
文章

[Jingwei Wang](#) · 九月 26, 2021 阅读大约需 6 分钟

IRIS 2021 技术文档 First Look 11 用于.Net的InterSystems IRIS Native API

本文解释了如何使用 InterSystems IRIS Native 从 .NET 应用程序访问 InterSystems IRIS® 数据平台 globals。IRIS Native 还允许您运行 ObjectScript 方法、函数和 Routine。在本文中，您首先将连接到 InterSystems IRIS。然后您将在 InterSystems IRIS 中设置和检索一个 global 节点的值，并在另一个 global 节点上进行迭代。您还将调用 InterSystems IRIS 类方法。所有这些活动都将在 .NET 应用程序中执行。

为了让您体验 IRIS Native，而又不陷入细节困境，本次探索特意设计得很简洁。这些活动被设计成只使用默认设置和功能，这样您就可以熟悉功能的基本原理，而不必处理那些离题或过于复杂的细节。当您把 IRIS Native 引入您的生产系统时，您可能需要做一些不同的事情。请确保不要把这种对 IRIS Native 的探索与真实的情况相混淆！

Globals 简介

InterSystems IRIS

提供了一种易于使用的方法将数据存储在持久的多维数组中。global 是存储在 InterSystems IRIS 物理数据库中的多维数组。在应用程序中，globals 到物理数据库的映射基于当前命名空间，命名空间提供一个或多个物理数据库的逻辑统一视图。例如，要使用一个名为 ^Settings 的 global 将值 "Red" 与键 "Color" 关联起来，打开 InterSystems 终端并输入以下代码：

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

您可以利用 globals 的多维特性来定义一个更复杂的结构：

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

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

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

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

探索 IRIS Native

现在，您已经准备好实验 IRIS Native 了。下面的简短演示将向您演示如何在 .NET 应用程序中使用 IRIS Native。（想试试 InterSystems IRIS .NET 开发和互操作性功能的在线视频演示？请看[.NET 快速入门](#)！

用前须知

要使用该程序，您需要一个安装了 .NET 框架和 Visual Studio 的 Windows 系统，以及一个运行中的 InterSystems IRIS 实例来连接。您对 InterSystems IRIS 的选择包括多种类型的已授权的和免费的评估实例；实例不需要由您正在工作的系统托管（尽管它们必须相互具有网络访问权限）。关于如何部署每种类型的实例的信息（如果您还没有可用的实例），请参见 InterSystems IRIS 基础：连接一个 IDE 中的部署 InterSystems IRIS

。使用同一文档中的 InterSystems IRIS 连接信息和 .Net IDE 中的信息将 Visual Studio 连接到您的 InterSystems IRIS 实例。

配置 Visual Studio 项目

首先，打开 Visual Studio 并创建一个新的控制台应用程序项目，选择 Visual C# 和 Console App (.NET Framework)（控制台应用程序 (.NET 框架)）选项。对于 Name（名称）字段，输入 netnative。

添加程序集引用

InterSystems.Data.IRISClient.dll 程序集必须存在于您的本地系统中；您可以从 <https://github.com/intersystems/quickstarts-dotnet/tree/master/EFPlay/bin/Debug> 下载它。如果 InterSystems IRIS 安装在您的本地系统或您可以访问的另一个系统上，则该程序集已经安装在子目录 install-dir\dotnet\bin\4.5 中，其中 install-dir 是该实例的安装目录。

要将 InterSystems.Data.IRISClient.dll 的程序集引用添加到项目：

1. 从 Visual Studio 主菜单中，选择 Project（项目） > Add Reference（添加引用） ...
2. 在出现的窗口中，点击 Browse（浏览） ...
3. 浏览到 InterSystems.Data.IRISClient.dll 文件的位置。
4. 选择文件并点击 Add（添加）。
5. 点击 OK（确定）。

在 Visual Studio Solution Explorer 中， InterSystems.Data.IRISClient.dll 程序集现在应该列在 Reference（引用）下。

使用 IRIS Native

现在，您已经准备好实验 IRIS Native 了。打开在创建 Visual Studio 项目时创建的文件（例如，Program.cs）。删除文件的默认内容并粘贴以下代码，用您的 InterSystems IRIS 实例的连接信息替换 conn.ConnectionString 中的值。（logfile 的值必须是本地系统上的可写路径）。您可以指定所示的 USER 命名空间，也可以选择在该实例上创建另一个命名空间。

```
using System;

using InterSystems.Data.IRISClient;

using InterSystems.Data.IRISClient.ADO;

public class IRISNative

{

    public static void Main(String[] args)

    {
```

```
try
{
// open connection to InterSystems IRIS instance using connection string
IRISConnection conn = new IRISConnection();

// edit this ConnectionString to match your environment

conn.ConnectionString = "Server=localhost; Port=1972; Namespace=User;
Password=SYS; User

ID=system ;

logfile=c:\\sandbox\\dbnative.log";

conn.Open();

// create IRIS Native object

IRIS iris = IRIS.CreateIRIS(conn);

Console.WriteLine("[1. Setting and getting a global]");

// setting and getting a global

// ObjectScript equivalent: set ^testglobal("1") = 8888 iris.Set(8888, "^testglobal",
"1");

// ObjectScript equivalent: set globalValue = $get(^testglobal("1")) Int16?
globalValue = iris.GetInt16("^testglobal", "1");

Console.WriteLine("The value of ^testglobal(1) is
" + globalValue); Console.WriteLine();

Console.WriteLine("[2. Iterating over a global]");

// modify global to iterate over

// ObjectScript equivalent: set ^testglobal("1") = 8888

// ObjectScript equivalent: set ^testglobal("2") = 9999 iris.Set(8888, "^testglobal",
"1");

iris.Set(9999, "^testglobal", "2");

// iterate over all nodes forwards Console.WriteLine("walk forwards");
```

```
IRISIterator subscriptIter = iris.GetIRISIterator("^testglobal"); foreach (var node in
subscriptIter)

{

Console.WriteLine("subscript=" + subscriptIter.CurrentSubscript + ",
value=" + node);

}

Console.WriteLine();

Console.WriteLine("[3. Calling a class method]");

// calling a class method

// ObjectScript equivalent: set returnValue = ##class(%Library.Utility).Date(5)
String returnValue = iris.ClassMethodString("%Library.Utility", "Date", 5);
Console.WriteLine(returnValue);

Console.WriteLine();

// close IRIS object and connection iris.Close();

conn.Close();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}}}
```

示例代码分为三个部分：

1. 第一部分展示了如何设置一个 global 的值以及稍后如何检索它。这部分里面执行的命令等同于 ObjectScript 的 SET 和 GET 命令。
2. 第二部分展示了如何迭代 global 的子节点，类似于 ObjectScript 中的 \$ORDER 函数。
3. 第三部分展示了如何使用 IRIS Native 从您的 .NET 应用程序调用 ObjectScript 类方法。

如果您使用的是本地安装的实例，并且因此连接使用 localhost 作为服务器地址，那么程序可能使用本地共享内存连接，这比标准的 TCP/IP 连接还要快。

注意：ObjectScript 中的 Globals 以插入符号 (^) 开头。在使用 InterSystems IRIS Native API 的 .NET 应用程序中，这不是必需的。

运行练习

现在可以按 Ctrl+F5 运行演示应用程序了。如果示例执行成功，您应该会看到带有示例代码结果的打印输出：

[1. Setting and getting a global] The value of ^testglobal(1) is 8888

[2. Iterating over a global] walk forwards

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

[3. Calling a class method] Oct 30, 2018

在管理门户（Management Portal）中确认变更

接下来，使用以下程序在管理门户（Management Portal）中确认您的结果：

1. 使用InterSystems IRIS 基础：连接一个 IDE中为您的实例描述的 URL，在浏览器中打开您的实例的管理门户（Management Portal）。
2. 如果您不在代码指定的命名空间中，请切换到该命名空间。
3. 导航到 Globals 页面(System Explorer（系统资源管理器） > Global)。您应该会看到在示例代码中创建的 testglobal。点击 VIEW 来查看其内容。您应该会看到 global 的两个节点：^testglobal(1) = 8888 和 ^testglobal(2) = 9999。

了解有关 IRIS Native 的更多信息

有关 IRIS Native、globals 和 InterSystems IRIS 的更多信息，请参见：

用于.Net的 Native API

技术概要:用于Java的 InterSystems IRIS Native API

使用 Globals

用于.NET 的 InterSystems Managed Provider

[#.NET #API #InterSystems API管理器 \(IAM\) #Ensemble #InterSystems IRIS for Health](#)

源

URL:

<https://cn.community.intersystems.com/post/iris-2021-%E6%8A%80%E6%9C%AF%E6%96%87%E6%A1%A3-first-look-11-%E7%94%A8%E4%BA%8E%E7%9A%84intersystems-iris-native-api>