Научна конференция „Иновационни ИКТ в научните изследвания и обучението: математика, информатика и информационни технологии“, 29-30 ноември 2018 г., Пампорово, България
Scientific Conference “Innovative ICT in Research and Education: Mathematics, Informatics and Information Technologies”, 29-30 November 2018, Pamporovo, Bulgaria

USING FIREBIRD EMBEDDED WITH NET CORE 2 AND ENTITY FRAMEWORK CORE FOR CLOUD-BASED SYSTEMS

Dimitar Nikolov, Nikolay Pavlov, Asen Rahnev

Abstract. This article describes how Firebird Embedded [2] can be used as local, maintenance free database for embedded and distributed systems built on Net Core. We present a proof of concept project implemented to verify the performance of Firebird Embedded [2] in distributed embedded systems. Source code snippets are provided.

Keywords: *IoT, Firebird, Entity Framework, .Net Core*

1. Introduction

When building embedded and distributed systems it is necessary to provide independent, reliable nodes. Common designs of distributed systems contain either a mesh of standalone nodes, which communicate between each other or a centralized network of nodes with one master node. In order to ease user experience the designed system will have one master node (accessible through a web page) and several slave nodes which should gather information from nearby IoT devices through Bluetooth, WiFi or RF communication. Slave nodes will store information locally in order to reduce load to master node by making data validations, send automated command to devices and others.

This article describes proof of concept project implemented to verify that Firebird Embedded [2] could be used as local, maintenance free database for embedded and distributed systems [5]. There are insufficient resources on the topic of integrating Firebird [2] with .Net Core [1]. Online materials and documentation are really poor when it comes to the embedded version of Firebird [3]. Although there are nuget packages for .Net Core user will be surprised to find out that they don’t work out of the box for the embedded version. Additional setup is required to achieve reliable database access and this setup is described later on.

Дата на получаване: 03.12.2018 г.
Дата на рецензиране: 29.03.2019 г.
Дата на публикуване: 17.05.2019 г.

2. Functionality

Application should be able to store and retrieve data from Firebird [2] database using Entity Framework [4] and it is Object-Relational Mapping (ORM) features. The main focus is on CRUD actions and simple queries.

```
    Running in x86 environment:
    C:\Program Files (x86)\dotnet> ./dotnet.exe
'C:\publish\fb\FirebirdEmbeddedDemo.dll'
    Running in x64 environment:
    C:\Program Files\dotnet> ./dotnet.exe
'C:\publish\fb\FirebirdEmbeddedDemo.dll'
    Running in Linux x64 environment:
    dotnet FirebirdEmbeddedDemo.dll
```

3. Technical description

In order to proof integration, simple console application was created targeting .Net Core 2.1 [1]. The project references two nuget packages:

```
FirebirdSql.Data.FirebirdClient
```

This package provides all necessary classes to connect and use Firebird [2] database from .Net Core [1] application.

```
FirebirdSql.EntityFrameworkCore.Firebird
```

This package inherits Entity Framework Core [4] classes and provides extensions to work with Firebird [2] databases from .net application as you would do it with Microsoft Sql Server.

Creating database

Database creation was performed with isql [5] tool from Firebird [2]. Start isql [5] run

```
CREATE DATABASE 'D:\data\test.fdb' page_size 8192
```

then

```
CON>user 'SYSDBA' password 'masterkey';
```

The first row creates database in the specified location with the specified page_size. The second command connects to the database using user and password.

Creating schema:

```
create table "Todos" (
  "Id"  integer generated by default as identity primary key,
  "Name" varchar(50) not null,
  "Description" varchar(500) );
```

The schema was used just to verify usability.

Creating dotnet core project

Creating project with visual studio

There are many guides on this topic. In this case dotnet core Console Application targeting v2.1 was created.

Installing required nuget packages

```
FirebirdSql.Data.FirebirdClient
FirebirdSql.EntityFrameworkCore.Firebird
Microsoft.EntityFrameworkCore
```

Download required Firebird version

From Linux setup

```
sudo ln -sf /usr/lib/x86_64-linux-gnu/libtommath.so.1 
/usr/lib/x86_64-linux-gnu/libtommath.so.0
```

https://www.ibphoenix.com/files/Embedded_fb3.pdf

Detect Operating System and Architecture

The application cannot load single library file and work with it, because Firebird assemblies are platform dependant. This requires the application to load required library files on startup based on operating system. In order to achieve that we will be using RuntimeInformation class provided by .NET Core Framework [1].

Checking for operating system:

```
RuntimeInformation.IsOSPlatform(OSPlatform.Windows)
```

or

```
RuntimeInformation.IsOSPlatform(OSPlatform.Linux)
```

The method IsOSPlatform returns boolean if the platform matches the provided parameter.

To find which CPU architecture is on the current platform we can use a property of the class RuntimeInformation

```
RuntimeInformation.ProcessArchitecture == Architecture.X64
```

The property returned type is Enum having members x86, x64, Arm, Arm64.

Prove

In order to prove that the setup with Firebird Embedded [2] and Entity Framework [4] works we will create sample context with one table in the model. The model will use the previously created table Todos.

Model

Todos class is pretty simple. It is good to have the table name with attributes when using Firebird client tools. Firebird client tools assumes that default for Firebird naming convention is used which is all names are with uppercase letters.

```
[Table("Todos")]
public class Todos
{
     [Key]
     public int Id { get; set; }

     [Required]
     [StringLength(50)]
     public string Name { get; set; }

     [StringLength(500)]
     public string Description { get; set; }
}
```

Context

The defined context is very simple.

```
public class DemoContext : DbContext
{
  private readonly string _connectionString;

  public DemoContext(string connectionString)
  {
     _connectionString = connectionString;
  }

  protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
  {
     optionsBuilder.UseFirebird(_connectionString);
  }

  protected override void OnModelCreating(ModelBuilder builder)
  {
     base.OnModelCreating(builder);
  }

  public virtual DbSet<Todos> Todos { get; set; }
}
```

Usage

Simple data saving and retrieving operations are executed when the console application runs. Here is it:

```
DemoContext demoContext = new DemoContext(connectionString);

for (int i = 0; i < 10; i++)
{
   demoContext.Todos.Add(new Todos()
   {
      Name = "Blah blah",
      Description = "describe"
   });
}
demoContext.SaveChanges();

foreach(var todo in demoContext.Todos)
{
   Console.WriteLine($"ID: {todo.Id}   NAME:{todo.Name}
   DESCRIPTION:{todo.Description}");
}
```

4. Conclusion

This prove of concept project required deep learning of poorly documented Firebird [2] features and how they work under different operating systems. However, the result is worth all efforts, because at the end Firebird [2] was proven to work with .Net Core [1] under all operating systems. This gives the designed system easily maintainable relation database to use in its distributed nodes.

Acknowledgements

This paper is partially supported by project FP17-FMI-008 of the Scientific Research Fund of Plovdiv University “Paisii Hilendarski”, Bulgaria.

References

[1] Microsoft, ASP.NET Core Documentation, 2018, https://docs.microsoft.com/en-us/aspnet/core/

[2] Firebird, Firebirdsql 3.0.3 Documentation 2018, http://www.firebirdsql.org/en/documentation/

[3] Firebird, Firebirdsql https://www.firebirdsql.org/pdfmanual/html/ufb-cs-embedded.html

[4] Entity Framework Core

https://docs.microsoft.com/en-us/ef/core/

[5] Firebird, Firebirdsql 3.0.3 Documentation 2018,
https://www.firebirdsql.org/pdfmanual/html/qsg10-creating.html
http://www.firebirdfaq.org/Firebird-Embedded-Linux-HOWTO.html
https://www.firebirdsql.org/pdfmanual/html/ufb-cs-embedded.html
http://www.ibphoenix.com/files/Embedded_fb3.pdf

[6] Nikolov D., N. Pavlov, A. Rahnev, Home IoT Monitoring and Management System, Proc. of Scientific Conference "*Innovative Software Tools and Technologies with Applications in Research in Mathematics, Informatics and Pedagogy of Education*", 23-24 November 2017, Pamporovo, Bulgaria, pp. 25–32, ISBN: 978-619-202-343-0.

Faculty of Mathematics and Informatics
Plovdiv University "Paisii Hilendarski"
236 Bulgaria Blvd, 4003 Plovdiv, Bulgaria
e-mail: d.nikolov@windowslive.com, nikolayp@uni-plovdiv.bg,
asen@uni-plovdiv.bg

ИЗПОЛЗВАНЕ НА FIREBIRD EMBEDDED С NET CORE 2 И ENTITY FRAMEWORK CORE ЗА СИСТЕМИ В ОБЛАКА

Димитър Николов, Николай Павлов, Асен Рахнев

Резюме. Статията описва как можем да използваме Firebird Embedded като локална база данни, които не се нуждае от администрация, във вградени и разпределени приложения, на Net Core. Представен е прототип, който да провери производителността на Firebird Embedded в разпределени системи. Представен е изходен код.