الجمهورية العربية السورية

الجامعة الافتراضية السورية

ماجستير علوم الويب

وظيفة مقرر ADP

REMOTE DATABASE ACCESS

إعداد

م.مؤيد علي (شعبة C3)

mouiad_197974

م.مازن سودا(شعبة C3)

mazen_198082

م.عبدالله غزيل(شعبة C3)

abdullah_198778

إشراف

د.سيرا ستور

APRIL 20, 2022

فهرس المحتويات

الهدف

تطوير برنامج موزع يعمل على تقديم خدمة الوصول الى قاعدة بيانات بعيدة والقراءة منها)اما قراءة سطر، أو عدة أسطر، أو عامود، أو عدة أعمدة) وتقديمها للزبائن بشكل بعيد باستخدام تقانات البرمجيات الموزعة.

1. مقدّمة

NET Remoting. هي البنية التحتية الجديدة في Microsoft التي توفر مجموعة غنية من الصفوف التي تسمح للمطورين بتجاهل معظم تعقيدات نشر وإدارة الكائنات البعيدة. في NET Remoting. يتميز استدعاء الطرق على الكائنات عن بعد بأنه مطابق لاستدعاء الطرق المحلية.

Remoting هو إطار مدمج في وقت تشغيل اللغة المشتركة common language runtime (CLR) يمكن استخدامه لبناء تطبيقات موزعة متطورة وخدمات الشبكة. عندما يقوم العميل بإنشاء غرض كائن بعيد، فإنه يتلقى وكيل(proxy) إلى غرض الفصل على الخادم بالتالي سيتم إرسال جميع الطرق التي تم استدعاؤها بالوكالة تلقائيا إلى الفئة البعيدة وسيتم إرجاع أي نتائج إلى العميل ومن وجهة نظر العميل، لا تختلف هذه العملية عن إجراء استدعاء طريقة محلية.

لاستخدامNET Remoting. لبناء تطبيق يتصل فيه مكونان مباشرة عبر حدود مجال التطبيق، نحتاج إلى إنشاء ما يلي فقط:

- كائن بعيد.
- مجال تطبيق مضيف(مخدم) وذلك للاستماع والحصول على طلبات لهذا الكائن.
- مجال تطبيق العميل(الزبون) الذي ينشئ الطلبات لهذا الكائن.

سيكون لدينا في نهاية هذا البحث:

1- الكود الكامل للبرنامج الموزع ، مع إصدار قابل للتنفيذ وملف نوتة يتضمن مواصفات تشغيل البرنامج على أي جهاز

2- يجب أن يكون لديك ثلاث أجزاء منفصلة وهي: الواجهة والمخدم والعميل.

لنبدأ المثال، أولاً سنقوم بإنشاء كائن بعيد والوصول إليه باستخدام واجهة فقط.

2. الجزء الأول: إنشاء مكتبة الواجهة Interface_customerinfo

ننقر فوق ملف-> مشروع جديد> ثم نختار إنشاء "مكتبة C # جديدة" جديدة ونسميها interface_customerinfo ثم ننقر فوق "موافق". سيؤدي ذلك إلى إنشاء "المفردات المشتركة" التي سيستخدمها كلا من عميلنا .NET البعيد والخادم للتواصل وبالتالي أي نشاط قاعدة بيانات يحدث على جانب الخادم. الشيء المهم أن نلاحظ هنا هو أننا نحدد كائنICUSTOMERINFO كواجهة.

تعرض هذه الواجهة 3 طرق ويتم العمل الفعلي بواسطة الخادم، ولكن هذا شفاف (غير مرئي) للعميل أي ما ينظر إليه العميل هو الطرق والواجهة هي التي تعرض هذه الطرق فقط.

ويكون كود الواجهة كاملا كالآتي:

```
using System;
namespace Interface_CustomerInfo {
    /// Summary description for Class1.
    public interface ICustomerInfo {
        void Init(string username, string password);
        bool ExecuteSelectCommand(string selCommand);
        string GetRow();
    }
}
```

نقوم ببناء وترجمة المشروع لإنشاء Interface_CustomerInfo.DLL.

3. الجزء الثاني: إنشاء الخادم (CustomerServer):

ننقر فوق ملف–> مشروع جديد حيث نقوم بإنشاء تطبيق Windows بسيط أو تطبيق وحدة تحكم بسيط.

نحتاج إلى إضافة مرجع إلى Interface_CustomerInfo.DLL لأنه سيستفيد من الواجهة ويتم ذلك عن طريق النقر باليمينية فوق المشروع–> إضافة مرجع ، وأضف مرجعًا إلى ملف DLL الذي أنشأناه في الجزء الأول بالنقر فوق الزر "استعراض".

من أجل استخدام فعالية .NET عن بُعد ، يجب إضافة مرجع إلى System.Runtime.Remoting.DLL باستخدام Project-> Add Reference ضمن علامة التبويب .NET.

في البداية يجب علينا إنشاء قناة لتحقيق الاتصال من خلالها حيث أن Net Remoting. وهو يدعم نوعين من القنوات. بالنسبة للتطبيقات المحلية أو تطبيقات الشبكة الداخلية، يمكننا استخدام TcpChannel للحصول على أداء أفضل أما النوع الآخر HttpChannel فيمكن استخدامه لتطبيقات الإنترنت وفي هذا المثال يكفينا أن نستخدم TcpChannnel كونه محلي أما إذا كنت تعمل عبر الإنترنت ، يمكن أن يكون HTTP في بعض الأحيان هو الخيار الوحيد وذلك اعتماداً على تكوينات جدار الحماية firewall لدى المستخدم، وهذا سيعمل على إعداد رقم المنفذ port الذي نريد أن يستجيب كائننا للطلبات الموجودة عليه و ChannelServices.RegisterChannel ستربط رقم المنفذ هذا بالمكدس الموجود على نظام التشغيل.

وتكون تعليمتي التصريح عن القناة وتسجيلها كالتالي:

```
hts = new HttpServerChannel(8228);

ChannelServices.RegisterChannel(hts);
```

أما الآن فيجب تسجيل نوع الخدمة وهي التي تحدد نمط آلية الاستجابة عند استدعاء طريقة من طرق الكائن البعيد كما يلي:

```
RemotingConfiguration.RegisterWellKnownServiceType(typeof(CUSTOMER_
SERVER1) ,"CUSTOMER_SERVER1" , WellKnownObjectMode.Singleton);
```

البارامتر الأول هي الكائن الذي تقوم بربطه،typeof (CUSTOMER_SERVER1) والبارامتر الثاني هي السلسلة التي تمثل اسم الكائن على قناة TCP أو HTTP. على سبيل المثال، قد يشير العملاء البعيدون إلى الكائن

أعلاه كـ " http: // localhost: 8228 / CUSTOMER_SERVER1 " يخبر المعامل الثالث الحاوية بما يجب القيام به مع الكائن عندما يأتي طلب Request للكائن.

ينشئ استدعاء WellKnownObjectMode.Single غرضاً جديداً للكائن وذلك من أجل كل عميل أما WellKnownObjectMode.Singleton فإنه يستخدم غرضاً واحداً للكائن لجميع المتصلين.

من المنطقي استخدام خيار Singleton هنا حيث يتعين علينا الحفاظ على اتصال قاعدة بيانات فريد يمكن استخدامه عبر العملاء ومكالمات SQL.

صف الخادم هو من سيقوم بالفعل بتحقيق الطرق التي تم عرضها على الواجهةInterface_CustomerInfo.

يكون تعريف صف الخادم على الشكل التالي:

```
public class CUSTOMER_SERVER1 : MarshalByRefObject , ICustomerInfo
```

حيث أنه يرث من كائن MarshalByRefObject من أجل تمكين استدعاء الأغراض البعيدة وواجهة ICustomerInfo التي أنشأناها في واجهة DLL في الخطوة 1.

بالطبع نقوم بترجمة الكود كما فعلنا في الجزء الأول.

4. الجزء الثالث: تكوين العميل(CustomerClient):

كائن CustomerClient هو كائن اختبار للكائن البعيد CustomerServer المنشأ لدينا، حيث ننشئ مشروعاً جديداً بالنقر فوق ملف-> جديد-> مشروع وقمنا بإنشاء عميل windows بسيط يتصل بالكائن البعيد ، وينفذ أمر SQL ، ويعيد صفوفاً من قاعدة البيانات كسلسلة واحدة مفصولة بفاصلة.
هنا أيضاً نحن بحاجة إلى إضافة مرجع إلى Interface_CustomerInfo.DLL المشترك الذي تم إنشاؤه في الجزء الأول ، نضيف مرجعاً إلى System.Runtime.Remoting DLL باستخدام خيار المراجع> AddRefrences في مستكشف الحلول.
وما يلي جزء من الكود الخاص بالعميل:

```
ChannelServices.RegisterChannel(new TcpClientChannel());
```

```
custl = (ICustomerInfo) Activator.GetObject(typeof(ICustomerInfo), "
tcp://localhost:8228/CUSTOMER_SERVER2");

if (custl == null) {

    Console.WriteLine("IT SEEMS THAT OUR SERVER IS OFFLINE NOW,
...TRY AGAIN LATER, PLEASE");

    return;

}

custl.Init("DESKTOP-IIRMO93\\ABDULLAHSQL", "");
```

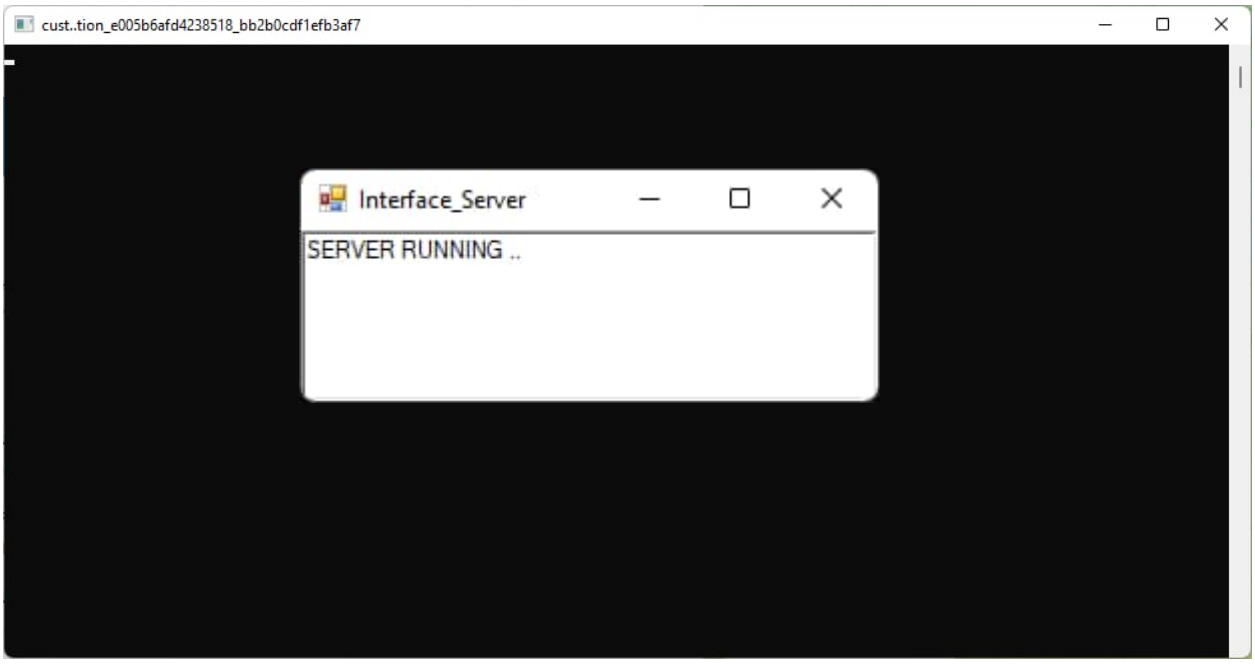
لاحظ الآن السطرين السابقين المهمين في البرنامج. السطر الأول ينشئ TcpClientChannel حيث أن هذه القناة ليست مرتبطة بمنفذ وفي السطر الثاني يحصل في الواقع على إشارة إلى كائن خادمنا CUSTOMER_SERVER2 ترجع طريقة Activator.GetObject نوعاً من الكائنات يمكننا بعد ذلك قصرها إلى CUSTOMER_SERVER2 حيث أن البارامترات التي نمررها تشبه إلى حد بعيد ما مررناه إلى كائن RemotingConfiguration في مشروع الخادم، البارامتر الاول هو نوع الكائن، والثاني هو URI للكائن البعيد. لاحظ أننا نستخدم كائن ICustomerInfo، يخدم الخادم الكائن CUSTOMER_SERVICE1 لكننا نستخدم كائن ICustomerInfo عندما تقوم باستدعاء طريقة التهيئة للواجهة ، يتم تنشيط الإصدار المطبق من الطريقة الذي يتعامل معه الخادم. هذه العملية شفافة تمامًا للعميل عند استخدام واجهة ICustomerInfo.

بالطبع نقوم بترجمة الكود كما فعلنا في الجزأين السابقين.

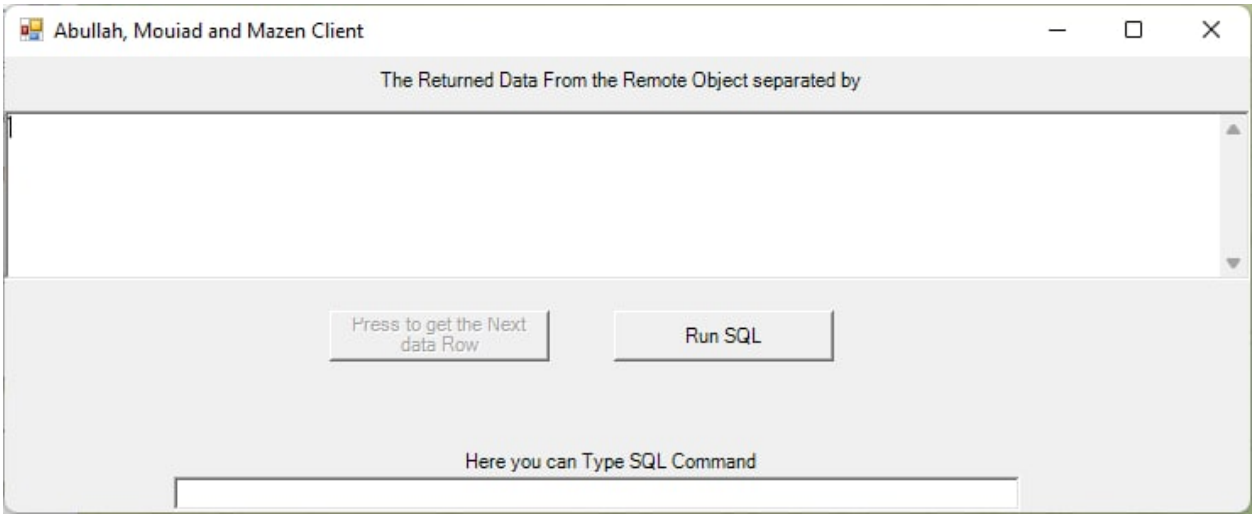
الآن وبعد أن اكلمنا كتابة وترجمة الأجزاء الثلاثة يمكننا التنفيذ حيث أننا نقوم بتشغيل السيرفر أولا وتهيئته لاستقبال الطلبات ثم العميل وفق الخطوات التالية.

5. التطبيق العملي (التنفيذ):

أولا نقوم بتشغيل المخدم (السيرفر) ونحصل على الواجهة التالية:

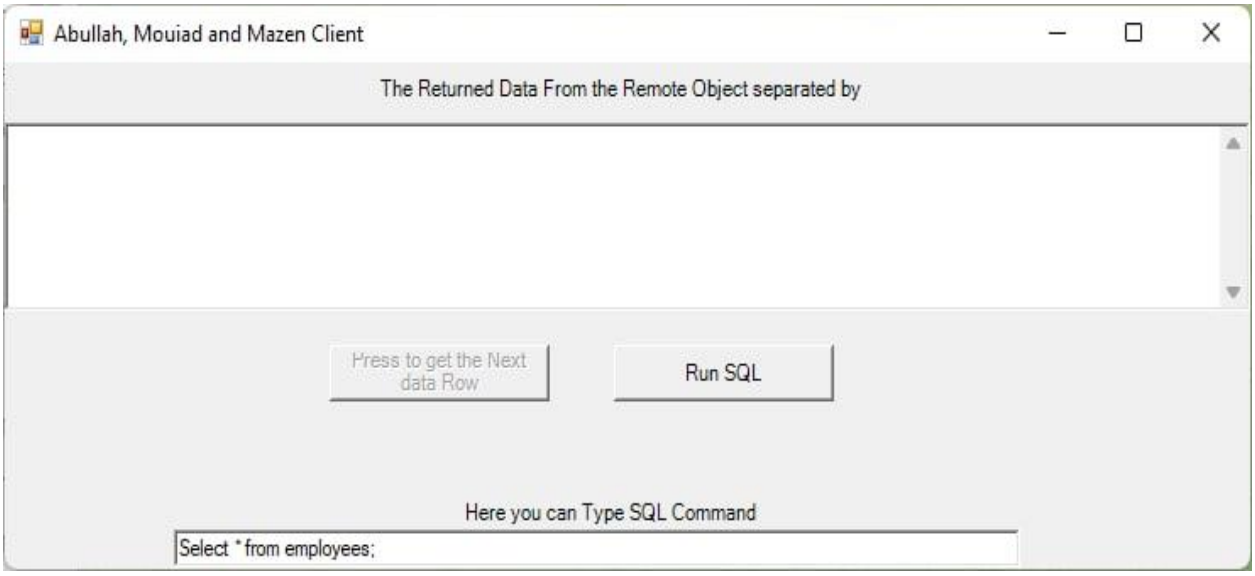


بعدها نقوم بتشغيل الزبون ونحصل على الواجهة التالية:

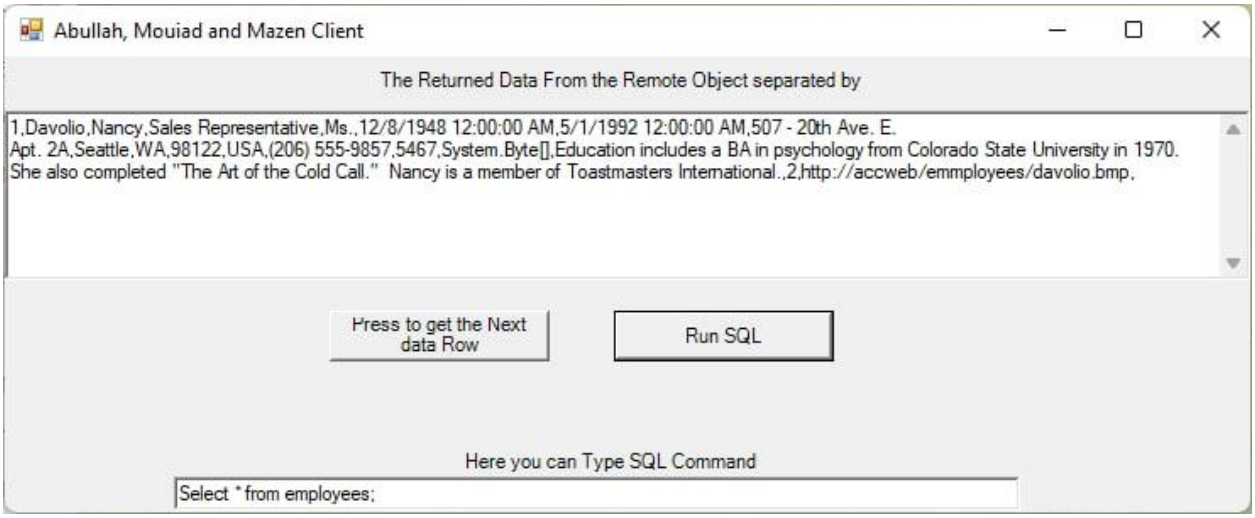


نقوم بتنفيذ استعلام على الداتا التي لدينا (northwind):

```
SELECT * FROM Employees;
```



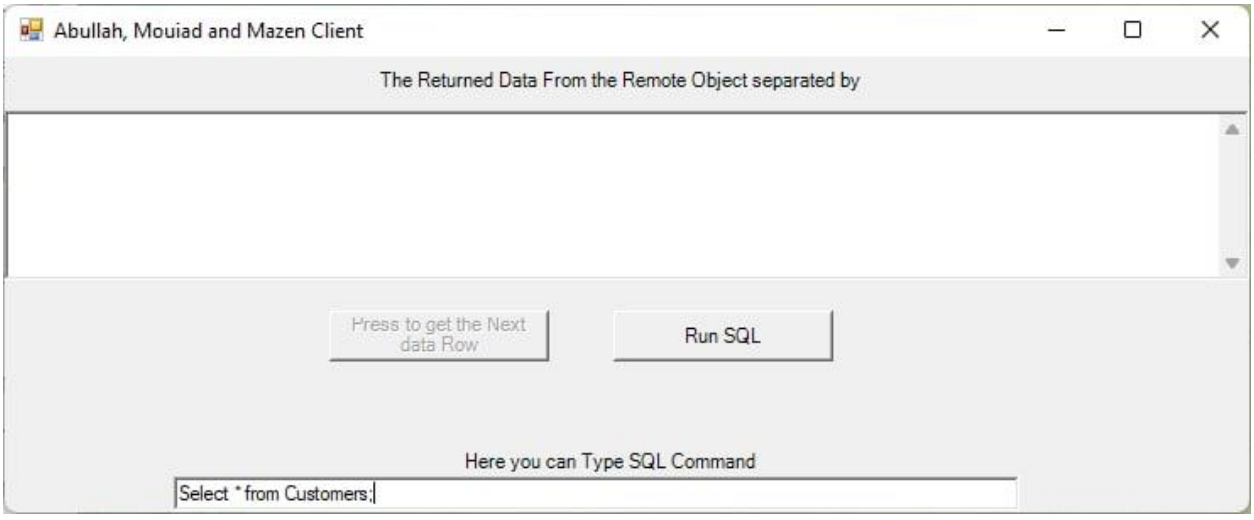
فيكون الخرج:



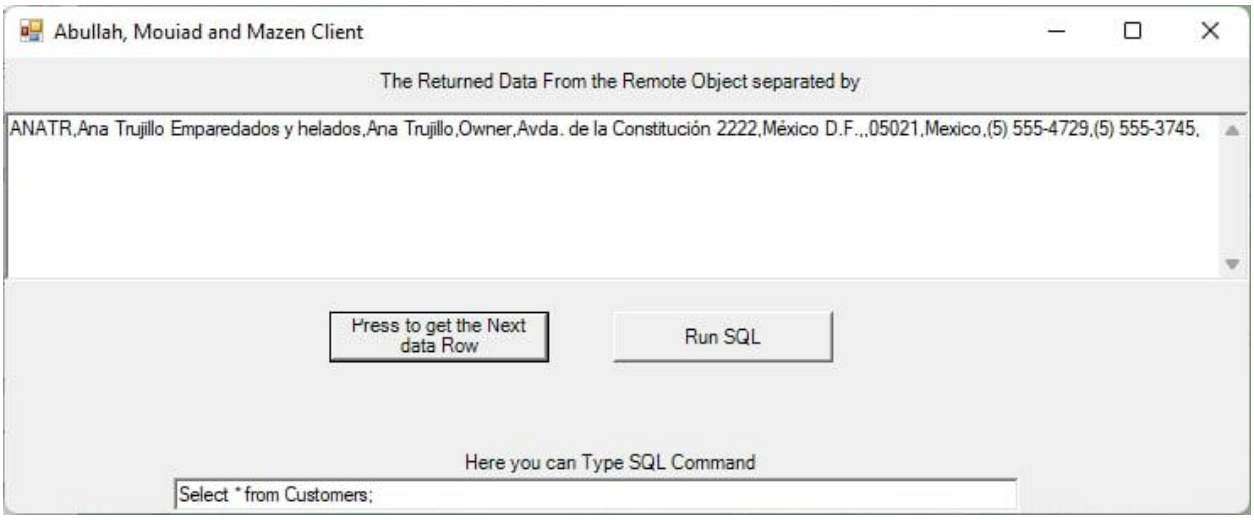
وهو المطلوب.

لننفذ استعلام أخر وليكن:

SELECT * FROM Customers;



فيكون الخرج:



وهو المطلوب.

6. المراجع

[1] .NET Remoting: The Interface Approach (c-sharpcorner.com)

[2] Sending DataSet Data across the Wire using .NET Remoting (akadia.com)

[3] Theoretical and practical lectures for .NET Remoting from 1 to 5 by Dr. Sira ASTOUR.