文章
jieliang liu · 九月 22, 2021 阅读大约需1 分钟

IRIS 2021 搜文档 First Look 4 ObjectScript 入门

搜概要：ObjectScript

搜概要：ObjectScript

本文档将向您介绍 ObjectScript 编程语言，并提供了几个示例，说明如何使用它来存储和检索来自 InterSystems IRIS® 数据平台的数据。本搜概要（First Look）并不试图提供对该语言或其功能的全面概述。您可以使用本文件末尾列出的参考资料，继续您的探索。

要浏览所有的搜概要（First Look），包括可以在 InterSystems IRIS 免费的评估实例上执行的那些，请参见 InterSystems First Looks（在 InterSystems 搜概要）。

什么是 ObjectScript？

ObjectScript 是一种编程语言，用于在 InterSystems IRIS 数据平台上快速开发复杂的业务应用。ObjectScript 源代码被编译成 object 代码，该代码可针对业务应用程序中的典型操作（包括字符串操作和数据库访问）进行高度优化。

ObjectScript 的独特之处之一是它的底层存储结构，即所谓的 *globals*。Globals 可以被认为是持久的多维稀疏数组。ObjectScript 允许您直接从 globals 访问数据，但也允许您通过其本机 object 和 SQL 支持来访问这些数据。

虽然您可以在 InterSystems IRIS 平台上使用 Java、.NET、node.js 或其他语言编写应用程序，但通过使用该平台的许多 API，您可以使用 ObjectScript 编写高效的、基于服务器的代码，让您对数据有更精细的控制。

试一试：使用 ObjectScript 存储和检索数据

在这篇搜概要（First Look）中，您将学习如何使用 ObjectScript 来：

- 在 globals 中存储和检索数据。
- 定义一个类，并实例化、使用和存储该类的 objects。
- 使用 SQL 查询访问为该类存储的数据，并对结果集进行处理。

正如您将看到的，ObjectScript 使您能够以多种方式存储和访问数据，既强大又灵活。

使用前须知

要使用本摘要(First Look)，您需一个正在运行的 InterSystems IRIS 实例。您的选择包括几种类型的已授权和免费的评估实例；该实例不需由您正在工作的系统托管(尽管您的系统必须能够通过网络访问该实例)。关于如何部署每种类型的实例的信息，如果您还没有一个实例可以使用，请参见 *InterSystems IRIS Basics:Connecting an IDE*(《*InterSystems IRIS 基础：连接一个 IDE*》)中的Deploying InterSystems IRIS(部署 InterSystems IRIS)。

您也需要知道：

- 实例的基本 web 管理门户(Management Portal)的 URL，这是 InterSystems IRIS 的系统管理用户界面。

试一试：使用 ObjectScript 存储和检索数据

- 如何访问终端(Terminal)，InterSystems IRIS 命令行工具。
- 您的实例的用户名和密码(InterSystems Learning Labs 上的 web 实例不需要)。

您还需要从 InterSystems GitHub repo 下载 ObjectScript 示例代码：https://github.com/intersystems/FirstLook-ObjectScript。

关于如何访问管理门户(Management Portal)或终端(Terminal)的更多信息，请参见 *InterSystems IRIS Basics:Connecting an IDE*(《*InterSystems IRIS 基础：连接一个 IDE*》)中的 "InterSystems IRIS Connection Information(InterSystems IRIS 连接信息)"。尽管您实际上不需要IDE来做这些练习，但您可以使用安装了 ObjectScript 扩展的 Studio 或 Visual Studio Code 查看和编辑示例代码。

导入 ObjectScript 代码示例

首先，将 ObjectScript 代码示例导入 InterSystems IRIS:

0. 从管理门户(Management Portal)的主页，选择 **System Explorer(系统资源管理器)** > **Classes(类)**。
1. 在 Classes 页面上，查看左栏，确保在 USER 命名空间。您可以把命名空间看作是工作空间或目录。
2. 点击 **Import(导入)**。
3. 在 Import Classes 的对话框中：a. 如果您的 InterSystems IRIS 实例在远程服务器上运行，请指定是将示例文件下载到远程服务器还是下载到您的本地计算机。b. 在"File"或 "Directory"的 Import(导入)区域，点击 **File(文件)**。c. 浏览您从 GitHub 下载的 FirstLookObjectScript.xml 文件。d. 选择 **Compile Imported Items(编译导入项)**。e. 对于 **Compile Flags(编译标志)**，指定 cuk。f. 点击 **Next(下一步)**。g. 点击 **Import(导入)**。h. 当出现加载成功的消息时，点击 **Done(完成)**。

现在，在 Classes 页面上，您应该到 FirstLook.ObjectScript.cls 和 FirstLook.Person.cls 在类的列表中。在 InterSystems IRIS 中，包含类的包名称(FirstLook)附加有类的名称(ObjectScript 或 Person)。扩展名 .cls 用来表示类文件。

注意：如果您的命名空间包含大量的类，可以在页面左栏的 **Class Name(类名)** 框中输入 F*.cls 来过滤列表。

ObjectScript 和 Globals

如果您想使用一些 ObjectScript 命令，一个很好的方法是使用 InterSystems 终端(InterSystems Terminal)。(如果您以前没有使用过终端(Terminal)，请参见 *InterSystems IRIS Basics:Connecting an IDE*(《*InterSystems IRIS 基础：连接一个 IDE*》)中的 "InterSystems IRIS Connection Information(InterSystems IRIS 连接信息)"。)

在您启动终端会话(Terminal session)后，提示会指示您处于哪个命名空间。如果您不在 USER 命名空间，执行以下命令：

```
set $namespace = "USER"
```

试一试：使用 ObjectScript 存储和检索数据

在 ObjectScript 中，您可以使用 **set** 命令给变量赋值：

```
USER>set x = "Welcome to ObjectScript!"
```

要显示一个变量的内容，请使用 **write** 命令：

```
USER>write x

Welcome to ObjectScript!
```

您刚刚创建的变量只存在于这个终端会话(Terminal session)的工作内存中。如果您想在数据库中存储一个值，您可以使用 global，它看起来像一个带插入符号(^)的变量：

```
USER>set ^Settings("Color") = "Red"
```

Globals 提供持久存储，这意味着在您关闭终端会话后，^Settings 将继续存在。

这个特殊的 global 也是一个数组，有一个下标。与许多其他语言中的数组不同，globals 可以有整数、小数或字符串等下标。Globals 也是稀疏的，这意味着下标可能是连续的，也可能不是连续的。

您可以利用 globals 的多维特性定义一个更复杂的结构：

```
USER>set ^Settings("Auto1", "Properties", "Color") = "Red"

USER>set ^Settings("Auto1", "Properties", "Model") = "SUV" USER>set ^Settings("Auto2", "Owner") = "Mo"

USER>set ^Settings("Auto2", "Properties", "Color") = "Green"
```

要显示 global 中的所有节点，您可以使用 **zwrite** 命令：

```
USER>zwrite ^Settings

^Settings("Auto1","Properties","Color")="Red"

^Settings("Auto1","Properties","Model")="SUV"

^Settings("Auto2","Owner")="Mo"

^Settings("Auto2","Properties","Color")="Green"

^Settings("Color")="Red"
```

要检索存储在 global 特定节点上的值，您可以使用 **$get()** 函数。如果您试图从不存在的 global 节点检索值，这个函数将返回空字符串，以避免潜在的 *未定义* 错误。

在终端(Terminal)尝试以下操作：

```
USER>set ^testglobal(1) = 8888 USER>set ^testglobal(2) = 9999

USER>set globalValue = $get(^testglobal(1))
```

USER>write "The value of ^testglobal(1) is ", globalValue The value of ^testglobal(1) is 8888

要在 global 中迭代节点，您可以使用 **$order()** 函数，它返回 global 中的一个下标。传入一个下标等于空字符串的 global 节点会导致 **$order()** 返回第一个下标。返回值等于空字符串表示 **$order()** 已经到达最后一个下标。

您可以像编写变量或 global 的值一样编写函数返回的值：

USER>write $order(^testglobal("")) 1

USER>write $order(^testglobal(1)) 2

USER>write $order(^testglobal(2))

USER>

在实践中，您通常会在 ObjectScript 类文件中的 *类方法*中创建一个 **$order()** 循环。在 FirstLook.ObjectScript.cls 中为您提供了以下方法。

试一试：使用 ObjectScript 存储和检索数据

/// Iterate over global ^testglobal ClassMethod Iterate() {

// Start by setting subscript to "" set subscript = ""

// "Argumentless" for loop for {

// Get the next subscript

set subscript = $order(^testglobal(subscript))

// When we get to the end, quit the for loop quit:(subscript = "")

// Otherwise, write the subscript and the value

// stored at ^testglobal(subscript)

write !, "subscript=", subscript, ", value=", ^testglobal(subscript)

}

}

请 注 意，"无参数(argumentless) "的 for 循环没有指定终止条件，如果我们不显式退出该循环，它将永远循环下去。 **Write** 语句中的感叹号告诉 ObjectScript 在编写 global 中每个节点的下标和值之前移动到下一行。

要在终端（Terminal）运行该方法，请输入：

USER>do ##class(FirstLook.ObjectScript).Iterate()

这产生了输出：

subscript=1, value=8888 subscript=2, value=9999

ObjectScript 和 Objects

您可以使用 ObjectScript 来创建具有方法和属性的类。然后您可以将该类的 objects 实例化。示例类 FirstLook.Person.cls 定义了一个类 Person, 然后让您创建该类的实例, 如 person John Smith 或 person Jane Doe。

基本的类定义如下所示:

Class FirstLook.Person Extends %Persistent

{

Property FirstName As %String [Required]; Property LastName As %String [Required];

}

Person 类扩展了内置的 InterSystems IRIS 类 %Persistent, 它允许您访问父类中一些有用的方法, 如 **%New()** 和 **%Save()**。 然后列出该类的属性。在这种情况下, 您只是存储一个 person 的名字和姓氏。

继续使用 终端(Terminal) 创建一个新 person。如果您不在 USER 命名空间, 执行以下命令:

set $namespace = "USER"

要创建一个新的 Person object, 请使用 **%New()** 方法, 该方法返回一个新 person 的 "句柄", 更正式的说法是一个 object 引用, 或 OREF。然后设置新 person 的属性并调用 **%Save()** 方法将新 person 存储在数据库中。

USER>set person = ##class(FirstLook.Person).%New() USER>set person.FirstName = "Sarah"

USER>set person.LastName = "Aarons" USER>set status = person.%Save()

USER>write status 1

试一试: 使用 ObjectScript 存储和检索数据

%Save() 方法返回一个状态, 如果成功的话, 其值为 1。

当您保存一个 object 时, InterSystems IRIS 会自动为您把它储存在一个 global 中。默认的 global 名称是末尾带有 Dappended 的类名, 在本例中是 ^FirstLook.PersonD。

如果显示 global 的内容, 您可以到根节点(没有下标的节点)持有一个 ID, 这个 ID 对存储的每个新对象来说是递增的。Global 的其余部分由 ID 下标。每个 Person 节点包含一个属值列表, 其中列表用 **$lb** 表示, 用于 "列表构建"。

USER>zwrite ^FirstLook.PersonD

^FirstLook.PersonD=1

^FirstLook.PersonD(1)=$lb("","Sarah","Aarons")

您还可以定义 *实例方法*, 它对特定的实例进行操作, 而类方法则是类的泛型方法。例如, FirstLook.Person.cls 包含一个 **WriteName()** 方法, 该方法编写 person 的名字。

/// Given an instance of a person, write person's name Method WriteName() {

write "The name of this person is:" write !, ..FirstName

write !, ..LastName

}

属性方法名称前面的额外点表示当前对象或类。

由于变量 person 目前指的是 Sarah Aarons，您可以这样写她的名字：

USER>do person.WriteName() The name of this person is:

Sarah Aarons

作为练习，创建、存储和编写 Person 类的一些新对象，例如，Andrew Shaw、Peter Shaw 和 Kate Aarons。

ObjectScript 和 SQL

您刚刚已经到了如何将创建的每个 person 存储为 global 中的节点。正如您将在本节中所到的，每个 person 也是表中的一行，可以使用 SQL 进行访问。

InterSystems 提供了几种在 ObjectScript 中使用 SQL 的方法。例如，您可以使用 *类查询*，这基本上是类文件中的 SQL 查询。

在 FirstLook.Person.cls 中，下面的类查询，对类中的所有对象执行 SELECT 命令：

/// Query for all stored names

Query Select() As %SQLQuery [SqlProc]

{

SELECT %ID, FirstName, LastName FROM Person

ORDER By LastName, FirstName

}

要测试该查询，您可以从终端（Terminal）运行它：

USER>do ##class(%ResultSet).RunQuery("FirstLook.Person", "Select")

输出将显示一个列表，其中包含您在前面的练习中创建并存储的每个 person，按姓氏，然后是名字排序：

ID:FirstName:LastName:

4:Kate:Aarons:

1:Sarah:Aarons:

2:Andrew:Shaw:

3:Peter:Shaw:

了解有关 ObjectScript 的更多信息

在实际生活中，您可能会编写一个类方法，比如 FirstLook.Person.cls 中为您提供的方法，将查询的结果放在结果集中，然后遍历这个结果集中的每一行：

/// Run select query and write all names in result set ClassMethod WriteAllNames()

{

```
// Create a new %SQL.Statement object

set stmt = ##class(%SQL.Statement).%New()

// Prepare the class query to execute by passing in

// the ClassName and QueryName.

set status = stmt.%PrepareClassQuery("FirstLook.Person", "Select")

// Handle any errors if $$$ISERR(status) {

do $system.OBJ.DisplayError(status) quit

}

// Execute the query

set resultSet = stmt.%Execute()

// Iterate over results while (resultSet.%Next()) {

// Write person's first and last name

write !, resultSet.%Get("FirstName"), " ", resultSet.%Get("LastName")

}

}
```

在调用 **%PrepareClassQuery()** 后，这个类方法使用 $$$ISERRmacro 来检查错误状态。然后，在执行查询后，代码使用 **resultSet.%Next()** 循环遍历结果集，如果存在另一行，则返回 true。

要在终端（Terminal）运行该方法，请输入：

```
USER>do ##class(FirstLook.Person).WriteAllNames() Kate Aarons

Sarah Aarons Andrew Shaw Peter Shaw
```

正如您所看到的，ObjectScript 为您提供了几个处理数据的选项。使用 globals 可以最大限度地控制数据的存储方式，使用 objects 可以轻松地处理类的单个实例，而 SQL 可以跨表的行进行操作。如何对待数据完全取决于您自己。

了解有关 ObjectScript 的更多信息

使用下面列出的参考资料，了解更多关于 ObjectScript 编程的知识。

- ObjectScript 教程 --- 提供 ObjectScript 语言的交互式介绍。
- Using ObjectScript（使用 ObjectScript）--- 提供 ObjectScript 编程语言的概述和详细信息。
- ObjectScript Reference（ObjectScript 参考资料） --- 提供 ObjectScript 的参考资料。
- Orientation Guide for Server-Side Programming（服务器端编程定向指南） --- 为使用 InterSystems 产品编写服务器端代码的程序员提供基本信息。
- InterSystems ObjectScript Basics（InterSystems ObjectScript 基础） --- 涵盖 ObjectScript 基础的交互式课程。

#ObjectScript #InterSystems IRIS #InterSystems IRIS for Health #文档

源 URL: https://cn.community.intersystems.com/post/iris-2021-%E6%8A%80%E6%9C%AF%E6%96%87%E6%A1%A3-first-look-4-objectscript-%E5%85%A5%E9%97%A8