文章
Nicky Zhu · 九月 7, 2021 阅读大约需 9 分钟

IRIS 2021 技术文档 First Look 12--技术概要：.NET Object Persistence with XEP

本文档介绍了 XEP API，它在 InterSystems IRIS®数据平台上极其快速的 .NET 对象存储和检索支持。它向您提供了一个关于 .NET 对象持久化的 XEP 方法的进阶概述，并引导您通过一个简单的场景来演示 API 的主要功能。

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 XEP 的基本原理，而不必处理超出本概述范围的细节。有关 XEP 的完整文档，请参见 *Persisting .NET Objects with InterSystems XEP*（《*使用 InterSystems XEP 持久化 .NET 对象*》）。

要浏览所有的技术概要（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（《InterSystems 技术概要》）。

快速的对象存储和检索

面向对象编程是 .NET 框架的核心，因此 .NET 应用程序将数据建模为对象是很自然的。然而，当应用程序需要将数据存储在数据库中时，这可能会导致问题。如果您使用 ADO.NET 来存储和检索对象，您就面临着将对象数据转换为一组关系表、然后再将查询结果集转换为对象的问题。这个问题的通常解决方案是使用对象-关系映射（ORM）框架，如实体框架（Entity Framework）来自动化这个过程。InterSystems IRIS 确实提供了实体框架（Entity Framework）接口，InterSystems 推荐它用于大型、复杂的对象层次结构。另一方面，对于执行实时数据采集等任务的应用程序，主要问题是速度而不是数据复杂性。虽然实体框架（Entity Framework）绝对不慢（如果适当优化的话），但对于需要极快地存储和检索简单或中等复杂数据的任务来说，XEP 是一个更好的选择。在大多数情况下，它比实体框架（Entity Framework）或 ADO.NET 快得多。

XEP 是如何工作的？

XEP 是一个轻量级的 .NET API，它将 .NET 对象数据作为*持久化事件（persistent event）*来投射。持久化事件（*persistent event*）是 InterSystems IRIS 类的一个实例（通常是 %Persistent 的子类），包含 .NET 对象中数据字段的副本。与任何此类实例一样，可以通过对象访问、SQL 查询或直接 global 访问来检索它。

在创建和存储持久化事件（*persistent event*）之前，XEP 必须分析相应的 .NET 类，并将*模式（schema）导入*数据库。模式（schema）定义了用于存储 .NET 对象的持久化事件类的结构。如果持久化事件类不存在相应的 ObjectScript 模式，则导入模式（schema）将自动为其创建相应的 ObjectScript 模式。来自分析的信息可能是 XEP 导入简单模式所需要的全部信息。对于更复杂的结构，您可以提供额外的信息，允许 XEP 生成索引并覆盖导入字段的默认规则。 在为类创建模式（schema）之后，您可以使用各种 XEP 方法来存储、更新或删除事件、运行 SQL 查询，并通过查询结果集进行迭代。

试一试！XEP 的实际操作

现在是您自己尝试 XEP 的时候了。这个 XepSimple 演示是一个非常小的程序，但它提供了大多数关键 XEP

功能的示例，并概述了如何使用 XEP API。(想试试 InterSystems IRIS .NET 开发和互操作性功能的在线视频演示吗？请看.NET QuickStart（.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（如果您还没有可使用的实例），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）。使用同一文档中的 InterSystems IRIS Connection Information（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（console）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 netxep。(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 QuickStart（.NET 快速入门）！)

添加程序集引用

XEP API 被打包到 InterSystems.Data.XEP.dll 库中，该库必须安装在您的本地系统上。您可以通过克隆 repo https://github.com/intersystems/quickstarts-dotnet/tree/master/XEP 或从该 repo 下载文件来获取它。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\v4.5 中，其中 *install-dir* 是实例的安装目录。

要将 InterSystems.Data.XEP.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.XEP.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.XEP.dll 程序集现在应该列在 **References（引用）** 下。

添加示例类

在创建程序源文件之前，您将添加一个类，主程序会访问该类以生成用于存储的示例对象。在 Visual Studio 中，使用 **Project（项目）** > **Add Class（添加类）** 添加一个 C# 类文件到 **netxep** 项目。删除类文件的默认内容，并将以下代码复制并粘贴到文件中。

```
using System; namespace xep.samples

{
```

```
public class SingleStringSample { public string name;

public SingleStringSample() { }

       internal SingleStringSample(string str) { name = str;}

public static SingleStringSample[] generateSampleData(int objectCount) { SingleString
Sample[] data = new SingleStringSample[objectCount];  for (int i = 0; i < objectCount
; i++)

{

data[i] = new SingleStringSample("single string test");

}

return data;

}

}

}
```

因为这个类非常简单，所以 XEP
不需要任何注释就可以从中生成有效的模式。对于真实应用程序中更复杂的类，您可以使用各种选项来优化导入。

创建 XEP 演示程序

现在您可以创建 XEPSimple 演示程序了。在 Visual Studio 中，找到您在创建项目时创建的默认 program.cs 文件。删除该文件的默认内容，并粘贴下面的代码，用connection information for your InterSystems IRIS instance（您的 InterSystems
IRIS 实例的连接信息）替换主机、端口、irisnamespace、用户名和密码变量的值。您可以指定所示的 **USERnamespace**，也可以选择在实例上创建的另一个命名空间。

```
using System;

using InterSystems.XEP; using xep.samples;

namespace XepSimpleNamespace

{

public class XepSimple

{

public static void Main(string[] args)

{

// Generate 12 SingleStringSample objects for use as test data SingleStringSample[] s
ampleArray = SingleStringSample.generateSampleData(12);

// EventPersister
```

```
EventPersister xepPersister = PersisterFactory.CreatePersister();

String host = "127.0.0.1"; // InterSystems IRIS host int port = 51774; // InterSystem
s IRIS Superserver port

String irisnamespace = "USER"; // InterSystems IRIS namespace String username = "_sys
tem"; // Credentials for InterSystems IRIS String password = "SYS"; //Credentials for
 InterSystems IRIS

xepPersister.Connect(host,port,irisnamespace,username,password); // connect to localh
ost xepPersister.DeleteExtent("xep.samples.SingleStringSample");   // remove old test
 data xepPersister.ImportSchema("xep.samples.SingleStringSample");   // import flat s
chema

// Event

Event xepEvent = xepPersister.GetEvent("xep.samples.SingleStringSample");

long[] itemIdList = xepEvent.Store(sampleArray); int itemCount = 0;

for (int i = 0; i < itemIdList.Length; i++)

{

if (itemIdList[i] > 0) itemCount++;

}

Console.WriteLine("Stored " + itemCount + " of " + sampleArray.Length + " events");



// EventQuery

EventQuery<SingleStringSample> xepQuery = null;

String sqlQuery = "SELECT * FROM xep_samples.SingleStringSample WHERE %ID BETWEEN ? A
ND ?";

xepQuery = xepEvent.CreateQuery<SingleStringSample>(sqlQuery); xepQuery.AddParameter(
3); // assign value 3 to first SQL parameter xepQuery.AddParameter(12); // assign val
ue 12 to second SQL parameter xepQuery.Execute();   // get resultset for IDs between
3 and 12

xepQuery.Close(); xepEvent.Close(); xepPersister.Close();

} // end main()

} // end class xepSimple

}
```

演示程序分为三个部分，每个部分演示了 XEP 三个主要类——EventPersister、 Event 和 EventQuery 中的一个。运行时，XepSimple 演示程序执行以下任务：

- 首先，通过调用示例数据类（sample data class） xep.samples.SingleStringSample 这一方法生成一些示例对象。

- EventPersister *是 XEP 的主要入口点（main entry point），并提供到数据库的连接。*

它创建一个名为 *xepPersister* 的实例，该实例在数据库中建立到 User 命名空间的 TCP/IP 连接，并删除任何现有的示例数据。然后调用 **ImportSchema()** 来分析示例类并将模式发送到数据库，从而创建相应的 ObjectScript 模式来保存持久化 SingleStringSample 对象。

- Event *封装了（encapsulate）.NET 对象和相应的数据库对象之间的接口。*

一旦生成了模式(schema)，*xepPersister* 就可以为示例类创建一个名为 *xepEvent* 的 Event 对象。**Store()** 方法将 .NET 对象数组存储为持久化事件。

- EventQuery *用于准备和执行 SQL 查询的有限子集。*

名为 *xepQuery* 的 EventQuery<SingleStringSample> 对象是通过将查询字符串传递给 *xepEvent* 对象的 **CreateQuery()** 方法创建的。 该字符串定义了一个接受两个参数（? 字符）的 SQL 查询。 参数值是通过调用 **AddParameter()** 定义的，调用 **Execute()** 来获取查询结果集。

- 当处理完成后，它通过调用 XEP 对象的 **close()** 方法进行清理。

1. 运行 XEPSimple 程序

现在您可以构建和运行 XEPSimple 演示程序了。按 Ctrl + F5 运行程序，以便当程序结束时命令提示符保持打开状态。

1. 下一步

XepSimple 演示的目的是让您体验 XEP，而又不陷入细节困境，它不是用于实际生产代码的模型——它甚至没有进行异常检查。最重要的简化是在示例数据中。您从一个不需要注释或其他机制来帮助模式（schema）生成和优化的类中持久化了一些很小的 .NET 对象。虽然在实际应用程序中通常需要注释和其他选项，但这些选项非常简单易用。API 也同样简单易用。

有关所有 XEP 功能的全面描述，请参见 *Persisting .NET Objects with InterSystems XEP*（*《使用 InterSystems XEP 持久化 .NET 对象》*）。

了解有关 .NET 支持的更多信息

InterSystems IRIS 提供了 .NET API，通过 SQL 表、对象和多维存储轻松访问数据库。有关每种类型的访问的详细信息，请参见以下书籍：

- *Using the Native API for .NET*（*《使用 Native API for .NET》*），了解从 .NET 应用程序访问 InterSystems

IRIS globals。

- *Using ADO.NET Managed Provider Classes（《使用 ADO.NET Managed Provider Classes》）*，了解SQL 表访问。允许您的 .NET 项目使用完全兼容的通用 ADO.NET Managed Provider 版本访问 InterSystems IRIS 数据库。
- *Using the InterSystems ODBC Driver（《使用 InterSystems ODBC Driver》）*，了解 SQL 表访问。InterSystems ODBC 驱动程序允许 InterSystems IRIS 建立到外部应用程序的 ODBC 连接，并通过 SQL 提供对外部数据源的访问。
- *Persisting .NET Objects with InterSystems XEP（《使用 InterSystems XEP 持久化 .NET 对象》）*，了解对象访问。XEP 对事务处理应用程序进行了优化，这类程序使用复杂度从简单到中等的对象层次结构，且要求极高的对象数据持久化和检索速度。
- *Using the Entity Framework Provider（《使用 Entity Framework Provider》）*，了解对象-关系映射。提供有关使用实体框架（Entity Framework）技术访问 InterSystems IRIS 数据库的信息。

#InterSystems IRIS

源 URL: https://cn.community.intersystems.com/post/iris-2021-%E6%8A%80%E6%9C%AF%E6%96%87%E6%A1%A3-first-look-12-%E6%8A%80%E6%9C%AF%E6%A6%82%E8%A6%81%EF%BC%9Anet-object-persistence-xep