#this class will be generating the manager pages

class DocGen camel case here but rest are lower camel case, I assume Ruby has some sort of standard here

# the constructor

def initialize(aConn)

#the connection

@conn=aConn

end

# the sql retrive code

def generatedDoc()

asciidocCode = self.generateColumnDef

#asciidocCode = asciidocCode + self.generateStoredProcDef

return asciidocCode

end

//this part here is commented out and probably should be removed

#this functin will generate the class def

# def genrateTableDef()spelling

# res=@conn.exec("select tablename from pg\_tables where schemaname='public'")

# retType = ""

# for i in res variables could be named more functionally

# retType = retType +"=="+ i[0].to\_s + "\n"

# end

# puts retType

# return retType

# end

#get the Comments of the table for the tables used

def genrateTabledesc(tabName)<-spelling mistake -generate\*

res=@conn.exec("select obj\_description(oid) from pg\_class where relname='#{tabName}'")

retType=""

for i in res

retType = retType + "#{i[0].to\_s}"

end

return retType

end

//spelling mistake

#get the defination of the columns for the tables used

def generateColumnDef()

res=@conn.exec("select tablename from pg\_tables where schemaname='public'")

classObj = ""

for i in res

classObj = classObj + "\n\n\n== "+ i[0].to\_s + "\n"

classObj = classObj + genrateTabledesc(i[0].to\_s)

classObj = classObj + "\n"

classObj = classObj + ".Properties\n"

classObj = classObj + "[frame=\"none\",options=\"header\"]\n"

classObj = classObj + "|======================================================================================\n"

classObj = classObj + "| | Data Type | Description \n"

classObj = classObj + self.singleTableColDef(i.to\_s)

classObj = classObj + "|======================================================================================\n"

classObj = classObj + "=== Links\n"

classObj = classObj + self.generateForeignKeyDef(i.to\_s)

classObj = classObj + "=== Methods\n"

classObj = classObj + self.generateStoredProcDef(i.to\_s)

end

puts classObj

return classObj

end

#required with generatecolumndef to get the defination <spelling of columns of a single table

def singleTableColDef(tabName)

res=@conn.exec("SELECT column\_name,data\_type,column\_default,is\_nullable FROM information\_schema.columns WHERE table\_name ='#{tabName}'")

retType=""

for i in res

colVal = "+"+i[0].to\_s+"+"

dataType = "+"+i[1].to\_s+"+"

retType = retType + "|"+colVal.center(25)+"|"+dataType.center(24)+"|"+self.singleTableColDesc(tabName)

retType = retType + "\n"

end

return retType

end

def singleTableColDesc(tabName)

res=@conn.exec("SELECT pg\_class.relname as classname, attname, description FROM pg\_class, pg\_attribute, pg\_description WHERE pg\_class.relname = '#{tabName}' AND pg\_class.oid = pg\_attribute.attrelid AND pg\_attribute.attnum > 0 AND (objoid = pg\_class.oid) AND (attnum = objsubid) ORDER BY attname;")

retType=""

for i in res

desc="+"+i[2].to\_s.strip+"+"

retType = retType + desc.center(35)

end

return retType.to\_s

end

very compressed code, hard to read

def generateForeignKeyDef(tabName)

res=@conn.exec("SELECT conrelid::regclass AS relname, confrelid::regclass AS mainTable, pg\_get\_constraintdef(oid) AS condef FROM pg\_constraint WHERE contype = 'f'

")

retType=""

for i in res

relName = i[0].to\_s

mainTable = i[1].to\_s

if relName == "#{tabName}" or mainTable== "#{tabName}" then

#.+films+ (++actors\_\_films++)

retType = retType + ".+#{i[1].to\_s}+(++#{i[0].to\_s}\_\_#{i[1].to\_s}++)\n"

retType = retType + "#{i[2].to\_s}\n"

end

end

#puts retype should not have puts here as a comment

return retType

end

def generateStoredProcDef(tabName)

res=@conn.exec("select proname, proargnames, type\_udt\_name from pg\_proc c inner join information\_schema.routines on routine\_name = proname where proname like 'lp\_%'")

retType=""

for i in res

funcName = i[0].to\_s

funcName = funcName[3,funcName.length-1].capitalize

paramVal = i[1].to\_s

paramVal = paramVal[1,paramVal.length-2]

@returnType = i[2].to\_s

if @returnType == "#{tabName}" then

retType = retType +".++"+funcName+"("+paramVal+")"

retType = retType +" RETURNS SETOF "+@returnType+"++\n"

retType = retType + self.generateStoredProcDesc

end

end

#puts retType

return retType

end

def generateStoredProcDesc

res=@conn.exec("select obj\_description(oid, 'pg\_proc') from pg\_proc where proname like 'lp\_%'")

retType=""

for i in res

funcDesc = i[0].to\_s

retType = retType + funcDesc +"\n"

end

#puts retype should not have puts here as a comment

return retType

end

end