li>
    <li>7 CORNERS | VA | 22044</li>
```

```
    <li>88 | KY | 42130</li>
    <li>9 MILE POINT | LA | 70094</li>
    <li>A A R P INS | PA | 19187</li>
    <li>A A R P PHARMACY | CT | 06167</li>
    <li>A H MCCOY FEDERAL BLDG | MS | 39269</li>
</ul>
```

Limitar la salida a filas específicas

Para limitar el resultado de la consulta a un rango específico de filas, especifique `startrow` y `endrow` .

```
<cfloop query="geo" startrow="100" endrow="150">
    <li>#geo.city# | #geo.stateCode# | #geo.zipCode#</li>
</cfloop>
```

Salida de agrupación

En los datos de ejemplo, el mismo estado aparece varias veces en relación con las múltiples ciudades que están asociadas a cada estado. También puede ver la misma ciudad enumerada varias veces en relación con los múltiples códigos postales asociados a cada ciudad.

Vamos a agrupar la salida por estado primero. Observe la segunda instancia de `cfloop` envuelve alrededor del contenido que se emitirá bajo el contenido agrupado de `stateCode` .

```
<cfoutput>
    <ul>
    <cfloop query="geo" group="stateCode">
        <!--- Scope the column names with the query name. --->
        <li>#geo.stateCode#
            <ul>
                <cfloop>
                    <li>#geo.city# | #geo.zipCode#</li>
                </cfloop>
            </ul>
        </li>
    </cfloop>
    </ul>
</cfoutput>
```

HTML generado (extracto) de una etiqueta `cfloop` agrupada.

```
<ul>
    <li>AK
        <ul>
            <li>KONGIGANAK | 99545</li>
            <li>ADAK | 99546</li>
            <li>ATKA | 99547</li>
            <!-- etc. -->
        </ul>
    </li>
    <li>AL
        <ul>
```

```
            <li>ALEX CITY | 35010</li>
            <li>ALEXANDER CITY | 35010</li>
            <li>ALEX CITY | 35011</li>
            <!-- etc. -->
        </ul>
    </li>
    <!-- etc. -->
</ul>
```

Finalmente, `stateCode` la salida por `stateCode` , luego por `city` para ver todas las entradas `zipCode` por ciudad. Observe que el segundo `cfloop` ahora está agrupado por `city` y existe un tercer `cfloop` para generar los datos del `zipCode` .

```
<cfoutput>
    <ul>
    <cfloop query="geo" group="stateCode">
        <li>#geo.stateCode#
            <ul>
            <cfloop group="city">
                <li>#geo.city#
                    <ul>
                        <cfloop>
                            <li>#geo.zipCode#</li>
                        </cfloop>
                    </ul>
                </li>
            </cfloop>
            </ul>
        </li>
    </cfloop>
    </ul>
</cfoutput>
```

HTML generado (extracto) a partir de dos etiquetas `cfloop` agrupadas.

```
<ul>
    <li>AK
        <ul>
            <li>ADAK
                <ul>
                    <li>99546</li>
                    <li>99571</li>
                </ul>
            </li>
            <li>AKHIOK
                <ul>
                    <li>99615</li>
                </ul>
            </li>
            <!--- etc. --->
            <li>BARROW
                <ul>
                    <li>99723</li>
                    <li>99759</li>
                    <li>99789</li>
                    <li>99791</li>
                </ul>
            </li>
```

```
            <!--- etc. --->
        </ul>
    </li>
    <!--- stateCodes etc. --->
</ul>
```

CFScript

ColdFusion 6 (MX) aunque actual

```
<cfscript>
    for (x = 1; x LTE geo.recordcount; x = x + 1) {
        writeOutput( '<li>' & geo.city[x] & ' | ' &
            geo.stateCode[x] & ' | ' & geo.zipCode[x] & '</li>');
    }
</cfscript>
```

ColdFusion 8 aunque actual

```
<cfscript>
    for (x = 1; x <= geo.recordcount; x++) {
        writeOutput( '<li>' & geo.city[x] & ' | ' &
            geo.stateCode[x] & ' | ' & geo.zipCode[x] & '</li>');
    }
</cfscript>
```

ColdFusion 10 aunque actual

Con la sintaxis de `FOR IN` , `x` es un objeto de fila de consulta, no el índice de fila.

```
<cfscript>
    for (x in geo) {
        writeOutput( '<li>' & x.city & ' | ' &
            x.stateCode & ' | ' & x.zipCode & '</li>');
    }
</cfscript>
```

ColdFusion 11 aunque actual

ColdFusion 11 permite que la mayoría de las etiquetas se escriban como cfscript.

```
<cfscript>
    cfloop(query: geo, startrow: 1, endrow: 2) {
        writeOutput( '<li>' & geo.city & ' | ' &
            geo.stateCode & ' | ' & geo.zipCode & '</li>');
    }
</cfscript>
```

Con `group` .

```
<cfscript>
    cfloop(query: geo, group: 'city') {
        writeOutput( '<li>' & geo.city & '<ul>');
        cfloop() { // no arguments, just as in the tag syntax.
            writeOutput('<li>'  & geo.zipCode & '</li>');
        }
        writeOutput('</ul></li>');
    }
</cfscript>
```

Lista

Considera esta lista:

```
<cfset foo = "one,two,three,four" />
```

Sintaxis de etiqueta

Parámetros

Atributo	Necesario	Defecto	Descripción
lista	cierto		Un objeto de lista. La variable debe ser evaluada (envuelta con ##)
índice	cierto		El elemento actual de la lista.

```
<cfoutput>
    <cfloop list="#foo#" index="x">
        <li>#x#</li>
    </cfloop>
</cfoutput>
```

HTML generado

Esto también tendrá un salto de línea entre cada línea de HTML.

```
<li>one</li>
<li>two</li>
<li>three</li>
<li>four</li>
```

CFScript

Anterior a ColdFusion 8

```
<cfscript>
    for (x = 1; x LTE listLen(foo); x = x + 1) {
        writeOutput("<li>" & listGetAt(foo, x) & "</li>");
    }
</cfscript>
```

ColdFusion 8 a través de la corriente

```
<cfscript>
    for (x = 1; x <= listLen(foo); x++) {
        writeOutput("<li>" & listGetAt(foo, x) & "</li>");
    }
</cfscript>
```

ColdFusion 9 a través de la corriente

```
<cfscript>
    for (x in foo) {
        writeOutput("<li>" & x & "</li>");
    }
</cfscript>
```

ColdFusion 11 a través de la corriente

> La función `cfloop` no tiene soporte para la `list` .

HTML generado

Observe que la salida de cfscript está en una sola línea.

```
<li>one</li><li>two</li><li>three</li><li>four</li>
```

Formación

> La capacidad de usar directamente un objeto de `array` con `cfloop` se agregó en ColdFusion 8.

Considera esta matriz;

```
<cfset aFoo = [
    "one"
    , "two"
    , "three"
    , "four"
] />
```

Sintaxis de etiqueta

ColdFusion 8 a través de la corriente

Usando el `index` atributos por sí mismo.

Parámetros

Atributo	Necesario	Defecto	Descripción
formación	cierto		Un objeto de matriz. La variable debe ser evaluada (envuelta con ##)
índice	cierto		El elemento actual de la matriz.

```
<cfoutput>
    <cfloop array="#aFoo#" index="x">
        <li>#x#</li>
    </cfloop>
</cfoutput>
```

HTML generado

Esto también tendrá un salto de línea entre cada línea de HTML.

```
<li>one</li>
<li>two</li>
<li>three</li>
<li>four</li>
```

ColdFusion 2016 a través de la corriente.

> El `item` atributo cambia el comportamiento de `cfloop` partir de Coldfusion 2016.

Usar el `item` atributo en lugar de o además del `index` .

Parámetros

Atributo	Necesario	Defecto	Descripción
formación	cierto		Un objeto de matriz. La variable debe ser evaluada (envuelta con ##)
ít	cierto		El elemento actual de la matriz.
índice	falso		El índice actual de la matriz.

```
<cfoutput>
    <cfloop array="#aFoo#" item="x" index="y">
        <li>#x# | #y#</li>
    </cfloop>
</cfoutput>
```

HTML generado

Esto también tendrá un salto de línea entre cada línea de HTML.

```
<li>one | 1</li>
<li>two | 2</li>
<li>three | 3</li>
<li>four | 4</li>
```

CFScript

Anterior a ColdFusion 8

```
<cfscript>
for (i = 1; x LTE arrayLen(aFoo); i = i + 1) {
    writeOutput("<li>" & aFoo[i] & "</li>");
}
</cfscript>
```

ColdFusion 8 a través de la corriente

```
<cfscript>
for (i = 1; i <= arrayLen(aFoo); i = i++) {
    writeOutput("<li>" & aFoo[i] & "</li>");
}
</cfscript>
```

ColdFusion 9 a través de la corriente

Con la sintaxis de `FOR IN` , x es el elemento de matriz actual, no el índice de matriz.

```
<cfscript>
for (x in aFoo) {
    writeOutput("<li>" & x & "</li>");
}
</cfscript>
```

ColdFusion 11 a través de la corriente

La función `cfloop` no tiene soporte para la `array` .

HTML generado

Observe que la salida de `cfscript` está en una sola línea.

```
<li>one</li><li>two</li><li>three</li><li>four</li>
```

Expediente

```
<cfloop list="#myFile#" index="FileItem" delimiters="#chr(10)##chr(13)#">
  <cfoutput>
   #FileItem#<br />
 </cfoutput>
</cfloop>
```

Estructura

Considera esta estructura:

```
<cfset stFoo = {
    a = "one"
    , b = "two"
    , c = "three"
    , d = "foue"
} />
```

Sintaxis de etiqueta

Parámetros

Observe el uso del `item` atributo en lugar del `index` .

Atributo	Necesario	Tipo	Defecto	Descripción
colección	cierto	estructura		Un objeto `struct` . La variable debe ser evaluada (envuelta con ##).
ít	cierto	cuerda		La `key` estructura actual,

Usando funciones de estructura

```
<cfoutput>
    <cfloop collection="#stFoo#" item="x">
        <li>#structFind(stFoo, x)#</li>
    </cfloop>
</cfoutput>
```

Sintaxis de estructura implícita

```
<cfoutput>
    <cfloop collection="#stFoo#" item="x">
        <li>#stFoo[x]#</li>
    </cfloop>
</cfoutput>
```

HTML generado

Esto también tendrá un salto de línea entre cada línea de HTML.

```
<li>one</li>
<li>two</li>
<li>three</li>
<li>four</li>
```

CFScript

Con la sintaxis de `FOR IN` , `x` es una `key` del objeto de estructura.

Salida de las claves de la estructura.

```
<cfscript>
    for (x in stFoo) {
        writeOutput("<li>" & x & "</li>");
    }
</cfscript>
```

HTML generado

```
<li>A</li><li>B</li><li>C</li><li>D</li>
```

Salida el valor de las claves de la estructura.

Usando funciones de estructura

```
<cfscript>
    for (x in stFoo) {
        writeOutput("<li>" & structFind(stFoo, x) & "</li>");
    }
</cfscript>
```

Sintaxis de estructura implícita

```
<cfscript>
    for (x in stFoo) {
        writeOutput("<li>" & stFoo[x] & "</li>");
    }
</cfscript>
```

ColdFusion 11 a través de la corriente

La función `cfloop` no tiene soporte para la `collection` .

HTML generado

Observe que la salida de cfscript está en una sola línea.

```
<li>one</li><li>two</li><li>three</li><li>four</li>
```

Index Loop

Use los atributos `from` y `to` para especificar cuántas iteraciones deberían ocurrir. El atributo de `step` (opcional) le permite determinar qué tan grandes serán los incrementos.

```
<cfloop from="1" to="10" index="i" step="2">
    <cfoutput>
        #i#<br />
    </cfoutput>
</cfloop>
```

Bucle condicional

Utiliza el atributo de `condition` para especificar la condición a usar.

```
<cfset myVar=false>
<cfloop condition="myVar eq false">
    <cfoutput>
    myVar = <b>#myVar#</b>  (still in loop)<br />
 </cfoutput>
    <cfif RandRange(1,10) eq 10>
        <cfset myVar="true">
    </cfif>
</cfloop>
<cfoutput>
    myVar = <b>#myVar#</b> (loop has finished)
</cfoutput>
```

Query Loop

Puede recorrer los resultados de una consulta de ColdFusion.

```
<cfquery name="getMovies" datasource="Entertainment">
    select top 4 movieName
```

```
    from Movies
</cfquery>
<cfloop query="getMovies">
    #movieName#
</cfloop>
```

Listar bucle

Puede usar el atributo `delimiters` (opcional) para especificar qué caracteres se usan como separadores en la lista.

```
<cfloop list="ColdFusion,HTML;XML" index="ListItem" delimiters=",;">
    <cfoutput>
        #ListItem#<br />
    </cfoutput>
</cfloop>
```

Archivo de bucle

Puede pasar sobre un archivo.

```
<cfloop file="#myFile#" index="line">
    <cfoutput>
        #line#<br />
    </cfoutput>
</cfloop>
```

Colección COM / Estructura Loops

Puedes pasar sobre una estructura o colección COM.

```
<cfset myBooks = StructNew()>
<cfset myVariable = StructInsert(myBooks,"ColdFusion","ColdFusion MX Bible")>
<cfset myVariable = StructInsert(myBooks,"HTML","HTML Visual QuickStart")>
<cfset myVariable = StructInsert(myBooks,"XML","Inside XML")>
<cfloop collection="#myBooks#" item="subject">
    <cfoutput>
        <b>#subject#:</b> #StructFind(myBooks,subject)#<br />
    </cfoutput>
</cfloop>
```

Lea CFLOOP Cómo hacer en línea: https://riptutorial.com/es/coldfusion/topic/3035/cfloop-como-hacer

Capítulo 5: Cómo invocar dinámicamente un método privado.

Observaciones

El uso de `<cfinvoke>` o `<cfinvoke> invoke()` debería ser más rápido que `evaluate()`

Examples

CFML

```
<cfinvoke method="#somePrivateMethodName#">
  <cfinvokeargument name="argument1" value="one">
</cfinvoke>
```

CFSCRIPT (CF10 +)

```
invoke("", somePrivateMethodName, {argument1='one'});
```

Lea Cómo invocar dinámicamente un método privado. en línea: https://riptutorial.com/es/coldfusion/topic/6110/como-invocar-dinamicamente-un-metodo-privado-

Capítulo 6: consulta

Parámetros

Parámetro	Detalles
nombre	Valor: *cadena* , Valor predeterminado: sí
dbtype	Valor: consulta / hql, Predeterminado: no, Observaciones: cuando se deja en blanco, es una consulta normal
fuente de datos	Predeterminado: no, Observaciones: base de datos
params	Valor: *estructura* , Predeterminado: no, Observaciones: ¡solo la sintaxis de cfscript! En cfml se escriben dentro del estado de SLQ usando `<cfqueryparam />`

Examples

Usando cfquery dentro de una función

```
<cffunction name="getUserById" access="public" returntype="query">
    <cfargument name="userId" type="numeric" required="yes" hint="The ID of the user">

    <cfquery name="local.qryGetUser" datasource="DATABASE_NAME">
         SELECT  id,
                 name
         FROM    user
         WHERE   id = <cfqueryparam value="#arguments.userId#" cfsqltype="cf_sql_integer">
    </cfquery>

    <cfreturn local.qryGetUser>
</cffunction>
```

Consulta de consulta

Llamadas de función

```
<!--- Load the user object based on the component path. --->
<cfset local.user = new com.User() />
<cfset local.allUsers = user.getAllUsers()>
<cfset local.specificUser = user.getUserIdFromQry(qry = local.allUsers, userId = 1)>
```

Usuario.cfc

```
<cfcomponent>
    <cffunction name="getAllUsers" access="public" returntype="query">
        <cfquery name="local.qryGetAllUsers" datasource="DATABASE_NAME">
            SELECT  id,
                    name
            FROM    user
        </cfquery>

        <cfreturn local.qryGetAllUsers>
    </cffunction>

    <cffunction name="getUserIdFromQry" access="public" returntype="query">
        <cfargument name="qry" type="query" required="Yes" hint="Query to fetch from">
        <cfargument name="userId" type="numeric" required="Yes" hint="The ID of the user">

        <cfquery name="local.qryGetUserIdFromQry" dbtype="query">
            SELECT  id,
                    name
            FROM    arguments.qry
            WHERE   id = <cfqueryparam value="#arguments.userId#" cfsqltype="cf_sql_integer">
        </cfquery>

        <cfreturn local.qryGetUserIdFromQry>
    </cffunction>
</component>
```

Lea consulta en línea: https://riptutorial.com/es/coldfusion/topic/6452/consulta

Capítulo 7: Consultas de base de datos

Examples

Trabajando con bases de datos

Una de las fortalezas de ColdFusion es lo fácil que es trabajar con bases de datos. Y, por supuesto, las entradas de consulta pueden y deben ser parametrizadas.

Implementación de etiquetas

```
<cfquery name="myQuery" datasource="myDatasource" result="myResult">
    select firstName, lastName
    from users
    where lastName = <cfqueryparam value="Allaire" cfsqltype="cf_sql_varchar">
</cfquery>
```

Implementación de CFScript

```
// ColdFusion 9+
var queryService = new query(name="myQuery", datasource="myDatasource");
queryService.addParam(name="lName", value="Allaire", cfsqltype="cf_sql_varchar");
var result = queryService.execute(sql="select firstName, lastName from users where lastName =
:lName");
var myQuery = result.getResult();
var myResult = result.getPrefix();

// ColdFusion 11+
var queryParams = {lName = {value="Allaire", cfsqltype="cf_sql_varchar"}};
var queryOptions = {datasource="myDatasource", result="myResult"};
var myQuery = queryExecute("select firstName, lastName from users", queryParams,
queryOptions);
```

Insertar valores es igual de fácil:

```
queryExecute("
    insert into user( firstname, lastname )
    values( :firstname, :lastname )
",{
    firstname: { cfsqltype: "cf_sql_varchar", value: "Dwayne" }
    ,lastname: { cfsqltype: "cf_sql_varchar", value: "Camacho" }
},{
    result: "local.insertResult"
});

return local.insertResult.generated_key;
```

Ejemplo básico

Las conexiones de la base de datos se configuran utilizando la herramienta Administrador de CF. Consulte Conexiones de base de datos para saber cómo conectar una fuente de datos.

Para ejecutar consultas, todo lo que necesita es la etiqueta `<cfquery>` . La etiqueta `<cfquery>` conecta y abre la base de datos, todo lo que necesita hacer es proporcionarle el nombre de la fuente de datos.

```
<cfquery name="Movies" datasource="Entertainment">
    SELECT title
    FROM   Movies
</cfquery>
```

Para mostrar los resultados de la consulta:

```
<cfoutput query="Movies">
    #title#<BR>
</cfoutput>
```

Autenticación

Muchas configuraciones de base de datos requieren autenticación (en forma de nombre de usuario y contraseña) antes de poder consultar la base de datos. Puede proporcionarlos utilizando los atributos de nombre de usuario y contraseña.

Nota: el nombre de usuario y la contraseña también se pueden configurar contra la fuente de datos en el Administrador de ColdFusion. El suministro de estos detalles en su consulta anula el nombre de usuario y la contraseña en el Administrador de ColdFusion.

```
<cfquery datasource="Entertainment" username="webuser" password="letmein">
    select *
    from Movies
</cfquery>
```

Consultas en caché

Una consulta en caché es una consulta que tiene sus resultados almacenados en la memoria del servidor. Los resultados se almacenan cuando la consulta se ejecuta por primera vez. A partir de entonces, cada vez que se solicite esa consulta, ColdFusion recuperará los resultados de la memoria.

Puede almacenar en caché una consulta usando el atributo `cachedAfter` . Si la consulta se ejecutó por última vez después de la fecha proporcionada, se utilizan datos almacenados en caché. De lo contrario, la consulta se vuelve a ejecutar.

```
<cfquery datasource="Entertainment" cachedAfter="July 20, 2016">
    select *
    from Movies
</cfquery>
```

Para que se utilice la memoria caché y se eviten varias llamadas a la base de datos, la consulta actual debe usar la misma declaración SQL, fuente de datos, nombre de la consulta, nombre de usuario y contraseña que la consulta almacenada en caché. Esto incluye espacios en blanco en la

consulta.

Como tal, las siguientes consultas crean diferentes cachés, aunque los caracteres recortados son los mismos y los resultados de la consulta son idénticos:

```
<cfquery datasource="Entertainment" cachedAfter="July 20, 2016">
    select *
    from Movies
    <cfif false>
    where 1 = 1
    </cfif>
    <cfif true>
    where 1 = 1
    </cfif>
</cfquery>

<cfquery datasource="Entertainment" cachedAfter="July 20, 2016">
    select *
    from Movies
    <cfif true>
    where 1 = 1
    </cfif>
    <cfif false>
    where 1 = 1
    </cfif>
</cfquery>
```

Limitar el número de registros devueltos

Puede limitar el número de filas que se devolverán utilizando el atributo `maxrows` .

```
<cfquery datasource="Entertainment" maxrows="50">
    select *
    from Movies
</cfquery>
```

Tiempos de espera

Puede establecer un límite de tiempo de espera utilizando el atributo de `timeout` . Esto puede ser útil para evitar que las solicitudes se ejecuten por mucho más tiempo de lo que deberían e impactar en toda la aplicación como resultado.

El atributo de `timeout` establece el número máximo de segundos que cada acción de una consulta puede ejecutar antes de devolver un error.

```
<cfquery datasource="Entertainment" timeout="30">
    select *
    from Movies
</cfquery>
```

Lea Consultas de base de datos en línea:
https://riptutorial.com/es/coldfusion/topic/4582/consultas-de-base-de-datos

Capítulo 8: Creando APIs REST en coldfusion

Introducción

Las API de REST son interesantes cuando se debe acceder a los datos desde cualquier lugar, incluidos diferentes idiomas (servidor y cliente). Eso requiere separación de datos y procesamiento.

Examples

Creación de backend

```
<cfcomponent displayname="myAPI" output="false">
    <cffunction name="init" access="public" output="no">
        <!--- do some basic stuff --->
        <cfreturn this>
    </cffunction>

    <cffunction name="welcome">
        <cfreturn "Hello World!">
    </cffunction>
</cfcomponent>
```

La interfaz

```
<cfscript>
    api_request = GetHttpRequestData();
    api = createObject("component","myAPI").init();
</cfscript>

<cfif api_request.method is 'GET'>
    <cfoutput>#api.welcome()#</cfoutput>
<cfelseif api_request.method is 'POST'>
    <cfheader statuscode="500" statustext="Internal Server Error" />
</cfif>
```

Lea Creando APIs REST en coldfusion en línea:
https://riptutorial.com/es/coldfusion/topic/10698/creando-apis-rest-en-coldfusion

Capítulo 9: Trabajar con RegExp Reemplazar devoluciones de llamada

Introducción

Si desea algo más que un simple reemplazo de cadena con expresiones regulares comunes, ciertamente se encontrará con problemas y llegará a la pared cuando descubra los límites de las funciones de expresión regular que tiene Coldfusion. No hay una función incorporada como la `preg_replace_callback` de php.

Parámetros

Parámetro	Detalles
`re`	La expresion regular
`str`	La cadena a la que debe aplicarse la expresión regular.
`callback`	La función donde se agrupó el grupo capturado se pasará si se encuentra una coincidencia. Allí se pueden procesar los partidos.

Observaciones

Debido a que Coldfusion en sí no ofrece lo que queremos, recurrimos a la variedad de Java, que es, como todos sabemos, por encima de Coldfusion. Java nos ofrece `java.util.regex.Pattern` .

Así que aquí está lo que realmente hacemos:

1. Invoque el método de `Compile` desde el objeto de Clase de `Pattern` y pasándole el patrón de expresiones regulares (que probablemente deposite el patrón de expresiones regulares para su uso posterior).
2. Invoque el método `Matcher` sobre lo que devolvió el método `Compile` y pasando la cadena donde se debe ejecutar el patrón.
3. Compruebe si la coincidencia fue exitosa invocando el método de `find` sobre lo que devolvió el método de `Matcher` .

Si `matcher.find()` devuelve `true` , podríamos decir "Eso es todo", pero hay una pequeña cosa que debemos tener en cuenta: el objeto Patrón de Java almacena los grupos y nos da acceso a través de otra función, que no siempre es la mejor manera de Procesamiento adicional y no tan consistente con respecto a cómo otros lenguajes de programación manejan este caso. Por lo tanto, hacemos un bucle sobre `matcher.group()` para que podamos pasar una matriz que contiene los grupos capturados a la función de devolución de llamada. Y ahora podemos decir: "¡Eso es!"

Examples

Función de REReplaceCallback definida por el usuario

```
function REReplaceCallback(re,str,callback) {
    /*
        Thanks to Ben Nadel
        "Learning ColdFusion 8: REMatch() For Regular Expression Matching"
        from 2007-06-13
        https://www.bennadel.com/blog/769-learning-coldfusion-8-rematch-for-regular-
expression-matching.htm
    */
    pattern = CreateObject("java","java.util.regex.Pattern").Compile(Arguments.re);
    matcher = pattern.Matcher(Arguments.str);
    if(matcher.find()) {
        groups = [];
        for(var i = 1; i lte matcher.groupCount(); i++) {
            ArrayAppend(groups,matcher.group(Val(i)));
        }
        return Arguments.callback(groups);
    }
    else {
        return Arguments.callback(false);
    }
}
```

Usando la función REReplaceCallback

```
REReplaceCallback('YOUR REGEX GOES HERE','AND YOUR STRING HERE',function(groups) {
    //now you can access the 'groups' array containing all the captured groups
    return result; //return whatever you've processed inside
});
```

Lea Trabajar con RegExp Reemplazar devoluciones de llamada en línea: https://riptutorial.com/es/coldfusion/topic/10655/trabajar-con-regexp-reemplazar-devoluciones-de-llamada

Capítulo 10: Variables

Parámetros

Atributo	Descripción
nombre	(Requerido) Nombre del parámetro / variable.
defecto	(Opcional) Valor para establecer el parámetro si no existe.
max	(Opcional) El valor máximo válido; Usado solo para la validación de rangos.
min	(Opcional) El valor mínimo válido; Usado solo para la validación de rangos.
modelo	(Opcional) Una expresión regular de JavaScript que el parámetro debe coincidir; Se usa solo para expresiones regulares o validación de expresión regular.
tipo	(Opcional) El formato válido para los datos.

Examples

Utilizando cfset

Puede establecer una variable de ColdFusion usando la etiqueta `<cfset>` . Para generar la variable, debe rodear el nombre de la variable con símbolos `#` hash y encerrarlo dentro de las etiquetas `<cfoutput>` .

```
<cfset variablename="World!">
<cfoutput>
    Hello #variablename#
</cfoutput>
```

Utilizando cfparam

La etiqueta `<cfparam>` crea una variable si aún no existe. Puede asignar un valor predeterminado utilizando el atributo `default` . Se puede usar si desea crear una variable, pero no desea sobrescribirla si se ha creado anteriormente en otro lugar.

Aquí la variable no se ha establecido previamente, por lo que se asignará con la etiqueta `<cfparam>` .

```
<cfparam name="firstName" default="Justin">
<cfoutput>
    Hello #firstName#
</cfoutput>
```

Aquí la variable ya se ha asignado utilizando la etiqueta `<cfset>` , por lo que este valor reemplazará el valor predeterminado en la etiqueta `<cfparam>` .

```
<cfset firstname="Justin">

<cfparam name="firstName" default="Barney">
<cfoutput>
    Hello #firstName#
</cfoutput>
```

Comprobando si existe una variable

Puede verificar si una variable se ha definido en un ámbito utilizando la función `StructKeyExists()` incorporada de ColdFusion. Esto se puede usar dentro de una etiqueta `<cfif>` para evitar mensajes de error en el caso de que intentes referirte a una variable que no existe. También puede usar esta función para determinar si un usuario ha realizado una determinada acción o no. La sintaxis de la función es

```
StructKeyExists(structure, "key")
```

El siguiente ejemplo comprueba si la variable `firstName` existe en el ámbito de las `variables` .

```
<cfif StructKeyExists(variables, "firstName")>
    Hello #variables.firstname#!
<cfelse>
    Hello stranger!
</cfif>
```

Alternativamente, puede utilizar la función:

```
isDefined("scopeName.varName")
```

Para evitar ambigüedades, se recomienda declarar el alcance. Por ejemplo, si tienes una variable en la `test` alcance

```
<cfset test.name = "Tracy" />
```

y prueba su `name` en el ámbito global, obtendrá un resultado de `true` .

```
isDefined("name") <!--- true --->
isDefined("x.name") <!--- false--->
isDefined("test.name") <!--- true --->
```

Estableciendo un alcance variable

Es una práctica común establecer variables de aplicación en un ámbito de objeto. Esto hace que sean fáciles de identificar y distinguir de las variables en otros ámbitos.

El alcance de las Variables en un CFC es privado al CFC. Cuando configura variables en este

ámbito, las páginas que invocan el CFC no las pueden ver.

```
<cfparam name="variables.firstName" default="Timmy">
<cfset variables.firstName="Justin">
```

Los ámbitos compartidos con la página de llamada incluyen: Formulario, URL, Solicitud, CGI, Cookie, Cliente, Sesión, Aplicación, Servidor y Flash. Las variables en estos ámbitos también están disponibles para todas las páginas incluidas por un CFC.

CFC:

```
<cfset url.sessionId="23b5ly17">

<cfinclude template="check_session.cfm">
```

check_session.cfm

```
<cfif url.sessionId eq "23b5ly17">
    <p>Welcome back!</p>
</cfif>
```

Lea Variables en línea: https://riptutorial.com/es/coldfusion/topic/4904/variables

Creditos

S. No	Capítulos	Contributors
1	Empezando con la fusión en frío	Adam Tuttle, Adrian J. Moreno, Community, Justin Duhaime, mhatch, RamenChef, shaedrich, Steven Benjamin, user3071284, William Giles
2	Alcances en Coldfusion	James A Mohler, shaedrich
3	Arreglos ColdFusion	4444, Justin Duhaime, mhatch, Mishra Shreyanshu, shaedrich
4	CFLOOP Cómo hacer	Adrian J. Moreno, Bonanza, mhatch, RRK
5	Cómo invocar dinámicamente un método privado.	Henry
6	consulta	Bubblesphere, shaedrich
7	Consultas de base de datos	Adam Tuttle, Leigh, mhatch, nosilleg, shaedrich, user3071284
8	Creando APIs REST en coldfusion	shaedrich
9	Trabajar con RegExp Reemplazar devoluciones de llamada	shaedrich
10	Variables	mhatch