

# ColdFusion 8 : CFML Enhancements

Rupesh Kumar  
Computer Scientist  
Adobe

June 27, 2007

<http://coldfused.blogspot.com>

# Agenda

- Syntax Enhancements
- CFC Enhancements
- Multi-Threading
- New File I/O
- Misc
- Q&A

# Syntax Enhancements

# New Operators

- Prefix and Postfix operators
  - ++, --
- Modulus
  - %
- Assignment operators
  - +=, -=, \*=, /=, %=, &=
- Logical operators
  - &&, ||, !
- Comparison Operators (Only in cfscrip) t
  - ==, !=, <, <=, >, >=

# Implicit Array

Before :

```
<cfset myarray = ArrayNew(1)>  
<cfset ArrayAppend(myArray, "foo")>  
<cfset ArrayAppend(myArray, 2*3)>  
<cfset ArrayAppend(myArray, someVar)>
```

Now :

```
<cfset myArray = ["foo", 2*3, someVar]>
```

# Implicit Struct

Before :

```
<cfset Struct mystruct = StructNew()>  
<cfset mystruct.key1 = "Value1">  
<cfset mystruct.key2 = 2*3>  
<cfset mystruct.key3 = someVar>
```

Now :

```
<cfset mystruct = {key1 = "Value1",  
                  key2 = 2*3,  
                  key3 = someVar}>
```

# Implicit Array and Struct Creation

- Elements in array and Struct can be any valid CFML expression
- Nested syntax like below not allowed

```
<cfset arrayOfStructs =  
    [{key= "value"}, {s2key= "value"}]
```

- Use Instead

```
<cfset struct1={key= "value"}>  
<cfset struct2={s2key= "value"}>  
<cfset arrayOfStructs =[struct1, struct2]>
```

# AttributeCollection

- Instead of passing tag attributes, pass a struct to tags.
- All the attributes of any tag can be passed in a single Struct
- Much cleaner code for conditional tag attributes.

```
<cfset application.baseattrs = {Server="xxxx",  
Username="yyyy", password="zzzz", From="rupeshk@xx.com"}>
```

```
<cfset attrs = {To="xyz@xxxx.com", Subject="blah blah",  
Type="HTML"}>
```

```
<cfset StructAppend(attrs, application.baseattrs)>
```

```
<cfmail attributeCollection = "#attrs#">
```

- PrecisionEvaluate(expression)

- Needed when precision is crucial

```
<cfoutput>#(1001.18 + 30 - 1031.18)#</cfoutput> // Non-zero  
<cfoutput>#precisionEvaluate(1001.18 + 30- 1031.18)#</cfoutput>
```

- CFLoop Enhancements

- Loop over an array

```
<cfloop array="myarr" index="elmt">  
    <cfoutput>#elmt#</cfoutput>  
</cfloop>
```

- Loop over a text file

```
<cfloop file = "a.txt" index = "curLine">  
    <cfoutput>#curLine#</cfoutput>  
</cfloop>
```

- ListToArray(list, delimiter, includeEmptyElements)

# CFC Enhancements

# Interface

- Interface provides a contract between two entities
- Helps in better design of application specially frameworks
- **<CFINTERFACE>** used in a CFC which can only contain **<CFFUNCTION>** and **<CFARGUMENT>** tags
- The function does not contain any body
- Interface implemented by component
  - All methods must be implemented
  - Method signature must match

# Interface Example

```
<!--- Worker interface --->
<cfinterface>
  <cffunction name="execute">
    <cfargument name="task" type="struct"
      required="true">
    </cffunction>

</cfinterface>

<!--- SmartWorker component --->
<cfcomponent implements="Worker">
  <cffunction name="execute">
    <cfargument name="Task" type="struct"
      required="true">
    <!--- Pretend to work --->
    <cfset sleep(5000)>
    </cffunction>

</cfcomponent>
```

# Interface ...

- Performance cost is negligible
  - Validation check only on the first component creation
- `IsInstanceOf(comp, type)`
- `GetComponentMetadata`
  - Returns the meta data of component/interface without instantiation

# CFC – Others

- Huge performance improvement.
  - Upto 20x improvement
- Serializable
  - CFC stored in the session can be retrieved back on failover in cluster
  - CFC referencing a CFC also supported
- Can be duplicated now.
  - Creates a deep copy
  - Should be careful so as not to duplicate singleton cfc
- OnMissingMethod callback to component
  - Missing method signature available in the callback
  - handle errors
  - Handle all getXXX and setXXX calls without defining those methods



# Application.cfc

- OnMissingTemplate callback to application
  - Function invoked on file not found condition
  - OnMissingTemplate (URI)
  - Handle Errors
  - Build controller like logic to handle a particular type of URL.
- Per Application Settings
  - Mappings
  - Custom tag paths
  - Via THIS scope in Application.cfc
  - Overrides server wide settings in Administrator

```
<cfset THIS.mappings["MyMap"]="c:\inetpub\myStuff">  
<cfset THIS.customtagpaths = "c:\mapped1,c:\mapped2">
```

# Multi-Threading

# Multi-Threading in CFML – CFThread

- Allows cf page to launch multiple tasks to run in parallel.
- New threads run asynchronous to the page
- Can be used when
  - Lot of independent tasks that need not be synchronous
  - Lot of tasks for which end user need not be made to wait

```
<cfthread name="mythread">  
  <!--- any cfml code --->  
  <cfset avail = getSeatAvail(airline)>  
  <cfset price = getPrice(airline)>  
</cfthread>
```


# CFThread

 `<cfset = 10>`

`<cfset threadname = "T1">`

`<cfthread name="#threadname#" myurl="http://cfunited.com/">`

```
<cfhttp url="#myurl#"
        method="GET" result=#cfunited">
  <cfmail attributeCollection=attrs>
    #cfunited.FileContent#
  </cfmail>
<cfset thread.result = #cfunited#>
```



`</cfthread>`

`<cfset foo()>`

...  
...

`<cfthread action="join" />`

`<cfdump var="#T1.result#">`

 Request Thread

 CFThread

# Passing Data to thread

Thread unsafe

```
<cfloop query= "files">
  <cfthread name="thread#i++#">
    <cffile action="COPY" source="#src#\#files.name#"
            destination="#dest#">
  </cfthread>
</cfloop>
```

- Pass data in thread-safe way as attribute (attribute scope)
- Will be accessed using attributes.xxx
- Data passed is duplicated for thread safety.

```
<cfthread name="mythread" filename="#files.name#">
  <cffile action="COPY"
          source="#src#\#attributes.filename#"
          destination="#dest#">
</cfthread>
```

# Inter-Thread Communication

- Join

- Current thread waits for another thread to finish.
- The waiting thread can be a cfthread or request thread
- can specify wait timeout.

```
<!--- wait for th1 and th2 to finish.  
      If they don't finish in 5 seconds, resume --->  
<cfthread action="join" name="th1, th2" timeout="5000"/>
```

- Terminate

- A cfthread or request thread can terminate another cfthread.

```
<cfthread action="terminate" name="t1">
```

- Data exchange using “Thread” scope.

# Thread Scope

- Shared scope between threads.
- Used to make thread specific data available to other threads.
- Only owner thread can write but anyone can read.
- Accessed using thread name.
- Can also be accessed using "Thread" by owner thread.
- Also contains thread metadata
  - Name
  - Priority
  - StartTime
  - ElapsedTime
  - Output
  - Error
  - Status : NOT\_STARTED | RUNNING | WAITING | COMPLETED | TERMINATED



# CFThread – What else you need to know

- Can live much beyond the request thread
- Thread name must be unique in the request
- Like a function, it has its own local scope
- Can access all the scopes and its variables
- cfoutput inside thread body will not write to the page response.
- Data is written to a special buffer named "Output" in thread scope.
- Any error in thread does not affect the request.
- Error available to other threads using "Error" in thread scope

# Example

```
<cfloop query="dir">
  <cfset threadname = "thread_#i++#">
  <cfthread name="#threadname#" filename="#dir.name#">
    <cfset source = "#src#\#filename#">
    <cffile action="COPY" source="#source#"
           destination="#dest#">
      <!--- Set variable in THREAD scope --->
      <cfset thread.newpath =
        "#dest#\#attributes.filename#">
      Copied Successfully
    </cfthread>
  </cfloop>
  <!--- Access THREAD scope outside --->
  <cfset reading=thread_1.newpath>
  <cfoutput>#thread_1.output#</cfoutput>
```

Pass as  
attribute to  
thread

Local to the  
thread

Set the result  
in Thread  
scope

Access using  
Attributes  
scope

Access data  
from Thread  
scope

Access  
Output from  
thread

## CFThread Demo

# New File I/O

# New File I/O

- Much better performance in reading/writing of file than cfile
- Read/Write is done in chunk and not on the entire file
- Very useful for reading/writing large files
- Works on stream
- Open a File → Do Read/Write → Close the file.
- FileOpen (filepath, [mode], [charset]) → **fileobj**
  - Mode - read (default) | readBinary | write | append
- FileObject
  - handle to the stream
  - must be closed after you are done with it.

# Working with File Object

- Read/Write from/to fileobject
  - FileRead(fileobj, no of chars/bytes to read)
  - FileReadLine(fileobj)
  - FileWrite(fileobj, text or binary data)
- FileIsEOF(fileobj) - Checks whether end of file reached while reading.
- FileClose(fileobj)

```
myfile = FileOpen("c:\temp\myfile.txt", "read");
while (! FileIsEOF(myfile)) {
    x = FileReadLine(myfile); // read line
    writeoutput(x);
}
FileClose(myfile);
```

# File I/O using cfloop

- Read lines from the text file

```
<cfloop file="#filepath#" index="myline">  
  <cfoutput>#myline#</cfoutput>  
</cfloop>
```

```
<cfloop file="#filepath#" index="myline" from="5" to="10">  
  <cfoutput>#myline#</cfoutput>  
</cfloop>
```

- Read characters from the file

```
<!--- read 1000 characters at a time --->  
<cfloop file="#filepath#" index="chars" characters="1000">  
  <cfoutput>#chars#</cfoutput>  
</cfloop>
```

# File Functions

- FileRead ( filepath, FileReadBinary(filepath) )
- FileWrite( filepath, text/binary data)
- FileCopy (source, destination)
- FileMove (source, destination)
- FileDelete (filepath)
- FileSetAttribute (filepath, attributes)
- FileSetAccessMode (filepath, mode)
- GetFileInfo (filepath)
  - Returns the meta-data for file or directory.
    - Name, path, parent, type, size, lastModified etc

# Misc Enhancements



# DB related Enhancements

- Auto-generated key retrieval
- Transaction Savepoints

```
<cftransaction>
...
<cftransaction action="setsavepoint" savepoint="pointA" />
...
<cftransaction action="rollback" savepoint="pointA" />
...
</cftransaction>
```

- New **<CFDBINFO>** tag, used to obtain:
  - Tables in a data source
  - Table columns
  - Version information
  - Index, PK, and FK details
  - Stored procedure details

# CFDUMP

- Show the first 'n' keys of a structure
- Show or hide query columns or structure keys
- Include SQL and other metadata when dumping query objects

```
<cfdump
  var = "#variable#"
  expand = "yes" or "no"
  label = "text"
  top = "number of rows or levels"
  show = "columns or keys"
  hide = "columns or keys"
  keys = "number of keys to display for structures"
  output = "browser or console or file"
  showUDFs = "yes or no"
  format= "text or html"
>
```

# Others

- CFObject/CreateObject also supports .NET classes
- Javacast enhanced to include
  - Array types e.g; “int[]”
  - byte, short, char, bigdecimal.
- CFZip tag to manipulate zip/jar files
- Secure FTP support
- File Upload
  - A file of any size can be uploaded to the server now.
  - Struct returned by GetHTTPRequestData() will not have any content if called in request in which a file is uploaded.

# Performance Enhancements

- CFC Creation – 20x
- Cfparam – 30x
- List operations – 3x
- Struct operations – 2x
- cfswitch-cfcase – 2.8x
- Cffloop – 2.2x
- Date functions – 6x
- IsDefined – 2x
- Evaluate – 2.75x
- ...

**Questions ?**

**Better by Adobe.™**