文章
姚 鑫 · 八月 30, 2021 阅读大约需 12 分钟

Caché百讲，前言，Caché 简介，初识M程序，语法规则

第0讲 前言

自我介绍

大家好，我简单的自我介绍一下，我是姚鑫，

为什么开这次的课程

首先Caché，M这门技术相对比较冷门，资料方面都是英文，没有系统的相关资料，记得刚入职时学习的东西都是很基础的，稍微有一些复杂的结构变化，就不知道如何下手。之后，群里的小伙伴总有人问我，希望我出一个系统的Caché视频，这件事我也酝酿了很久，目的就是帮助到更多的同学，让每个小伙伴在日后的工作学习中能更加的得心应手。遇到困难或问题时，大家可以随时在群里咨询，群里的氛围很和谐，有很多热心的大佬，不会说有那种自己觉得简单的问题，不好意思问，没人回答的情况。只要你觉得是个问题就可以发出来。

因为这段时间比较忙，准备的比较仓促，可能有一些内容会有遗漏，大家有啥想法可以群里留言，会后可以答疑，大家可以提出来，一起探讨一下，后续的会准备的更加充分一些。

课程特点

- 本次课程什么时候讲完什么时候结束。
- 全方位无死角覆盖所有知识点，讲解详细，上手容易。
- 内容实际，实用性强，每个知识点都可以找到对应的参考实例。

受众

- 0基础刚接触Caché的同学。
- 有一定接触，但是对一些细节和概念不清楚的同学。
- 想掌握更多编程细节的同学。

效果

- 可以掌握Caché编程所有的知识点，避免对一些冷门只是无从下手。
- 对于一门技术而言，只有对其了解足够深入，才能用最浅，通俗的话语将其描述出来。做到深入浅出就是本次课程的一个主要目标。

如何去学习这门课程

- 每次的课件都会共享到群里。
- 培训视频也会共享到群里。
- 把上课的每一个例子手动的去敲一遍。
- 有一定针对的作业。

第1讲 Caché 简介

- Caché 一种集成对象编程语言的数据库，最底层是多维数组存储，又可以使用对象和sql来访问，也就是说有三套接口来同时瞄准一套数据和元数据，最大化开发灵活性了。
- 国内很多大型三甲医院用了10多年足以见证它的健壮性，国外更加普遍。
- ObjectScript是一种对象编程语言，ObjectScript源代码被编译为在Caché虚拟机中执行的目标代码。该目标代码针对通常在业务应用程序中发现的操作进行了高度优化，包括字符串操作和数据库访问。

可以在以下上下文中使用Caché ObjectScript：

- 从Caché终端的命令行中进行交互。
- 作为Caché对象类方法的实现语言。
- 创建Caché ObjectScript例程：Caché中包含并执行的各个程序。
- 作为Caché SQL中存储过程和触发器的实现语言。
- 作为Caché Server Pages应用程序中的服务器端脚本语言。

Caché ObjectScript是ISO 11756-1999年标准M编程语言的超集。 与ISO标准M相比，Caché ObjectScript提供了许多重大改进。

- 集成了面向对象编程的支持。
- 使用{}语法的过程块和控制块。
- 放宽了空格要求。

特点

CachéObjectScript的一些关键功能包括：

- 强大的内置函数可用于处理字符串。
- 对面相对象的支持，包括方法，属性和多态性。
- 用于在应用程序中直接控制流的各种各样的命令。
- 一组用于处理I / O设备的命令。
- 支持多维数组：局部变量和全局变量（Global）。
- 支持高效的嵌入式SQL。
- 支持间接以及运行时计算和命令执行。

第2讲 初识M程序

语言概述

- 变量名称，类名称和方法名称区分大小写。
- 语句不能从一行的第一个字符位置开始。
- 所有命令都必须缩进。注解也必须缩进。
- 大多数ObjectScript命令(以及许多函数和特殊变量)都有长形式和短(缩写)形式(通常是一个或两个字符)

```
/// d ##class(YX.First).Command()
ClassMethod Command()
{
    set yx = "??"
    write yx,!

    s yx = "??"
```

```
    w yx,!
}
```

- s是set命令的缩写，是赋值定义变量命令，将yx值为姚鑫。
- w是write命令的索引，是显示yx变量的值。

变量

- 在ObjectScript中，变量是可以内存运行时值的位置的名称。也就是说变量存在内存中。
- 必须使用SET命令定义变量，但不必指定类型。也可以理解为泛型。
- 变量是非类型化的；也就是说，它们没有指定的数据类型，可以接受任何数据值。
- 支持的几种变量:
    - 局部变量 - 只有创建它的Caché进程才能访问的变量，该变量在进程终止时自动删除。局部变量可以从任何命名空间访问。

```
s name = "yao xin"
```

    - 进程私有全局变量 - 仅可由Caché进程访问并在进程结束时删除的变量。进程私有全局可以从任何名称空间访问。进程私有全局变量对于临时存储大数据值特别有用。

```
s ^||yx = "yao xin"
```

    - 全局变量 - 存储在Caché数据库中的永久变量。全局可以从任何进程访问，并在创建它的进程终止后持续存在。全局变量特定于单个命名空间。

```
s ^yx = "yao xin"
```

    - 数组变量 - 具有一个或多个下标的变量。所有用户定义的变量都可以用作数组，包括局部变量、进程私有全局变量、全局变量和对象属性。

```
s color("red")  = "red"
s color("yellow")  = "yellow"
s color("green")  = "green"
```

    - 特殊变量(也称为系统变量) 一组特殊的内置变量之一，其中包含Caché操作环境的特定方面的值。Caché定义了所有特殊变量；Caché将所有特殊变量设置为初始值(有时为空字符串值)。一些特殊变量可以由用户设置，其他变量只能由caché设置。特殊变量不是数组变量。

```
w $zversion
w $namespace
```

    - 对象属性 - 与对象的特定实例相关联并存储在其中的值。

```
Property name As %String;

/// d ##class(YX.First).Variable()
ClassMethod Variable()
{
    s mYx = ..%New()
    s mYx.name = "??"
```

```
        w mYx.name,!
    }
```

表达式

- 表达式是可以计算为生成单个值的任何标记集

```
/// d ##class(YX.First).Expression()
ClassMethod Expression()
{
    s name = "yao xin"
    w name,!
    w 1 + 2,!
    w $l(name),!
    w $zversion
}
```

函数

- 函数是执行操作(例如，将字符串转换为其等效的ASCII代码值)并返回值的例程。在命令行中调用函数。此调用向函数提供参数值，该函数使用这些参数值执行某些操作。然后，该函数向调用命令返回单个值(结果)。可以在任何使用表达式的地方使用函数。

(一般例程里叫做函数，类里叫做方法。)

- Caché提供了大量的系统函数，不能修改这些函数。这些函数美元符号(“ $ ”)开头，参数括在括号中；即使没有指定参数，括号也是必需的。

(特殊变量名也以单个美元符号开头，但没有括号。)

- 许多系统提供的函数名称都有缩写。

```
/// d ##class(YX.First).Funciton()
ClassMethod Funciton()
{
    s yx = $length("??")
    w "??" _ yx,!

    s yx = $l("??")
    w "??" _ yx,!
}
```

- 函数总是返回值。通常，此返回值被提供给命令。
- 在一些函数中，不需要为返回值提供接收者。可以用do 或 job命令

```
d $classmethod("YX.First","Command")
```

- 一个函数可以没有参数、单个参数或多个参数。函数参数是位置参数，用逗号分隔。许多参数是可选的。如果省略某个参数，Caché将使用该参数的默认值。由于参数是位置参数，因此通常不能省略指定参数列表中的参数。

```
    s yx = "??????????"

    w "?????????????" _ $e(yx),!

    w "????????" _ $e(yx, 1),!

    w "?????????" _ $e(yx, 1 , 2),!
```

- 通常，参数可以指定为字符串、变量或另一个函数的返回值。在少数情况下，参数必须作为字符串提供。

```
ClassMethod GetStr()
{
    q "??????????"
}


    w "???????" _ $e("??????????",1),!

    w "?????" _ $e(yx, 2),!

    w "???????????" _ $e(..GetStr(), 3),!
```

- 大多数情况下，必须先定义变量，然后才能将其指定为函数参数，否则会生成错误。在少数情况下，不必定义参数变量。

```
    w "???????" _ $d(str),!

    s str = 0

    w "?????" _ $d(str),!
```

- 函数参数是向函数提供值的输入参数。函数不修改作为输入参数提供的变量的值。在少数情况下，函数既返回值又设置输出参数。

```
    s color = "red,green,yellow"
    w "???color???" _ color,!

    s $p(color, ",", 1) = "write"

    w "??color???" _ color,!
```

第3讲 语法规则

区分大小写

- 区分大小写
    - 变量名(局部变量、全局变量和进程私有全局变量)
    - 变量下标
    - 类名
    - 方法名
    - 属性名
    - 属性的实例变量的i%

    - 例程名
    - 宏名
    - 宏包含文件(.inc文件)名
    - 标签名
    - 锁名
    - 嵌入式代码指令标记字符串、
    - 嵌入式SQL主机变量名。
- 不区分大小写

    - 命令名
    - 函数名
    - 特殊变量
    - 命名空间名
    - sql语句
    - 预处理器指令(如#include)
    - 嵌入式代码指令(&html、&js和&sql)

空格

空格视为语法上有意义的。除非另有说明，否则空白指的是空格、制表符和换行符。

不需要空格

- 标签(也称为标记或入口点)：标签必须出现在第一列，前面没有空格字符。如果一行有标签，则标签与该行上的任何代码或注释之间必须有空格。如果标签有参数列表，则在标签名称和参数列表的左括号之间不能有空格。参数列表中的参数之前、之间或之后可以有空格。
- 多行注释：多行注释的第一行前面必须有一个或多个空格。多行注释的第二行和后续行不需要前导空格。

```
 /*
asd */
```

需要空格

- 命令及其第一个参数之间必须且只有一个空格

```
s yx = "yx"
 s yx = "yx"
```

- 如果命令使用后置条件，则命令与其后置条件之间没有空格。

```
 q:yx '= ""
 q:(yx '= "")
```

- 在任何一对命令参数之间可以有任意数量的空格。

```
 if yx = ""       s yx = "yao xin"
```

- 如果一行包含代码，然后是单行注释，则它们之间必须有空格。

```
 s yx = "yx"//??
 s yx = "yx" //??`
```

- 通常，每个命令都显示在自己的行上，不过可以在同一行上输入多个命令。在这种情况下，它们之间必须有空格；

```
yx = "yx"   w "1"
```

- 如果一个命令是无参数的，那么它后面必须跟两个空格。

```
l  w "11"
```

- CachéStudio提供内置的语法检查，因此它将标记任何非法使用的空格。

注释

- /**/ 注释 可以显示在一行内，也可以跨行显示。
- // 注释 指定该行的其余部分是注释；它可以是该行的第一个元素，也可以跟在其他元素之后。
- ; 注释 指定该行的其余部分为注释；它可以是该行的第一个元素，也可以跟在其他元素之后。
- ;; 注释 ;注释类型的特殊类型。（不推荐）
- /// 注释用作紧随其后的类或类成员的类引用内容。对于类本身，类定义开始之前的///注释提供类引用内容的类描述，这也是类的Description关键字的值)。

（注意：因为Caché在目标代码(实际解释和执行的代码)中保留了；；注释，所以包含它们会影响性能，并且它们不应该出现在循环中。）

```
/// ????
/// d ##class(YX.First).Comment()
ClassMethod Comment()
{
    w $p("apple,banana,organe" /* ?? */,",",2),!
    w "apple",/* ??" */" banana",!
    s x = "organe"/* ??" */ WRITE x,!
    w "organe"/* ??" */ // "????"
}
```

```
DHC-APP>d ##class(YX.First).Comment()
banana
apple banana
organe
organe
```

文字值

文字值就是组成变量的表达式，由一系列表示特定字符串或数值的字符组成的常量值

```
    s x = "hello"
    s y = 10
```

字符串

- 最大字符串大小是可配置的。如果启用了长字符串，则最大字符串大小为3，641，144个字符。否则，最大字符串大小为32，767个字符。默认情况下启用长字符串。
- 字符串文字是由引号包含的一组零个或多个字符
- 字符串可以包含任何字符，包括空格和控制字符。字符串文字的长度以字符串中的字符数而不是字节数来度量。

```
s str ="???????"
w $l(str),!
s str ="this is a string"
w $l(str),!
```

- 所有字符串都可以输入的。可以使用$CHAR函数指定不可输入的字符.

```
s str = "abc"
w str,!
zzdump str

s str = $c(945) _ $c(946) _ $c(947)
w str,!
zzdump str
```

- 并非所有字符串字符都可显示。它们可以是非打印字符或控制字符。

```
s str = "a" _ $c(0) _ "b" _ $c(9) _ "c" _ $c(10, 13) _ "d"
w !!,str,!
w $l(str),!
zzdump str
```

- 要在字符串中包含引号字符(“)，请将字符加倍

```
s str = """""""
w !!,str,!
w $l(str),!
zzdump str
```

- 不包含任何值的字符串称为空字符串。它由两个引号字符("")表示。空字符串被视为已定义的值。它的长度为0,空字符串与由空字符($CHAR(0))组成的字符串不同

```
s str = ""
w "????:" _ $l(str),!

s str = $c(0)
w "$c(0)????" _ $l(str),!
```

数字

数字文字值是ObjectScript计算为数字的值。

```
/// d ##class(YX.First).Number()
ClassMethod Number()
{
    s x = +++00008.000
    w "?????", $l(x),!
```

```
    w "????" , x,!
    w "???" , x = 8,!
    w " + 1?" , x + 1,!
}
```

数字字符串表示为用引号括起来的字符串数字。

```
    s x = "+++00008.000"
    w "?????", $l(x),!
    w "????" , x,!
    w "???" , x = 8,!
    w " + 1?" , x + 1,!
```

值	数量
数字0-9	任何数量，但至少有一个。
符号运算符：减号(-)和加号(+)。	任何数量，但必须在所有其他字符之前。
小数分隔符字符(默认情况下为句点或小数点字符；在欧洲区域设置中为逗号字符)。	最多一个。
字母 “ E ” (用于科学记数法)。 科学记数法：用指数记数法表示的数字。	最多一个。必须出现在两个数字之间。

```
    w 3.5E10,!
```

标识符

- 标识符是变量、例程或标签的名称。通常，合法标识符由字母和数字字符组成，标识符区分大小写。
- 标识符的第一个字符可以是百分号(%)字符。以%字符开头的名称保留为系统元素。
- Caché中没有保留字；可以使用任何有效的标识符作为变量名、函数名或标签。
- 最好避免使用命令名、函数名或其他此类字符串的标识符。
- ObjectScript代码包括对嵌入式SQL的支持，避免使用SQL保留字命名任何函数、对象、变量或其他实体。

标签

标签即子例程

标签具有以下命名约定：

- 第一个字符必须是字母数字字符或百分比字符(%)。
- 它们最长可达31个字符。标签可以长于31个字符，但在前31个字符内必须是唯一的。标签引用仅与标签的前31个字符匹配。
- 区分大小写。

```
/// d ##class(YX.First).Number()
ClassMethod Label()
{
yao
XIN
31yao
%xin
}
```

- 可以使用$ZNAME函数来验证标签名称。验证标签名称时，请不要使用参数括号。

```
w $zname("%xin")
w $zname("#%xin")
```

注意：标签提供入口点，但它没有定义封装的代码单元。这意味着一旦标记的代码执行，执行将继续到下一个标记的代码单元，除非执行被停止或重定向到其他地方。

```
/// d ##class(YX.First).Label()
ClassMethod Label()
{
    s x = $random(2)
    if x=0 {
        d label0
        w "label0??",!
        q
    } else {
        d label1
        w "label1??",!
        q
    }
label0
  w "label0? ??",!

  w "label0? ??",!
label1
  w "label1? ??",!

  w "label1? ??",!
}
```

有三种方法可以停止执行代码单元：

- 执行遇到退出或返回 （ QUIT or RETURN ）。
- 执行遇到尝试的右大括号(“ } ”)。发生这种情况时，执行将从关联的CATCH块后面的下一行代码继续执行。
- 执行遇到下一个过程块(带有参数括号的标签)。遇到带括号的标签行时，即使括号内没有参数，执行也会停止。

#Caché

源URL: https://cn.community.intersystems.com/post/cach%C3%A9%E7%99%BE%E8%AE%B2%EF%BC%8C%E5%89%8D%E8%A8%80%EF%BC%8Ccach%C3%A9-%E7%AE%80%E4%BB%8B%EF%BC%8C%E5%88%9D%E8%AF%86m%E7%A8%8B%E5%BA%8F%EF%BC%8C%E8%AF%AD%E6%B3%95%E8%A7%84%E5%88%99