Article
Gevorg Arutiunian · Nov 1, 2018 9m read

Utility to compare class and include file definitions between builds

The following code provides information about software builds. Read the original description below for information about the different methods:

```
///Description
///This class enables developers to compare class and INCLUDE routine definitions bet
ween software builds.
///It navigates through all aspects of a class definition and INCLUDE file code and u
ses a 32 bit crc on
///each element to produce a checksum for comparison purposes.
///The utility can simply return a checksum value, provides details at certain levels
 and output the
///results to a file for comparison if required.
///
///The class has three primary methods
///
///Class (class, details, filename)
///This method will provide a checksum for 1 class
///
///Package (package, details, filename)
///This method will checksum information for all classes that are members of a given
package
///
///Namespace (details , filename)
///This method will provide a checksum for all non-
system classes in a namespace and a checksum for all
///INCLUDE files in that namespace.
///This method is recommended as the preferred mechanism for comparing two software d
efinitions in two
///different namespaces.
///
///The details flag operates with the following values
///0 - No details written, total checksum returned
///1 - Class total written, total checksum returned
///2 - Class total and element total written , total checksum returned
///3 - Class total,element total and named element total written , total checksum ret
urned

Class objectscript.checkBuild extends (%RegisteredObject, %XML.Adaptor) [ClassType =
"", Inheritance = right, ProcedureBlock]
{
    /// Define the crcmode = 7 "A correct 32-bit CRC"
    Parameter CRCMODE = 7;

    /// Provides a checksum for 1 class based on summation of 32 bit CRC checking
    ///
Details
    ///
0 - No details written, total checksum returned
```

```
    ///
1 - Class total written, total checksum returned
    ///
2 - Class total and element total written , total checksum returned
    ///
3 - Class total,element total and named element total written , total checksum return
ed
    ///
    ClassMethod Class(class As %String, details As %Integer = 0, filename As %String
= "") As %Integer
    {
        //Open the file if requested
        if filename'="" {
            set file=..FileOpen(filename)
            if file="Error" {
                write "Unable to open file : ",filename
                quit 0
            }
        }
        else {
            set file=""
        }

        //Checksum 1 class
        set ccs=..CheckClass(class,details,file)

        if file {
            do ..FileClose(file)
        }
        quit ccs
    }

    ClassMethod CheckClass(class As %String, details As %Integer = 0, file = "") As %
Integer
    {
        //Write out a blank line as a seperator followed by the classname
        if details {
            do ..Write("Class "_class,file)
        }

        //Initialize some iteration variables
        set (element,name,node,sub,snode)=""
        set selectivity=""

        //Initialize checksum totals
        //class|element|name
        set cst="0|0|0"

        //Process Header element checksum
        for  {
            set element=$order(^oddDEF(class,element))

            //Completed header information
            if (element="")||(element'?.n) {
                quit
            }

            //Eliminate date/timestamp from header - may vary
            if (element'=63)&&(element'=64)&&(element'=69) {
```

```
                set cst=..Add(cst,$zcrc(^(element),..#CRCMODE))
            }
        }

        //Write out Header checksum details if wanted
        if details>1 {
                do ..Write($char(9)_"Header: "_$piece(cst,"|",2),file)
        }

        //Class elements
        set element("a")="Attributes"
        set element("f")="Foreign Keys"
        set element("i")="Indexes"
        set element("m")="Methods"
        set element("p")="Parameters"
        set element("q")="Queries"
        set element("s")="Storage"
        set element("t")="Triggers"

        //Process all other element checksums
        set element=""

        //Iterate though the Elements
        for  {
            //Get next element
            set element=$order(element(element)) quit:element=""

            //Reset Element checksum
            set $piece(cst,"|",2)=0

            //Iterate through Named Elements
            for  {

                //Iterate through named elements
                set name=$order(^oddDEF(class,element,name)) quit:name=""

                //Reset Named element Checksum
                set $piece(cst,"|",3)=0

                //Iterate through nodes of Named Elements
                for  {
                    set node=$order(^oddDEF(class,element,name,node)) quit:node=""

                    //Add to checksums if data at this level
                    if ($data(^oddDEF(class,element,name,node))'=10)&&(node'=11) {
                        set cst=..Add(cst,$zcrc(^(node),..#CRCMODE))
                    }

                    //Iterate through sub-nodes of nodes of Named Elememts
                    for  {
                        set sub=$order(^oddDEF(class,element,name,node,sub)) quit:sub
=""

                        //Add to checksums
                        if $data(^oddDEF(class,element,name,node,sub))'=10 {
                            set cst=..Add(cst,$zcrc(^(sub),..#CRCMODE))
                        }

                        //Iterate through storage nodes
```

```
                        for  {
                            set snode=$order(^oddDEF(class,element,name,node,sub,"V",
snode)) quit:snode=""

                            //Add to checksums
                            set cst=..Add(cst,$zcrc(^(snode,21),..#CRCMODE))

                            //Update selectivity selectivity if selectivity exists in
 storage definition
                            if $data(^oddDEF(class,"s",name,"M")) {
                                set selectivity="*"
                            }
                        }
                    }
                }

                //Write out Named Element checksum details if requested
                if details>2 {
                    do ..Write($char(9)_$char(9)_name_": "_$piece(cst,"|",3),file)
                }
            }

            //Write out Element checksum details if requested
            if details>1 {
                do ..Write($char(9)_element(element)_": "_$piece(cst,"|",2)_selectivi
ty,file)

                //Reset selectivity indicator to ""
                set selectivity=""
            }
        }

        //Write out Class checksum details if requested
        if details {
                do ..Write($char(9)_"Checksum: "_$piece(cst,"|",1),file)
        }

        //Return Class checksum
        quit $piece(cst,"|",1)
    }

    /// Provides a checksum for a package(s) based on summation of 32 bit CRC checkin
g
    ///
"PackageName" - 1 package
    ///
"" - All packages in a namespace (excludes % - Sydtem classes)
    ///
Details
    ///
0 - No details written, total checksum returned
    ///
1 - Class total written, total checksum returned
    ///
2 - Class total and element total written , total checksum returned
    ///
3 - Class total,element total and named element total written , total checksum return
ed
    ///
```

```
    ClassMethod Package(package As %String = "", details As %Integer = 0, filename As
 %String = "") As %Integer
    {
        //Open the file if requested
        if filename'="" {
            set file=..FileOpen(filename)
            if file="Error" {
                write "Unable to open file : ",filename
                quit 0
            }
        }
        else {
            set file=""
        }

        //Checksum Package(s)
        set pcs=..CheckPackage(package,details,file)

        //Close the file
        if file {
            do ..FileClose(file)
        }

        //Return package checksum
        quit pcs
    }

    ClassMethod CheckPackage(package As %String = "", details As %Integer = 0, file A
s %File = "") As %Integer
    {

        //Initiate package anc total checksum
        set (tcs,pcs)=0

        //Eliminate "%" system classes and checksum all packages if package=""
        if package="" {
            set package="@"
            set cpackage=$piece($order(^oddDEF(package)),".",1)
        }

        //Iterate through a package/packages(s) sending classes off to CheckClass
        set class=package
        for  {
            set class=$order(^oddDEF(class))
            if (class="")&&(package="@") {
                if details {
                    do ..Write(cpackage_": "_pcs,file)
                    do ..Write("",file)
                }
                quit
            }
            elseif (package'="@")&&($piece(class,".",1)'=package) {
                if details {
                    do ..Write(package_": "_pcs,file)
                    do ..Write("",file)
                }
                quit
            }
            elseif (package="@")&&($piece(class,".",1)'=cpackage) {
```

```
                if details {
                    do ..Write(cpackage_": "_pcs,file)
                    do ..Write("",file)
                }
                set pcs=0
                set cpackage=$piece(class,".",1)
            }
            else {
                set ccs=..CheckClass(class,details,file)
                set tcs=tcs+ccs
                set pcs=pcs+ccs
            }
        }

        //Write the package total checksum
        if details {
            do ..Write("Checksum: "_tcs,file)
        }

        quit tcs
    }

    /// Provides a checksum for a Namespace based on summation of 32 bit CRC checking
    ///
 This includes all INCLUDE files for code generation
    ///
"" - All packages in a namespace (excludes % - Sydtem classes)
    ///
Details
    ///
0 - No details written, total checksum returned
    ///
1 - Class total written, total checksum returned
    ///
2 - Class total and element total written , total checksum returned
    ///
3 - Class total,element total and named element total written , total checksum return
ed
    ///
    ClassMethod Namespace(details As %Integer = 0, filename As %String = "") As %Inte
ger
    {

        //Open the file if requested
        if filename'="" {
            set file=..FileOpen(filename)
            if file="Error" {
                write "Unable to open file : ",filename
                quit 0
            }
        }
        else {
            set file=""
        }

        //Go through the class packages first
        set ncs=..CheckPackage("",details,file)

        //Calculate the INCLUDE files
```

```
        if details {
            do ..Write("",file)
            do ..Write("Include Files",file)
        }

        //Initialize INCLUDE files checksum
        set ics=0

        set routine="@",line=""

        for  {
            set routine=$order(^rINC(routine)) quit:routine=""

            //Initialize INCLUDE ROUTINE checksum
            set rcs=0

            //Iterate through the include file routines
            for  {
                set line=$order(^rINC(routine,0,line)) quit:line=""
                set rcs=rcs+$zcrc(^(line),..#CRCMODE)
                set ics=ics+$zcrc(^(line),..#CRCMODE)
                set ncs=ncs+$zcrc(^(line),..#CRCMODE)
            }

            if details>1 {
                do ..Write($char(9)_routine_": "_rcs,file)
            }
        }

        //Write out the INCLUDE files checksum
        if details {
                do ..Write("Checksum: "_ics,file)
        }

        //Write out the Namespace checksum
        if details {
                do ..Write("",file)
                do ..Write("Namespace: "_ncs,file)
        }

        //Close the file
        if file {
            do ..FileClose(file)
        }

        //Return the namespace checksum
        quit ncs
    }

    ClassMethod FileOpen(filename As %String) As %File
    {
        set file=##class(%File).%New(filename)
        set ok=file.Open("WNS")
        if 'ok {
            do $system.OBJ.DisplayError(ok)
            quit "Error"
        }
        else {
            quit file
```

```
        }
    }

    ClassMethod FileClose(file As %File)
    {
        do file.Close()
        quit
    }

    ClassMethod Write(string As %String, file As %File)
    {
        if file {
            do file.WriteLine(string)
        }
        write !,string
        quit
    }

    ClassMethod Add(cst As %String, crc As %Integer) As %String
    {
        set $piece(cst,"|",1)=$piece(cst,"|",1)+crc
        set $piece(cst,"|",2)=$piece(cst,"|",2)+crc
        set $piece(cst,"|",3)=$piece(cst,"|",3)+crc
        quit cst
    }
}
```

Here's a link to the code on GitHub: https://github.com/intersystems-community/code-snippets/blob/master/src/...

#Caché #Code Snippet #InterSystems IRIS #ObjectScript