文章
Nicky Zhu · 一月 11, 2021 阅读大约需 9 分钟

跟踪数据更改 - 审计日志 - 上篇

简介

许多应用程序都需要记录数据库中的数据变化，包括：哪些数据被更改、更改人和更改时间（审计日志记录）(维基百科audit logging)。 关于这个问题已经有了很多文章，而关于如何在Caché中实现也有很多不同的方法。

本文将介绍一个机制，帮助您实现用一个框架来跟踪和记录数据更改。一旦您的持久类继承自“ 审计抽象类 ”（Sample.AuditBase），此机制将通过“ objectgenarator ”方法创建一个触发器。由于这个持久类继承了Sample.AuditBase，所以当您编译持久类时，将自动生成用于审计更改的触发器。

Audit Class

这是将记录更改的类。

```
Class Sample.Audit Extends %Persistent
{
  Property Date As %Date;

  Property UserName As %String(MAXLEN = "");

  Property ClassName As %String(MAXLEN = "");

  Property Id As %Integer;

  Property Field As %String(MAXLEN = "");

  Property OldValue As %String(MAXLEN = "");

  Property NewValue As %String(MAXLEN = "");
}
```

Audit Abstract Class

这是您的持久类将继承的抽象类。这个类包含触发器方法（objectgenerator），除了在审核表（Sample.Audit）中写入更改之外，该触发器方法还知道如何识别哪些字段被更改、更改人、新旧值等。

```
Class Sample.AuditBase [ Abstract ]
{

Trigger SaveAuditAfter [ CodeMode = objectgenerator, Event = INSERT/UPDATE, Foreach =
row/object, Order = 99999, Time = AFTER ]
{
  #dim %compiledclass As %Dictionary.CompiledClass
  #dim tProperty As %Dictionary.CompiledProperty
  #dim tAudit As Sample.Audit

  Do %code.WriteLine($Char(9)_"; get username and ip adress")
  Do %code.WriteLine($Char(9)_"Set tSC = $$$OK")
  Do %code.WriteLine($Char(9)_"Set tUsername = $USERNAME")

  Set tKey = ""
```

```
  Set tProperty = %compiledclass.Properties.GetNext(.tKey)
  Set tClassName = %compiledclass.Name

  Do %code.WriteLine($Char(9)_"Try {")
  Do %code.WriteLine($Char(9,9)_"; Check if the operation is an update - %oper = UPDATE")
  Do %code.WriteLine($Char(9,9)_"if %oper = ""UPDATE"" { ")

  While tKey '= "" {
    set tColumnNbr = $Get($$$EXTPROPsqlcolumnnumber($$$pEXT,%classname,tProperty.Name))
    Set tColumnName = $Get($$$EXTPROPsqlcolumnname($$$pEXT,%classname,tProperty.Name))

    If tColumnNbr '= "" {

      Do %code.WriteLine($Char(9,9,9)_";")
      Do %code.WriteLine($Char(9,9,9)_";")
      Do %code.WriteLine($Char(9,9,9)_"; Audit Field: "_tProperty.SqlFieldName)
      Do %code.WriteLine($Char(9,9,9)_"if {" _ tProperty.SqlFieldName _ "*C} {")
      Do %code.WriteLine($Char(9,9,9,9)_"Set tAudit = ##class(Sample.Audit).%New()")
      Do %code.WriteLine($Char(9,9,9,9)_"Set tAudit.ClassName = """_tClassName_"""")
      Do %code.WriteLine($Char(9,9,9,9)_"Set tAudit.Id = {id}")
      Do %code.WriteLine($Char(9,9,9,9)_"Set tAudit.UserName = tUsername")
      Do %code.WriteLine($Char(9,9,9,9)_"Set tAudit.Field = """_tColumnName_"""")
      Do %code.WriteLine($Char(9,9,9,9)_"Set tAudit.Date = +$Horolog")

      Do %code.WriteLine($Char(9,9,9,9)_"Set tAudit.OldValue = {"_tProperty.SqlFieldName_"*O}")
      Do %code.WriteLine($Char(9,9,9,9)_"Set tAudit.NewValue = {"_tProperty.SqlFieldName_"*N}")

      Do %code.WriteLine($Char(9,9,9,9)_"Set tSC = tAudit.%Save()")
      do %code.WriteLine($Char(9,9,9,9)_"If $$$ISERR(tSC) $$$ThrowStatus(tSC)")

      Do %code.WriteLine($Char(9,9,9)_"}")
    }
    Set tProperty = %compiledclass.Properties.GetNext(.tKey)
  }

  Do %code.WriteLine($Char(9,9)_"}")

  Do %code.WriteLine($Char(9)_"} Catch (tException) {")

  Do %code.WriteLine($Char(9,9)_"Set %msg = tException.AsStatus()")
  Do %code.WriteLine($Char(9,9)_"Set %ok = 0")
  Do %code.WriteLine($Char(9)_"}")

  Set %ok = 1
}

}
```

Data Class (Persistent Class)

这是用户数据类，用户（应用程序）可以在其中进行更改、创建记录、删除记录，以及任何您允许他做的事情。总之，这通常是持久化类。

要开始跟踪和记录更改，您需要从抽象类（Sample.AuditBase）继承这个持久类。

```
 Class Sample.Person Extends (%Persistent, %Populate, Sample.AuditBase)
{
  Property Name As %String [ Required ];
```

```
  Property Age As %String [ Required ];

  Index NameIDX On Name [ Data = Name ];
}
```

测试

由于您从审计抽象类（Sample.AuditBase）继承了数据类（Sample.Person），因此您能够插入数据、进行更改并查看审计类（Sample. Audit）上记录的更改。

如果要测试一下，您需要在Sample.Person类或您选择的任何其他类创建一个Test()类方法。

```
ClassMethod Test(pKillExtent = 0)
{
   If pKillExtent '= 0 {
  Do ##class(Sample.Person).%KillExtent()
   Do ##class(Sample.Audit).%KillExtent()
   }

   &SQL(INSERT INTO Sample.Person (Name, Age) VALUES ('TESTE', '01'))
   Write "INSERT INTO Sample.Person (Name, Age) VALUES ('TESTE', '01')",!
   Write "SQLCODE: ",SQLCODE,!!!

   Set tRS = $SYSTEM.SQL.Execute("SELECT * FROM Sample.Person")
   Do tRS.%Display()

   &SQL(UPDATE Sample.Person SET Name = 'TESTE 2' WHERE Name = 'TESTE')
   Write !!!
   Write "UPDATE Sample.Person SET Name = 'TESTE 2' WHERE Name = 'TESTE'",!
   Write "SQLCODE:",SQLCODE,!!!

   Set tRS = $SYSTEM.SQL.Execute("SELECT * FROM Sample.Person")
   Do tRS.%Display()

   Quit
}
```

运行Test()方法：

```
 d ##class(Sample.Person).Test(1)
```

参数1将消除Sample.Person和Sample.Audit类的内容。

```
d ##class(Sample.Person).Test(1)
INSERT INTO Sample.Person (Name, Age) VALUES ('TEST', '01')
SQLCODE: 0

ID  Age Name
1   01  TEST

1 Rows(s) Affected


UPDATE Sample.Person SET Name = 'TEST ABC' WHERE Name = 'TEST'
SQLCODE:0


ID  Age Name
1   01  TEST ABC

1 Rows(s) Affected
```

测试类方法负责：

- 插入一个名为 “ TEST ” 的新person；
- 显示插入结果；
- 将person “ TEST ” 更新为 “ TEST ABC ”
- 显示更新结果；

现在您可以检查审核日志表。为此，依次打开系统管理门户 -> 系统探索 -> SQL。（记得切换到您的命名空间）

运行以下 SQL 命令并检查结果：
SELECT * FROM Sample.Audit

注意，OldValue是 “ TEST ”，NewValue是 “ TEST ABC ”。现在您可以通过将 “ TEST ABC ” 的名称更改为 “ 您自己的名称 ” 或更改年龄值（举个例子）来自行测试。见：
UPDATE Sample.Person SET Name = 'Fabio Goncalves' WHERE Name = 'TEST ABC'

生成的代码

考虑到您已经实现了下面的审计机制，可以在您的计算机上启动Studio（或Atelier），打开持久类（Sample.Person）并检查编译Sample.Person类后生成的中间代码。为此，您只要按下Ctrl + Shift + V（查看其他源代码）- 检查.INT。然后向下滚动到zSaveAuditAfterExecute标签，查看生成的代码：

优点

通过滚出旧数据实现审计日志记录，很简单。您不需要附加表。维护也很简单。如果您决定删除旧数据，那么这就是一个SQL的问题。

如果您需要在更多表中实现审计日志记录，您只需从抽象类（Sample.AuditBase）继承即可。

根据您的需要进行更改，例如：在流上记录更改。

只记录已修改的字段。不保存已更改的整个记录。

缺点

有这样的问题：当数据更改时，那么整个记录会被复制，也就是没更改的数据也会被复制。

如果表person有一个带有包含图片的二进制数据（流）字段-“ photo ”，那么每次用户更改图片时，流都会被记录（消耗磁盘空间）。

另一个缺点是，增加了每个支持审计日志记录的表的复杂性。请时刻牢记，检索记录不是一件简单的事情。切记必须有条件地使用SELECT子句：“ ...WHERE Status = active ”或考虑添加“ 日期间隔 ”。

考虑事务和回滚。

对于某些应用程序来说，为了提高效率，审计是一个重要的需求。通常，要确定数据更改，应用程序开发人员必须在其应用程序中通过组合使用触发器、时间戳列和附加表来实现自定义跟踪方法。创建这些机制，通常需要做大量的工作来实现，因此导致方案改动，且常常大幅增加性能开销。这是一个简单的例子，可以帮助您开始创建自己的框架。

请阅读 下一篇文章 !

#对象数据模型 #ObjectScript #Caché

源 URL: https://cn.community.intersystems.com/post/%E8%B7%9F%E8%B8%AA%E6%95%B0%E6%8D%AE%E6%9B%B4%E6%94%B9-%E5%AE%A1%E8%AE%A1%E6%97%A5%E5%BF%97-%E4%B8%8A%E7%AF%87