



الفصل الخامس عشر:

التعامل مع واجهات الوب Web API

الصفحة	العنوان
3	1. التعامل مع واجهات الوب Web API
4	التعامل مع واجهات الوب Web API

الكلمات المفتاحية:

واجهة وب، طلب Http من النوع Get، طلب Http من النوع Post، الصف HttpAsyncTask.

ملخص:

نستعرض في هذا الفصل أساسيات الاتصال مع واجهات الويب.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- إنشاء واجهة وب.
- التعامل مع الطلبات Get و Post.
- آلية طلب واجهة الويب من تطبيق الموبايل.
- استخدام الصف HttpAsyncTask.

المخطط:

التعامل مع واجهات الويب

- 1 وحدة (Learning Objects)

1. التعامل مع واجهات الوب Web API

الأهداف التعليمية:

- الاتصال مع واجهات الوب

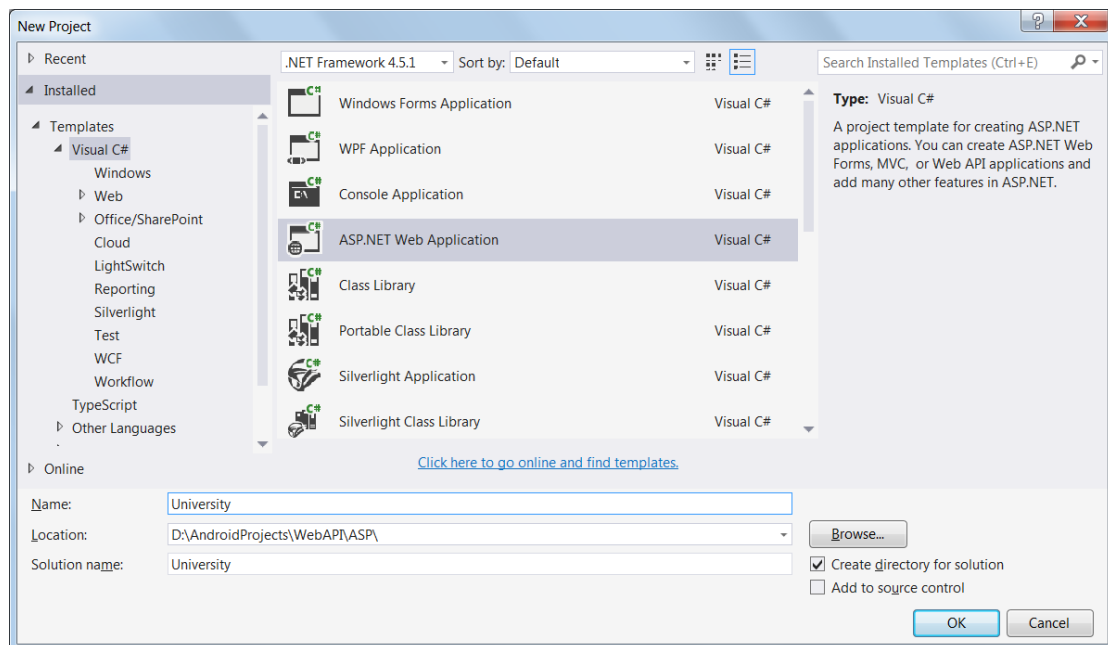
التعامل مع واجهات الوب Web API

- تحتاج تطبيقات الموبايل في الكثير من الحالات إلى الاتصال والتعامل مع واجهات الوب ولاسيما للتخاطب مع قواعد البيانات.
- سنقوم عبر مثال تعليمي بالتعرف على جميع المفاهيم والخطوات الأساسية من أجل تحقيق اتصال تطبيق موبايل مع واجهة وب.

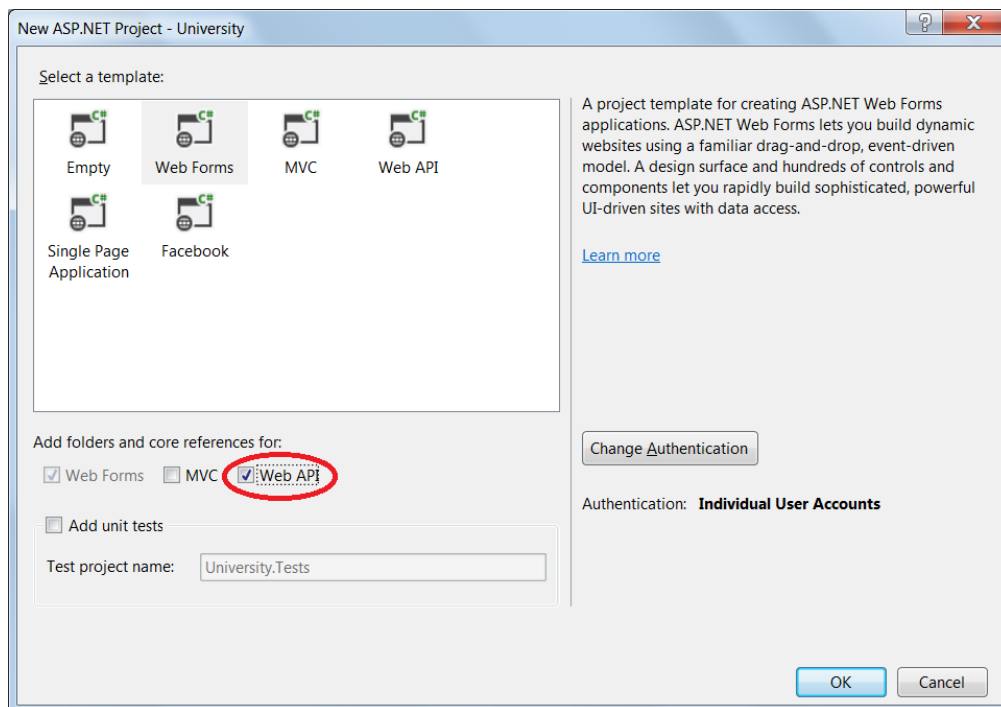
إنشاء تطبيق الوب باستخدام Visual Studio 2013

- سنقوم في مثالنا التعليمي بإنشاء تطبيق وب يحوي قاعدة بيانات طلاب جامعة. سيتمكن طلاب الجامعة في النهاية باستخدام الموبايل لمعرفة علاماتهم في المواد المسجلين فيها.
- نُبين فيما يلي الخطوات لإنشاء تطبيق الوب الأساسي:

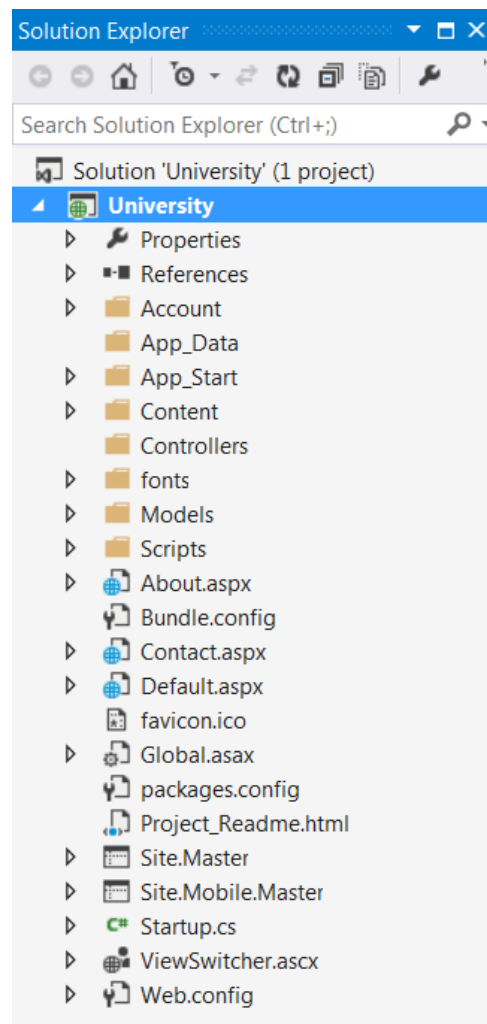
1. أنشئ أولاً تطبيق وب ASP.NET Web Application باستخدام محيط العمل Visual Studio 2013:



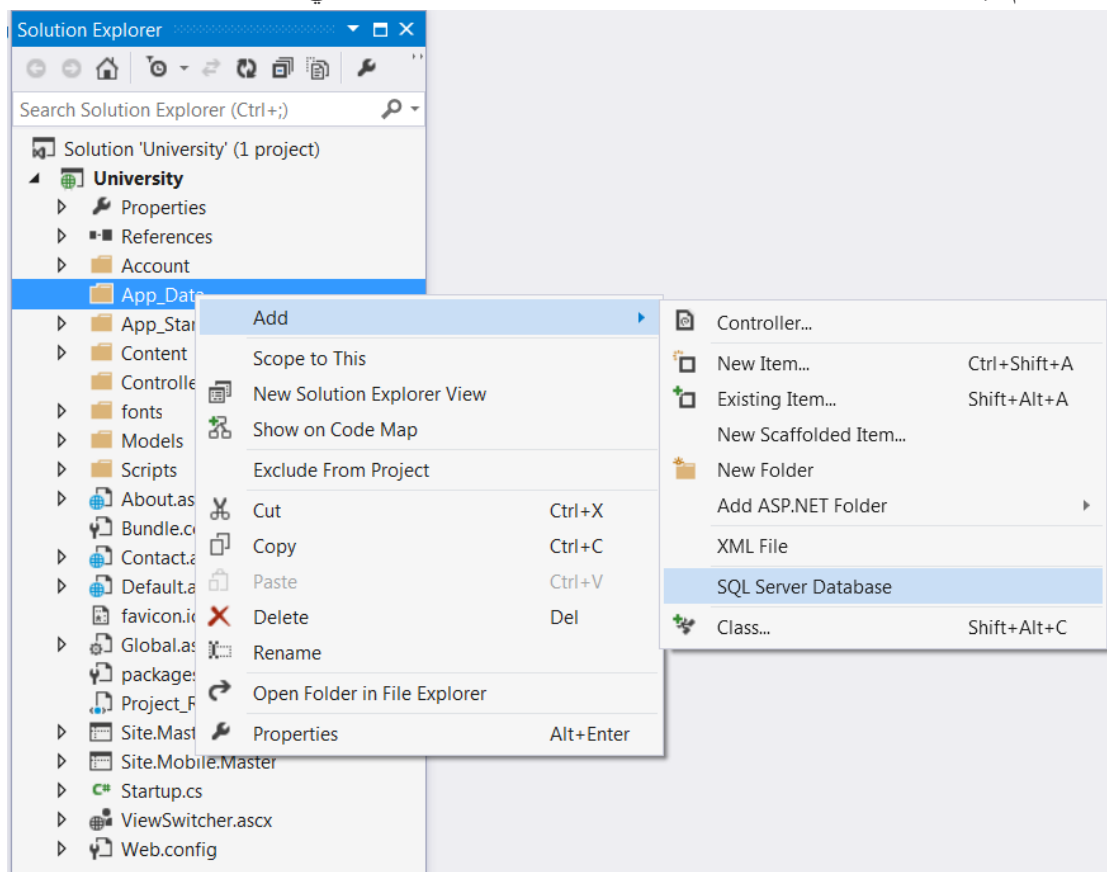
2. حدّد الخيار Web API مما يجعل محيط العمل يقوم بتوليد مجموعة من الملفات اللازمة لتحقيق واجهة الوب بسهولة.



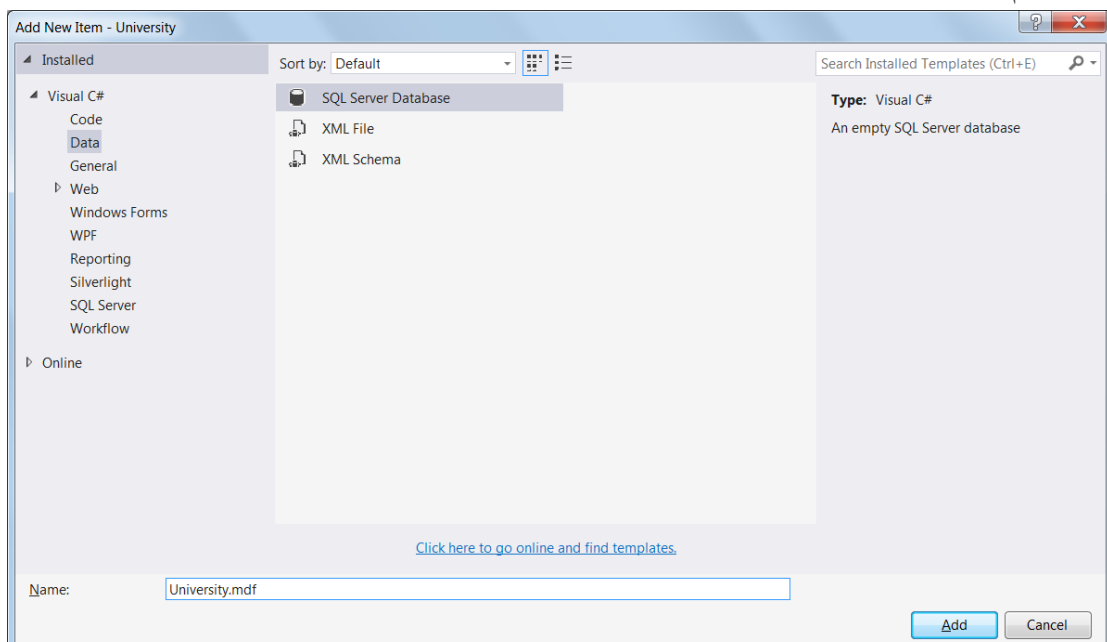
3. تأكد بعد فتح مستعرض الملفات Solution Explorer من البنية التالية:



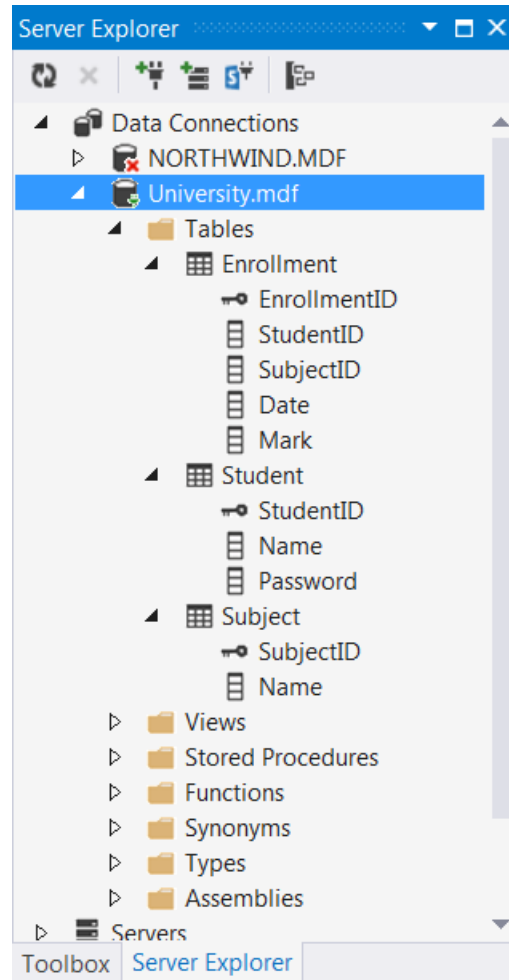
4. قم بإضافة قاعدة بيانات من النمط SQL Server Database في المجلد App_Data.



5. قم بتسمية الملف :University.mdf



6. قم بإنشاء الجداول الثلاثة التالية:

**جدول الطلاب Student:**

- رقم الطالب StudentID
- الاسم Name
- كلمة السر Password

جدول المواد Subject:

- رقم المادة SubjectID
- المادة Subject

جدول التسجيل Enrollment:

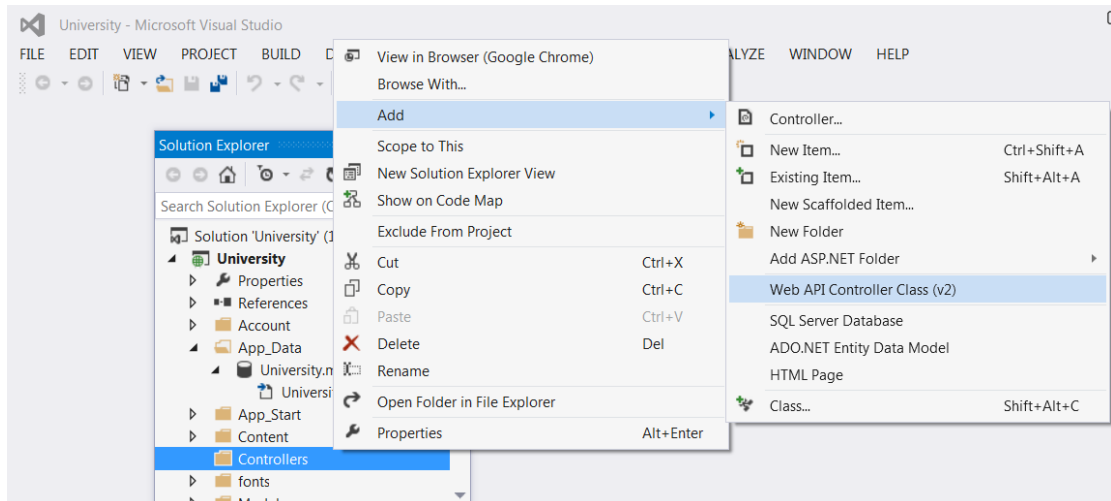
- رقم التسجيل EnrollmentID
- رقم الطالب StudentID
- رقم المادة SubjectID
- التاريخ Date
- العلامة Mark

7. أنشئ صفحات الويب اللازمة للتعامل مع الجداول وإدخال البيانات فيها بالشكل الذي تريده.

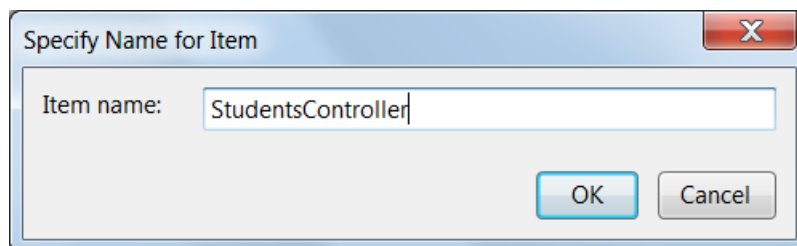
إنشاء واجهة وب Web API لجدول الطلاب

- سنقوم فيما يلي بإنشاء واجهة الويب اللازمة للتعامل مع جدول الطلاب:

1. قم في المجلد Controllers بإضافة عنصر جديد من النوع Web API Controller Class:



2. قم بتسمية الملف StudentsController:



3. عاين الملف المنشئ StudentController.cs:

لاحظ احتواء الملف بشكل أساسي على الإجراءات اللازمة لمعالجة طلبات http من النمط Get أو .Post

```
namespace University.Controllers
{
    public class StudentsController : ApiController
    {
        // GET api/<controller>
        public IEnumerable<string> Get()
        {
            return new string[] { "value1", "value2" };
        }
    }
}
```

```

// GET api/<controller>/5
public string Get(int id)
{
    return "value";
}

// POST api/<controller>
public void Post([FromBody]string value)
{
}

// PUT api/<controller>/5
public void Put(int id, [FromBody]string value)
{
}

// DELETE api/<controller>/5
public void Delete(int id)
{
}
}
}

```

4. قم باستدعاء واجهة الوب السابقة عن طريق كتابة العنوان:

//localhost:PortNumber/api/Students



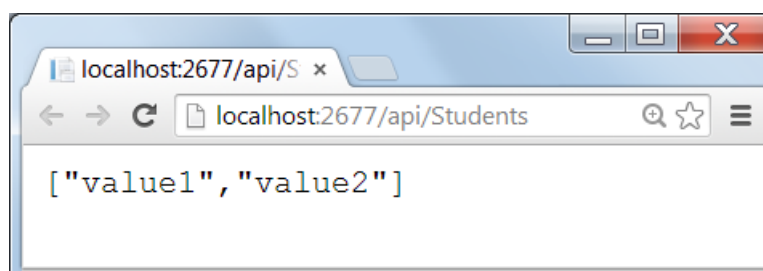
يكون الاستدعاء بشكل افتراضي من النمط Get ويتم إعادة النتائج بتنسيق XML.
5. نحتاج في مثالنا للتخاطب وفق التنسيق القياسي JSON. لجعل النتائج تُعاد وفق تنسيق JSON، قم بفتح الملف Global.asax وأضف السطر التالي لطلب إلغاء التنسيق XML:

```
GlobalConfiguration.Configuration.Formatters.XmlFormatter.SupportedMediaTypes.Clear();
```

لُيُصبح الملف Global.asax:

```
public class Global : HttpApplication
{
    void Application_Start(object sender, EventArgs e)
    {
        // Code that runs on application startup
        GlobalConfiguration.Configure(WebApiConfig.Register);
        RouteConfig.RegisterRoutes(RouteTable.Routes);
        BundleConfig.RegisterBundles(BundleTable.Bundles);
        // NO XML
        GlobalConfiguration.Configuration.Formatters.XmlFormatter.SupportedMediaTypes.Clear();
    }
}
```

6. قم بإعادة استدعاء واجهة الوب السابقة ولاحظ النتيجة الآن وفق التنسيق JSON:

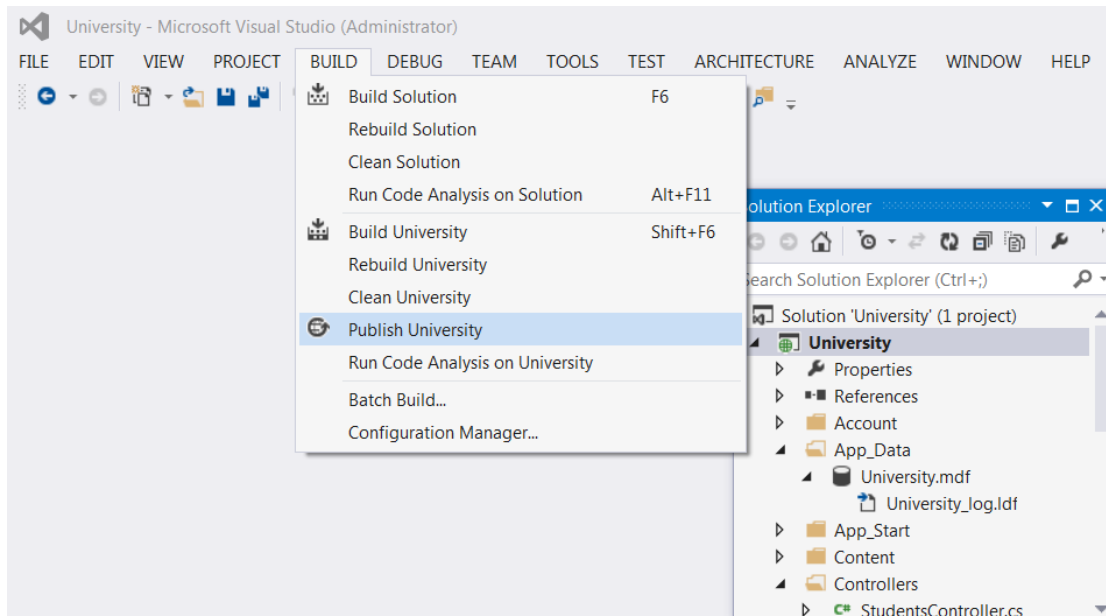


استضافة واجهة الويب

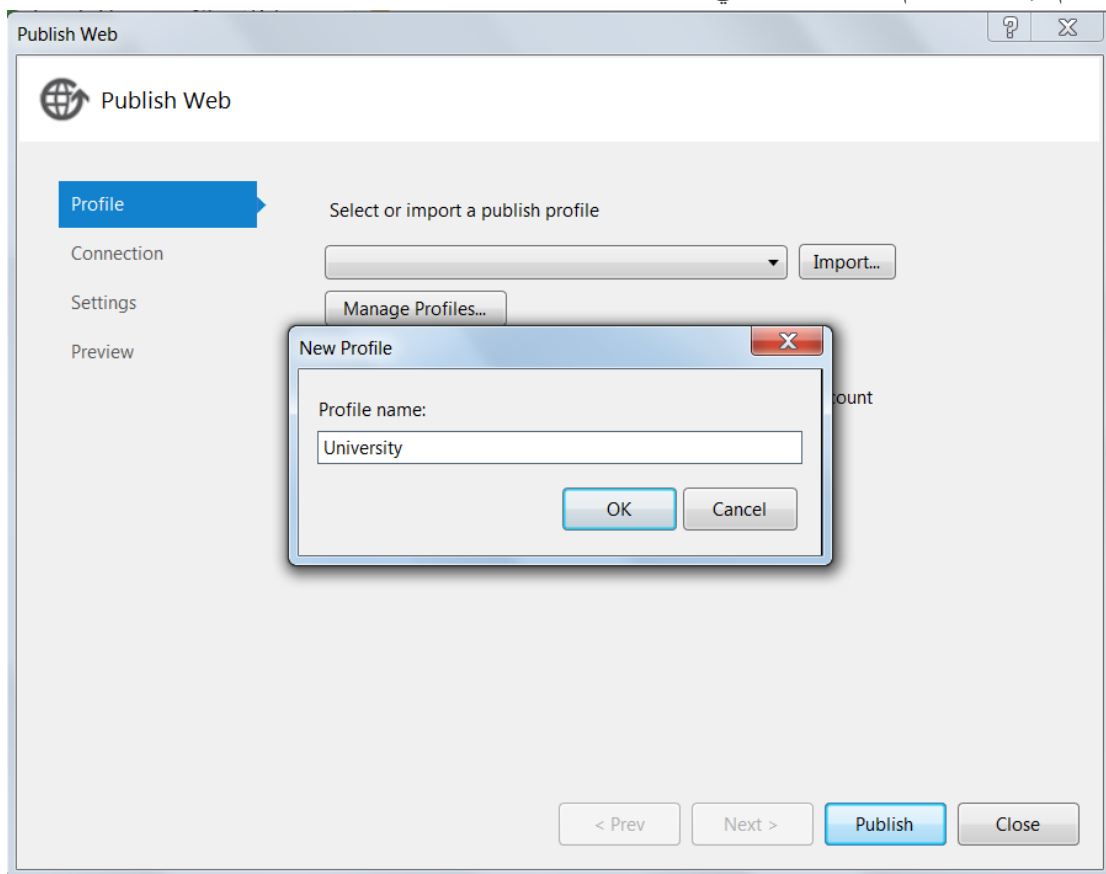
- لنقم الآن باستضافة الموقع محلياً على IIS:

1. افتح Visual Studio 2013 بشكل مدير Run as administrator

2. اختر من القائمة BUILD الخيار Publish University:



3. قم بإعطاء الاسم University في الحقل Profile name:



4. حدّد آليات الاتصال وفق ما يلي:

The screenshot shows the 'Publish Web' dialog box with the 'Connection' tab selected. The 'University *' section is visible. The 'Publish method' is set to 'File System'. The 'Target location' is 'http://localhost/'. The 'Publish' button is highlighted.

Publish Web

Profile

Connection

Settings

Preview

University *

Publish method: File System

Target location: http://localhost/

< Prev Next > Publish Close

5. حدّد الإعدادات كما يلي:

The screenshot shows the 'Publish Web' dialog box with the 'Settings' tab selected. The 'University *' section is visible. The 'Configuration' is set to 'Release'. The 'File Publish Options' section is expanded, showing 'Delete all existing files prior to publish' checked, 'Precompile during publishing' unchecked, and 'Exclude files from the App_Data folder' unchecked. The 'Databases' section is empty. The 'Publish' button is highlighted.

Publish Web

Profile

Connection

Settings

Preview

University *

Configuration: Release

File Publish Options

☒ Delete all existing files prior to publish

☐ Precompile during publishing [Configure](#)

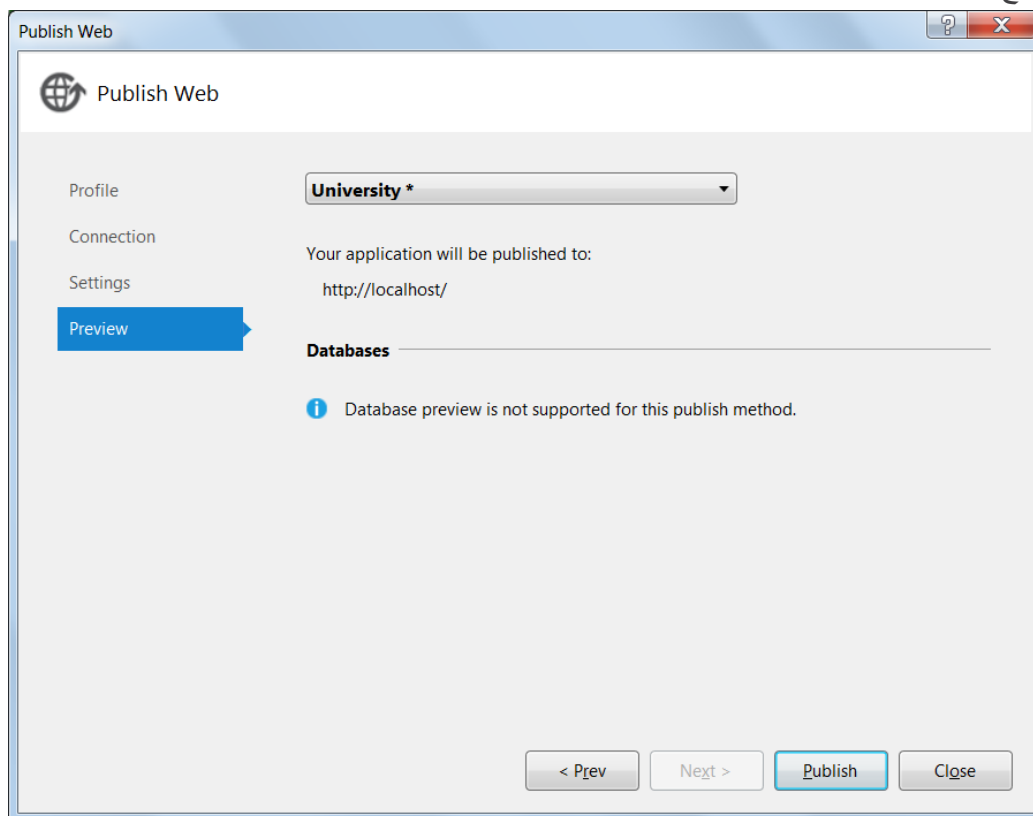
☐ Exclude files from the App_Data folder

Databases

i Database publishing is not supported for this publish method.

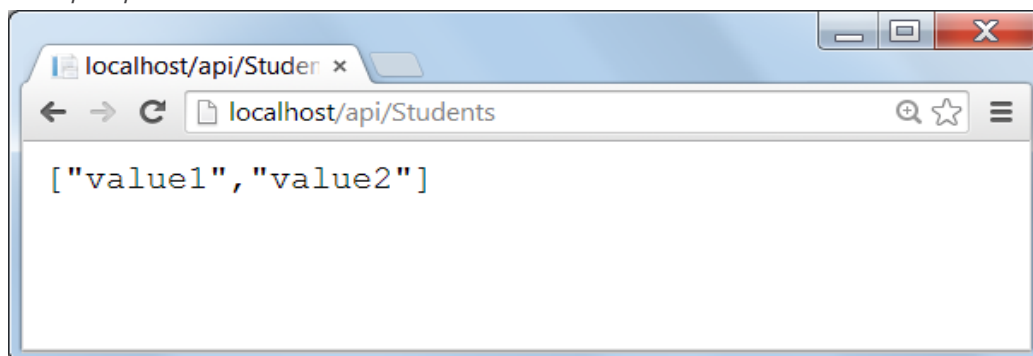
< Prev Next > Publish Close

6. تابع الخطوة الأخيرة:



7. تأكد من الاستضافة بشكل ناجح عن طريق طلب واجهة الوب بكتابة العنوان:

//localhost/api/Students



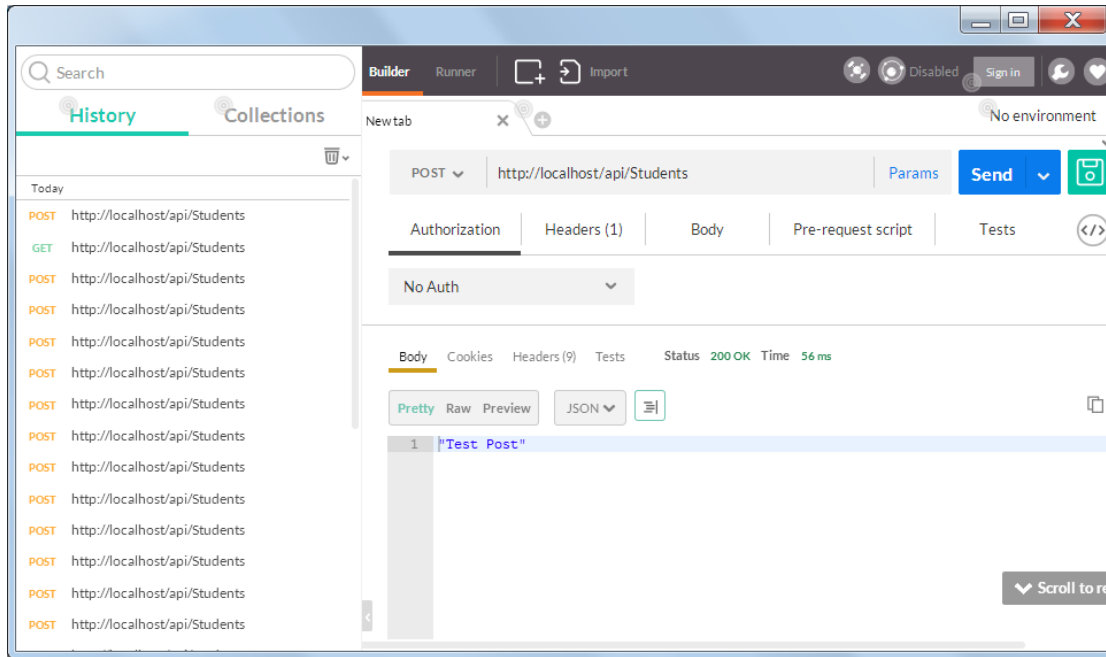
- اختبار استدعاء واجهة الوب بالطريقة Post

1. قم بإضافة الأداة Postman إلى Google Chrome

2. عدّل الإجراءية Post لتُعيد سلسلة نصية للاختبار:

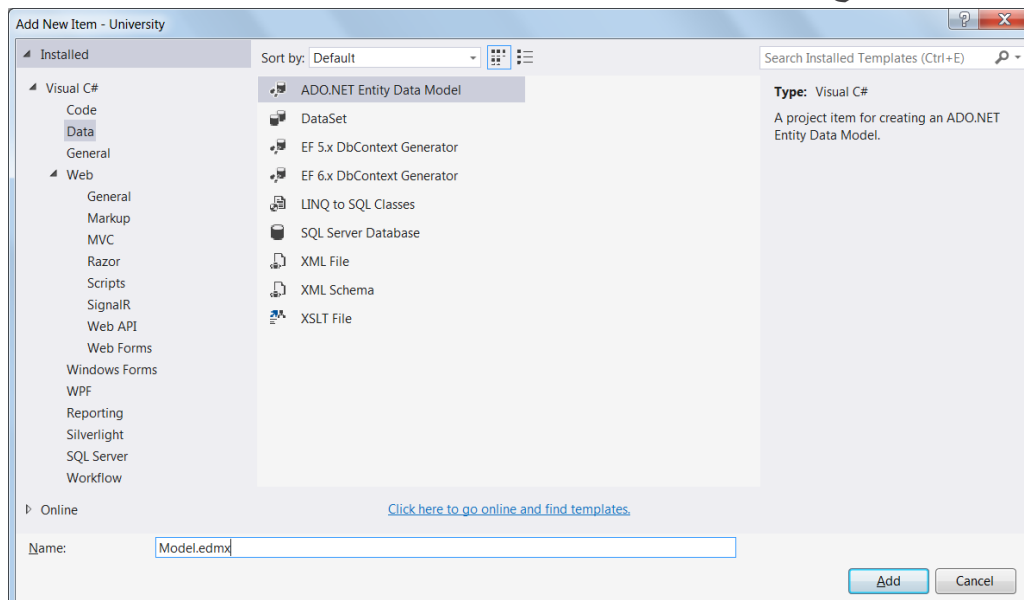
```
public string Post([FromBody]string value)
{
    return "Test Post";
}
```

3. قم بطلب واجهة الويب مع الخيار Post:

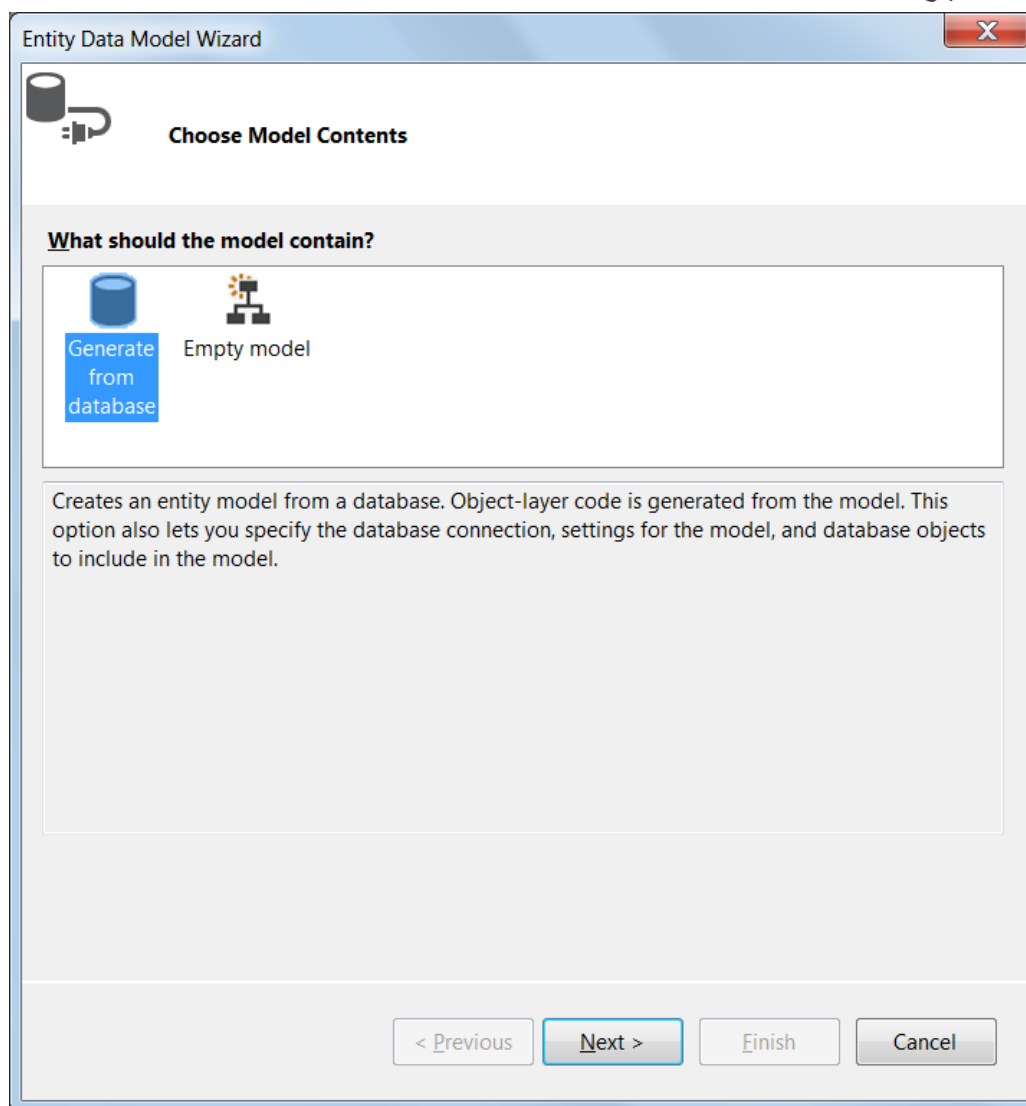


التعامل مع قاعدة البيانات

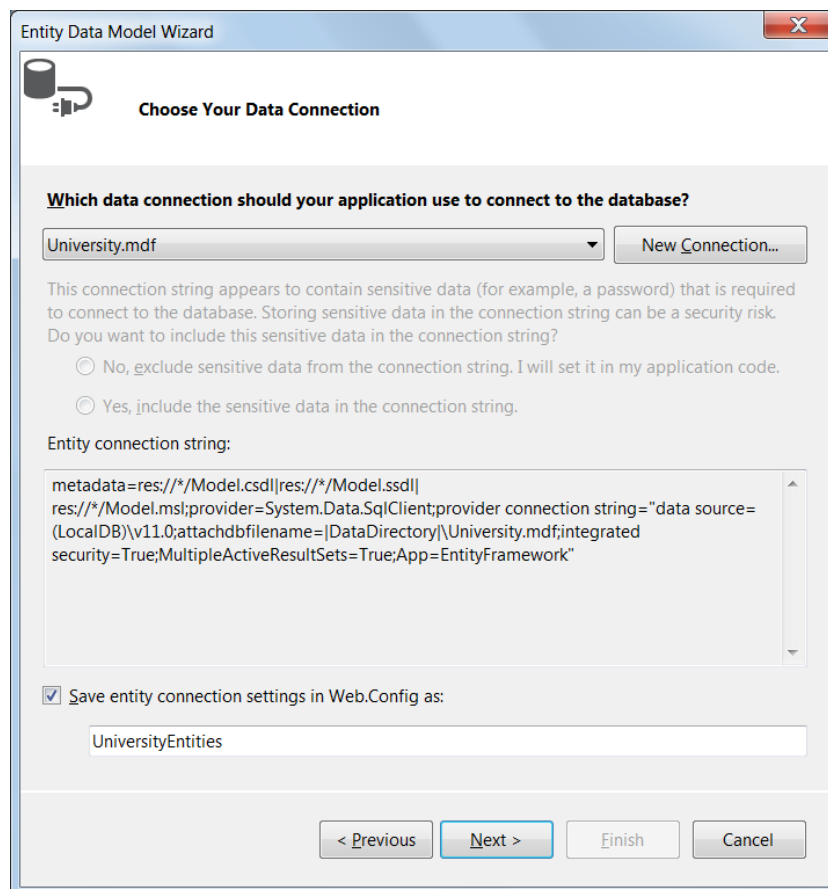
1. أضيف إلى المشروع عنصر جديد من النمط ADO.NET Entity Data Model:



2. حدد الخيار :Generate from database



3. اختر قاعدة البيانات :University.mdf



The image shows the 'Entity Data Model Wizard' dialog box. It has a title bar with the text 'Entity Data Model Wizard' and a close button. The main area is titled 'Choose Your Data Connection'. It contains a dropdown menu with 'University.mdf' selected and a 'New Connection...' button. Below this, there is a warning message about sensitive data in the connection string and two radio buttons: 'No, exclude sensitive data from the connection string. I will set it in my application code.' (selected) and 'Yes, include the sensitive data in the connection string.'. A text box labeled 'Entity connection string:' contains the following text: 'metadata=res://*/Model.csdl|res://*/Model.ssdl|res://*/Model.msl;provider=System.Data.SqlClient;provider connection string="data source=(LocalDB)\v11.0;attachdbfilename=|DataDirectory|\University.mdf;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework"'. At the bottom, there is a checkbox 'Save entity connection settings in Web.Config as:' which is checked, and a text box containing 'UniversityEntities'. Navigation buttons at the bottom are '< Previous', 'Next >' (highlighted), 'Finish', and 'Cancel'.

Entity Data Model Wizard

Choose Your Data Connection

Which data connection should your application use to connect to the database?

University.mdf New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string. I will set it in my application code.

☐ Yes, include the sensitive data in the connection string.

Entity connection string:

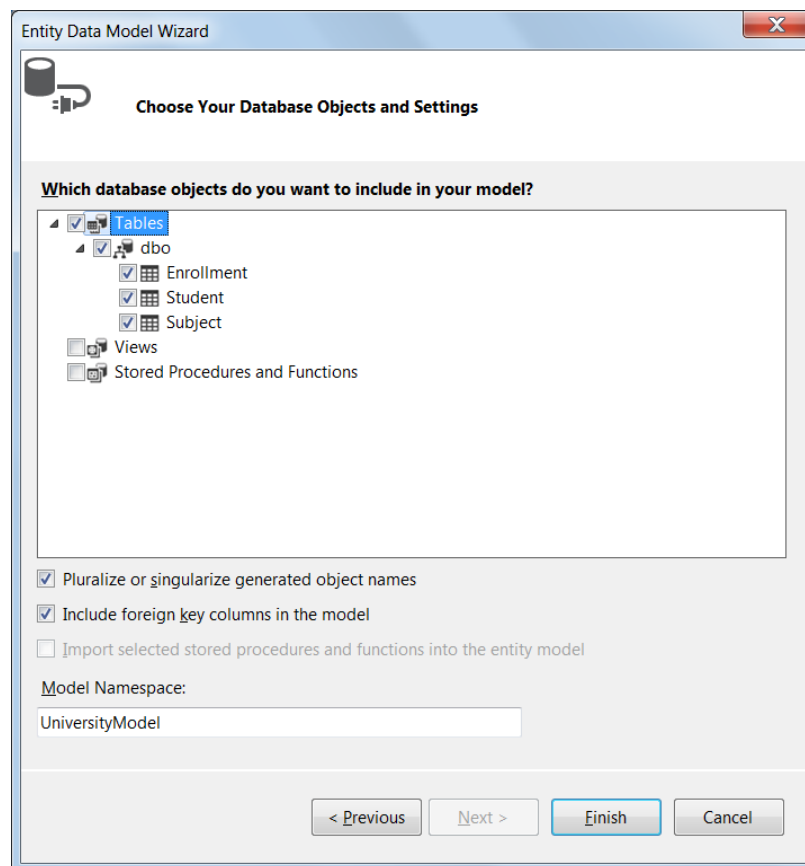
```
metadata=res://*/Model.csdl|res://*/Model.ssdl|
res://*/Model.msl;provider=System.Data.SqlClient;provider connection string="data source=
(LocalDB)\v11.0;attachdbfilename=|DataDirectory|\University.mdf;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
```

☒ Save entity connection settings in Web.Config as:

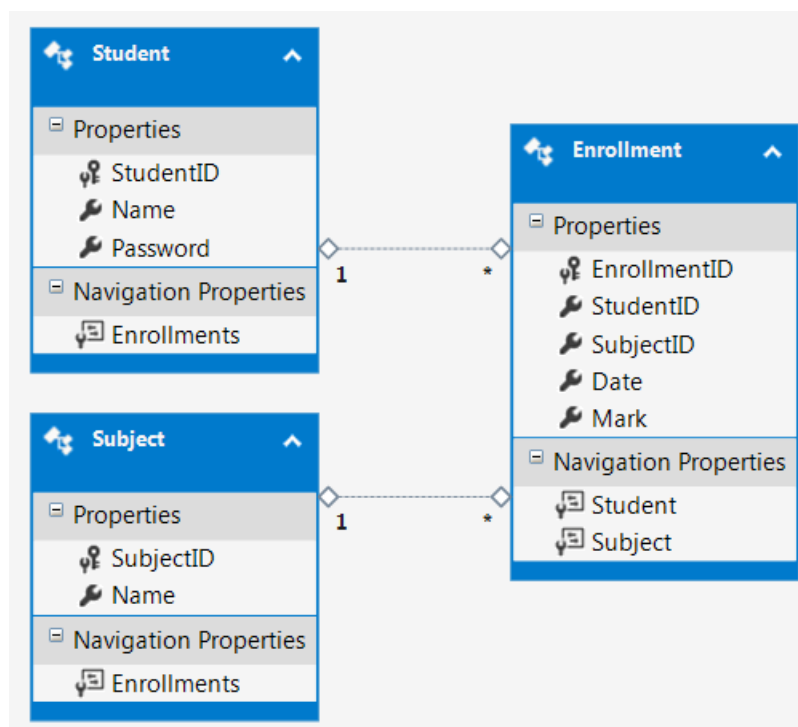
UniversityEntities

< Previous Next > Finish Cancel

4. اختر جميع جداول قاعدة البيانات:



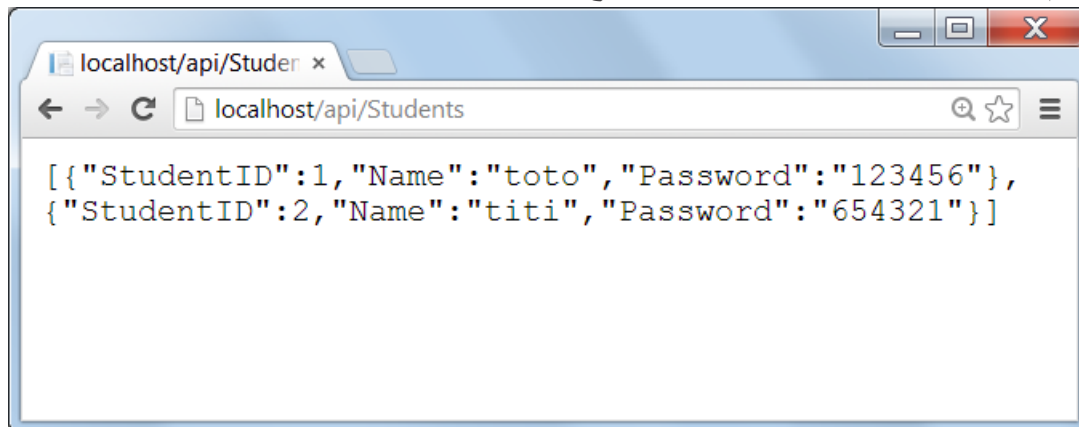
5. عاين الجداول:



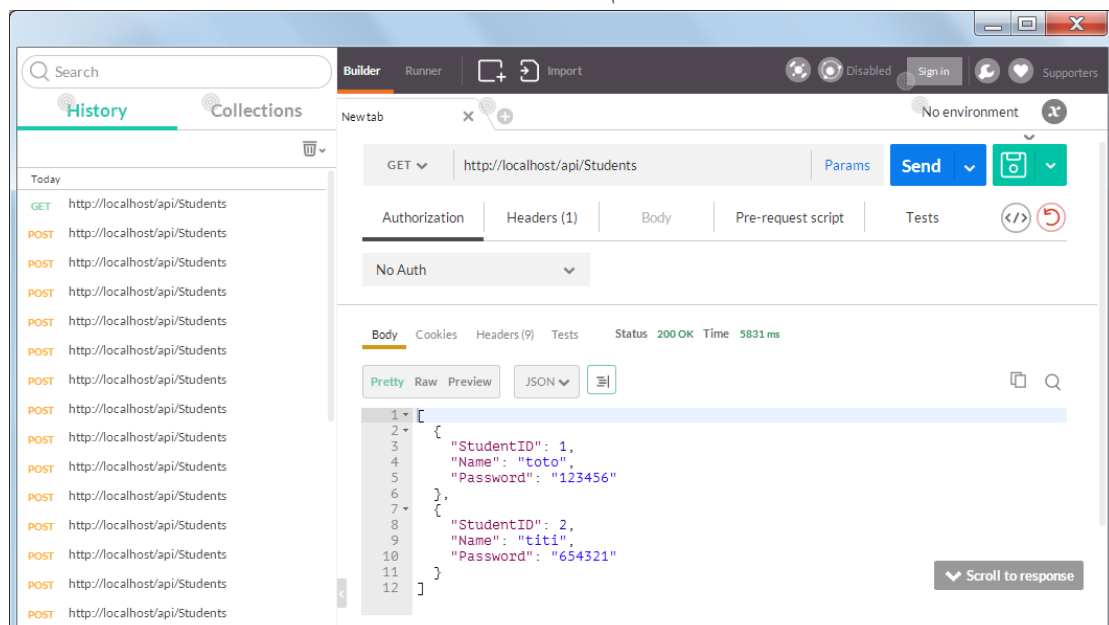
6. قم بهدف الاختبار بتعديل الإجرائية Get في الملف StudentController.cs لتُعيد بيانات جميع الطلاب:

```
public IEnumerable<Student> Get()
{
    UniversityEntities db = new UniversityEntities();
    return db.Students.ToList();
}
```

7. قم باستدعاء واجهة الوب ولاحظ ظهور جميع بيانات جدول الطلاب:



8. يُمكنك أيضاً المعاينة من خلال استخدام الامتداد Postman:



التحقق من بيانات المستخدم

- تُرسل بيانات المستخدم عادةً (الاسم وكلمة السر) باستخدام Post كي لا تكون ظاهرة في شريط العنوان.
- سنقوم فيما يلي بكتابة ما يلزم في خدمة الوب للتحقق من البيانات المرسلة بشكل Post في طلب :HTTP

1. نُصرّح أولاً عن الصفين:

```
public class Credentials
{
    public string UserName { set; get; }
    public string Password { set; get; }
}

public class LoginResult
{
    public string Result { set; get; }
}
```

يُستخدم الصف الأول لإنشاء أغراض لبيانات المستخدم. أما الصف الثاني فلنتيجة الاستعلام عن بيانات المستخدم (إما مُعرّف المستخدم أو -1).

2. نقوم بتعديل الإجراءية Post والتي تقوم بالاستعلام عن مطابقة الاسم وكلمة السر. في حال التأكد فإنها تُعيد مُعرّف الطالب وإلا فإنها تُعيد -1:

```
public LoginResult Post([FromBody]Credentials value)
{
    UniversityEntities db = new UniversityEntities();

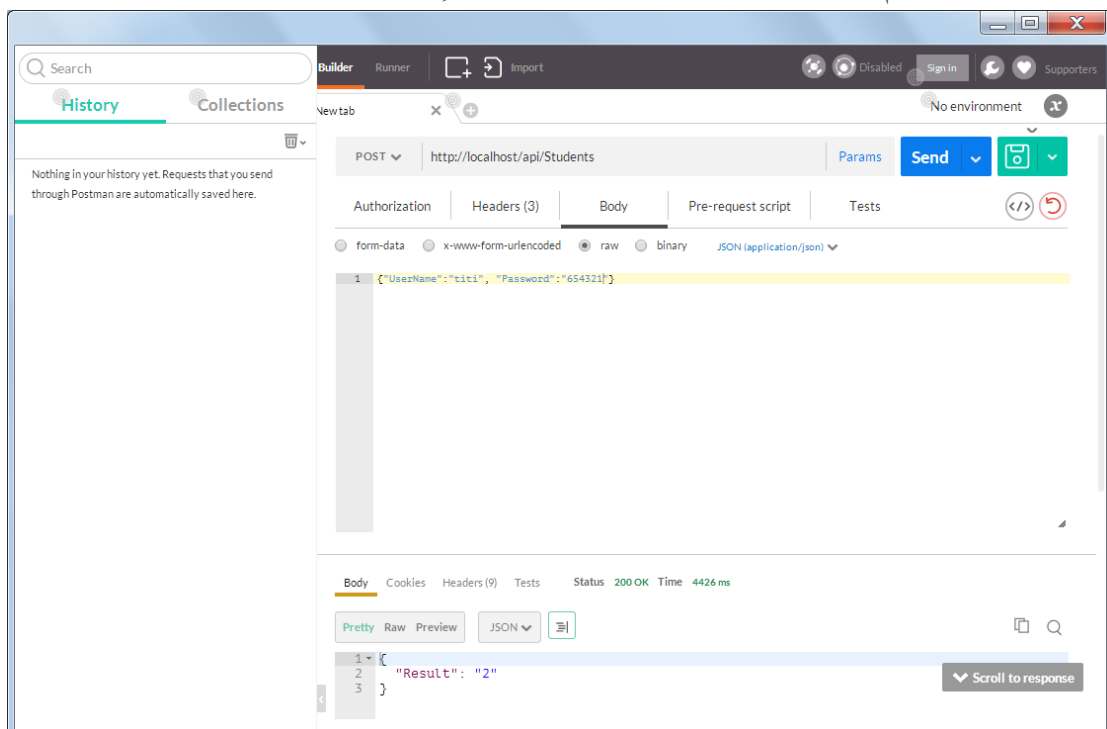
    var query = from record in db.Students
                where record.Name == value.UserName
                  && record.Password == value.Password
                select record;

    Student std = query.FirstOrDefault();

    if (std != null)
        return new LoginResult()
            { Result = std.StudentID.ToString() };

    return new LoginResult() { Result = "-1" };
}
```

3. يُمكن استخدام الامتداد Postman للتحقق من عمل الإجرائية Post:



إنشاء واجهة الويب للاستعلام عن نتائج طالب

- سنقوم فيما يلي بإنشاء واجهة الويب اللازمة للاستعلام عن نتائج طالب:

1. قم في المجلد Controllers بإضافة عنصر جديد من النوع Web API Controller Class وسمّه EnrollmentController.

2. قم بالتصريح عن الصف التالي والذي يحوي جميع البيانات اللازمة:

```
public class EnrollmentVM
{
    public int EnrollmentID { set; get; }
    public int SubjectID { set; get; }
    public int StudentID { set; get; }
    public DateTime Date { set; get; }
    public double Mark { set; get; }

    // the following properties are used to get some properties from the
    // navigation properties of Enrollment
    public string SubjectName { set; get; }
    public string StudentName { set; get; }
}
```

3. قم بتعديل الإجرائية Get بحيث تستقبل مُعرّف الطالب وتُعيد غرض من الصف السابق يحوي بيانات الطالب:

```
public IEnumerable<EnrollmentVM> Get(int id)
{
    UniversityEntities db = new UniversityEntities();

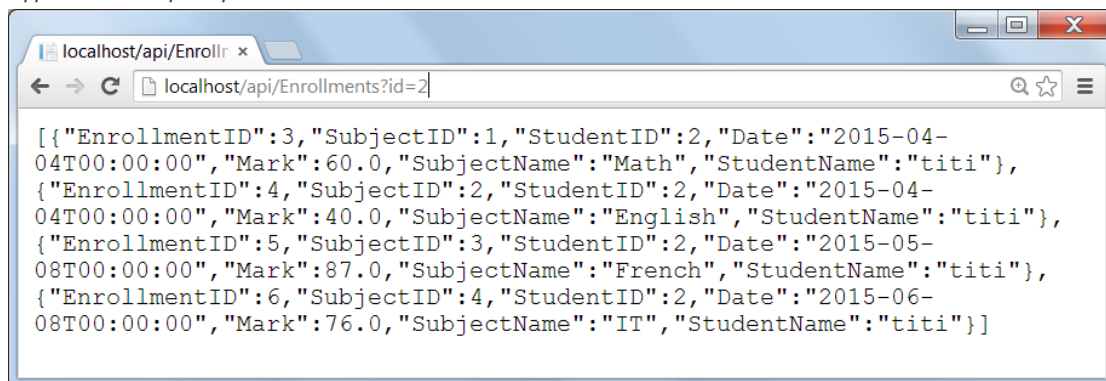
    var query = from record in db.Enrollments
                where record.StudentID == id
                select new EnrollmentVM()
                {
                    EnrollmentID = record.EnrollmentID,
                    StudentID = record.StudentID,
                    SubjectID = record.SubjectID,
                    Date = record.Date.Value,
                    Mark = record.Mark.HasValue ?
                        record.Mark.Value : 0,
                    StudentName = record.Student.Name,
                    SubjectName = record.Subject.Name
                } ;

    return query; }

```

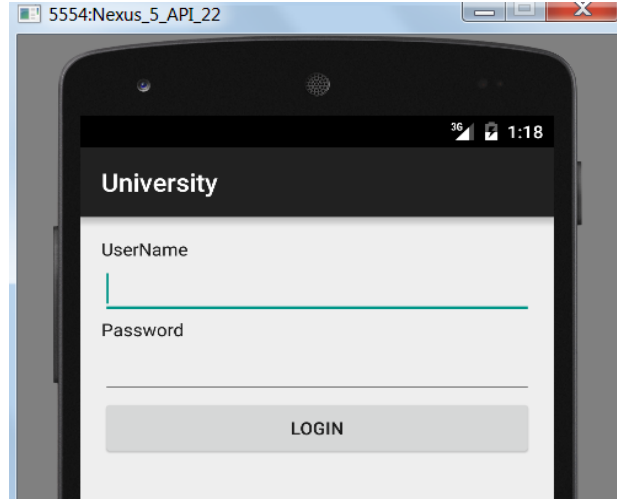
4. يُمكن اختبار الإجرائية السابقة بطلب العنوان:

<http://localhost/api/Enrollments?id=2>

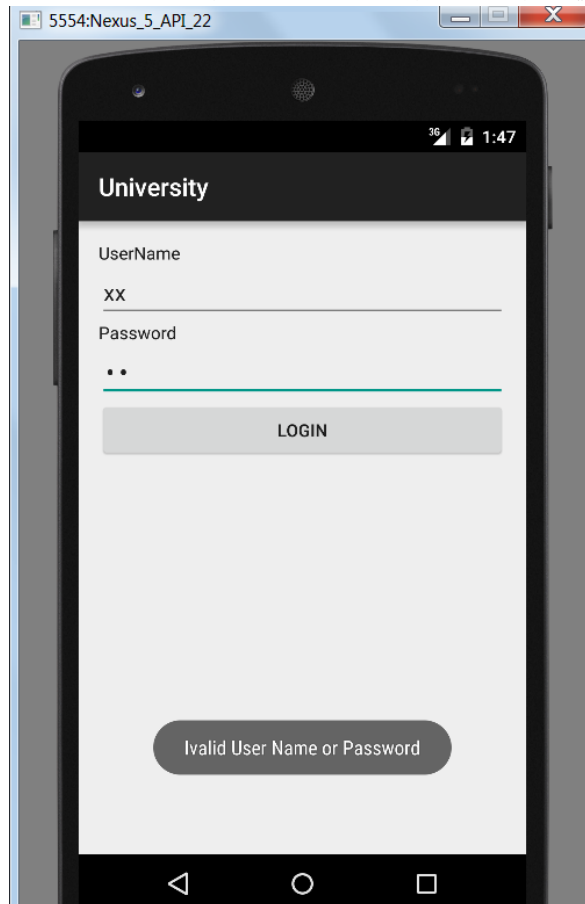


إنشاء واجهة أندرويد لإدخال اسم المستخدم وكلمة السر

- كي يتمكن الطالب من معاينة نتائجه، يجب عليه أولاً إدخال اسم المستخدم وكلمة السر عن طريق الواجهة التالية:



- عند وجود أي خطأ في اسم المستخدم أو كلمة السر، يتم إظهار رسالة خطأ موافقة:



- أما عند إدخال اسم مستخدم وكلمة سر صحيحين، فيتم الانتقال إلى الواجهة الثانية مع إظهار رسالة ترحيبية تحوي معرف الطالب.

(سنقوم لاحقاً بتعديل هذه الواجهة لتظهر نتائج الطالب).



- يكون ملف النشاط activity_main.xml للمواجهة السابقة:

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"

android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

<TextView

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"
```

```
    android:text="UserName"

    android:id="@+id/usernameTextView"

    android:layout_alignParentTop="true"

    android:layout_alignParentLeft="true"

    android:layout_alignParentStart="true"

    android:layout_alignParentRight="true"

    android:layout_alignParentEnd="true" />
```

<EditText

```
    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:id="@+id/usernameEditText"

    android:layout_below="@+id/usernameTextView"

    android:layout_alignParentLeft="true"

    android:layout_alignParentStart="true"

    android:layout_alignParentRight="true"

    android:layout_alignParentEnd="true" />
```

<TextView

```
    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:text="Password"

    android:id="@+id/passwordTextView"

    android:layout_below="@+id/usernameEditText"
```

```
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />
```

<EditText

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:ems="10"
    android:id="@+id/passwordEditText"
    android:layout_below="@+id/passwordTextView"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />
```

<Button

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Login"
    android:id="@+id/loginButton"
    android:layout_below="@+id/passwordEditText"
    android:layout_alignParentLeft="true"
```

```
        android:layout_alignParentStart="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:nestedScrollingEnabled="false"
        android:onClick="onClick" />
```

```
</RelativeLayout>
```

ويكون ملف الكود الموافق MainActivity.java

```
package com.example.basel.webapi;

import android.content.Intent;
import android.os.AsyncTask;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
```

```
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONObject;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

public class MainActivity extends ActionBarActivity {

    EditText usernameEditText ;

    EditText passwordEditText ;

    Button loginButton;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        usernameEditText =
            (EditText)findViewById(R.id.usernameEditText);

        passwordEditText =
```

```
(EditText)findViewById(R.id.passwordEditText);

loginButton = (Button)findViewById(R.id.LoginButton);}

public void onClick(View v) {

    // Here, we should call our Login post Web API

    // to call the Login method, we should run the call in an Asynchronous
    // task as Android prevents any web communication on the main thread

    new AsyncTask().execute("http://10.0.2.2/api/Students",
        usernameEditText.getText().toString(),
        passwordEditText.getText().toString());}

private String login
    (String url, String username, String password) {

    String result = "";

    try {

        // Create http request (POST Request)

        HttpClient httpClient = new DefaultHttpClient();

        HttpPost httpPost = new HttpPost(url);

        // Parameters that should be passed to server within the POST Request

        List<NameValuePair> nameValuePairs = new
            ArrayList<NameValuePair>(2);

        nameValuePairs.add(new
            BasicNameValuePair("UserName", username));

        nameValuePairs.add(new
            BasicNameValuePair("Password", password));

        httpPost.setEntity(new UrlEncodedFormEntity(nameValuePairs));

        // Execute the request and get the response
```

```
HttpResponse httpResponse = httpClient.execute(httpPost);

// Reading response using Java stream readers
// Convert the stream into String

InputStream inputStream =
    httpResponse.getEntity().getContent();

BufferedReader bufferedReader = new BufferedReader
    (new InputStreamReader(inputStream));

String line = "";

while ((line = bufferedReader.readLine()) != null) {

    result += line;

}

inputStream.close();

} catch (ClientProtocolException e) {

Toast.makeText(MainActivity.this, "Connection Error",
Toast.LENGTH_LONG).show();

} catch (IOException e) {

Toast.makeText(MainActivity.this, "Connection Error",
Toast.LENGTH_LONG).show();}

return result;}

private class HttpAsyncTask extends
    AsyncTask<String, Void, String> {

    // This class inherits AsyncTask

    // that creates a background thread

    // the following methods should be overrode

    @Override
```

```
        protected String doInBackground(String... params) {  
            // this method is the actual background work  
  
            // i.e the HTTP Request  
  
            return login(params[0], params[1], params[2]);  
        }  
  
        @Override  
  
        protected void onPostExecute(String result) {  
            // this method is called when the request is finished successfully  
  
            // the returned data (json text) is passed as an input parameter  
  
            // this text should be parsed here and processed as needed  
  
            try {  
  
                // here we should parse the json text we should have and display it  
                // using some control  
  
                // we'll use a ListView to display items  
  
                // Returned json data using the login web API has only one property  
                // i.e. "Result"  
  
                JSONObject jsonResult = new JSONObject(result);  
  
                result = jsonResult.getString("Result");  
  
  
                // Converting the returned String result into Integer  
  
                // and pass it to the second activity to be passed to server when needed  
  
                // or display an error message when there is an error  
  
                int loginResult = Integer.parseInt(result);  
  
                if (loginResult != -1) {  
  
                    Intent intent = new Intent  
                        (MainActivity.this, activity_enrollments.class);  
  
                    intent.putExtra("StudentID", loginResult);  
                }  
            }  
        }  
    }  
}
```



```

startActivity(intent);
}

else {

Toast.makeText(MainActivity.this, "Invalid User Name or Password",
Toast.LENGTH_LONG).show();

    }

    } catch (Exception e) {

Toast.makeText(MainActivity.this, "Connection Error",
Toast.LENGTH_LONG).show();

}}}}

```

- للتحقق من اسم المستخدم وكلمة السر، يتم العمل وفق التسلسل التالي:
 - تقوم الإجرائية `OnClick` المرتبطة مع زر الأمر الموجود على الواجهة باستدعاء الطريقة `execute` للصف `HttpAsyncTask` مع تمرير عنوان واجهة الوب الموافق مع اسم المستخدم وكلمة السر المدخلين في عناصر التحكم الموافقة:
 - (يتم استخدام الصف `HttpAsyncTask` لاستدعاء واجهة الوب في إجرائية غير متزامنة في نيسب `thread` خاص لأن الأندرويد لا يسمح بإجراء الاتصال بواجهات الوب في النيسب الرئيسي).

```

new HttpAsyncTask().execute("http://10.0.2.2/api/Students",
usernameEditText.getText().toString(),
passwordEditText.getText().toString());

```

- يؤدي استدعاء الطريقة `execute` أولاً إلى استدعاء الإجرائية `doInBackground` للصف `HttpAsyncTask`

```

protected String doInBackground(String... params) {

    // this method is the actual background work

    // i.e the HTTP Request

    return login(params[0], params[1], params[2]);

}

```

- تقوم الإجرائية `doInBackground` بدورها باستدعاء الطريقة `login` والتي تأخذ ثلاثة معاملات الأول هو عنوان واجهة الوب والثاني اسم المستخدم والثالث كلمة السر.

- نقوم في الإجرائية login بتنفيذ طلب HTTP من النمط POST والذي يُعيد في مثالنا سلسلة JSON تحوي معرف الطالب.

```
private String login(String url, String username, String password) {  
    String result = "";  
    try {  
        // Create http request (POST Request)  
        HttpClient httpClient = new DefaultHttpClient();  
        HttpPost httpPost = new HttpPost(url);  
  
        // Parameters that should be passed to server within the POST Request  
        List<NameValuePair> nameValuePairs = new  
            ArrayList<NameValuePair>(2);  
        nameValuePairs.add(new  
            BasicNameValuePair("UserName", username));  
        nameValuePairs.add(new  
            BasicNameValuePair("Password", password));  
  
        httpPost.setEntity(new UrlEncodedFormEntity(nameValuePairs));  
  
        // Execute the request and get the response  
        HttpResponse httpResponse = httpClient.execute(httpPost);  
  
        // Reading response using Java stream readers  
        // Convert the stream into String  
        InputStream inputStream =  
            httpResponse.getEntity().getContent();  
        BufferedReader bufferedReader = new BufferedReader  
            (new InputStreamReader(inputStream));
```

```
String line = "";

    while ((line = bufferedReader.readLine()) != null) {

        result += line;

    }

    inputStream.close();

} catch (ClientProtocolException e) {

    Toast.makeText(MainActivity.this, "Connection Error",
Toast.LENGTH_LONG).show();

} catch (IOException e) {

    Toast.makeText(MainActivity.this, "Connection Error",
Toast.LENGTH_LONG).show();

}

    return result;
}
```

- بعد الانتهاء من تنفيذ الإجراءات السابقة في الخلفية. ينتقل التحكم إلى الإجراءات `onPostExecute` للصف `HttpAsyncTask` والتي تقوم بمعالجة سلسلة JSON المعادة والتي تحوي إما معرف الطالب في حال كانت المعلومات المدخلة صحيحة أو -1 في حال كانت خاطئة.
- في الحالة الأولى يتم تمرير معرف الطالب إلى الواجهة الثانية أما في الحالة الثانية فيتم إظهار رسالة خطأ موافقة.

```
protected void onPostExecute(String result) {

    // this method is called when the request is finished successfully
    // the returned data (json text) is passed as an input parameter
    // this text should be parsed here and processed as needed

    try {

        // here we should parse the json text we should have and display it
        // using some control

        // we'll use a ListView to display items
```

```
// Returned json data using the login web API has only one property
// i.e. "Result"

JSONObject jsonResult = new JSONObject(result);

result = jsonResult.getString("Result");

// Converting the returned String result into Integer
// and pass it to the second activity to be passed to server when needed
// or display an error message when there is an error

int loginResult = Integer.parseInt(result);

if (loginResult != -1) {

    Intent intent = new Intent
(MainActivity.this, activity_enrollments.class);

    intent.putExtra("StudentID", loginResult);

    startActivity(intent);

}

else {

    Toast.makeText(MainActivity.this, "Invalid User Name or
Password", Toast.LENGTH_LONG).show();

}

} catch (Exception e) {

    Toast.makeText(MainActivity.this, "Connection Error",
Toast.LENGTH_LONG).show();

}

}
```

- يكون نشاط الواجهة الثانية activity_enrollments.xml مبدئياً (نقوم أولاً بهدف التجريب فقط بإظهار رسالة ترحيبية تحوي معرف الطالب المرسل من الواجهة الأولى):

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"

android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"

android:paddingRight="@dimen/activity_horizontal_margin"

android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin"

tools:context="EnrollmentsActivity">

<TextView

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:textAppearance="?android:attr/textAppearanceMedium"

    android:text="Welcome "

    android:id="@+id/textView"

    android:layout_alignParentTop="true"

    android:layout_alignParentLeft="true"

    android:layout_alignParentStart="true"

    android:layout_alignParentRight="true"

    android:layout_alignParentEnd="true" />

</RelativeLayout>
```

- ويكون دور الإجرائية إظهار الرسالة الترحيبية:

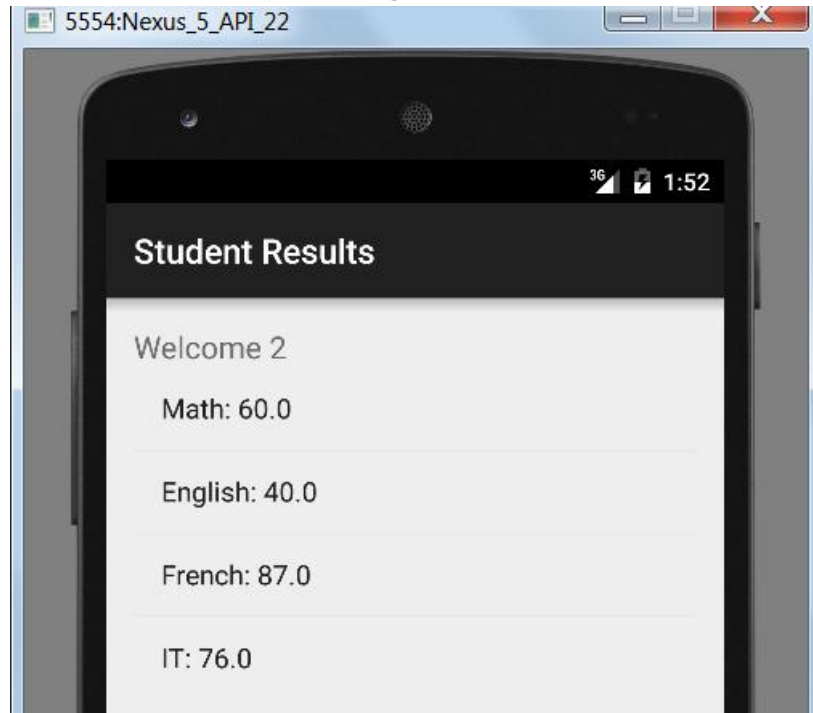
```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_enrollments);  
  
    int studentID = getIntent().getIntExtra("StudentID", 0);  
  
    TextView = (TextView)findViewById(R.id.textView);  
    textView.setText(textView.getText().toString() + studentID);}
```

- كما في المثال التالي:



واجهة إظهار نتائج الطالب

- سنقوم الآن بتعديل الواجهة الثانية وبحيث أنه عند إدخال بيانات المستخدم في الواجهة الأولى بشكل ناجح، يتم الانتقال إلى الواجهة الثانية لعرض نتائج الطالب في المواد المسجل بها:



- يكون النشاط الموافق للواجهة :activity_enrollments.xml

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"

    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"

    android:paddingRight="@dimen/activity_horizontal_margin"

    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"

    tools:context="EnrollmentsActivity">
<TextView

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"
```

```

        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Welcome "
        android:id="@+id/textView"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />
<ListView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/listView"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_below="@+id/textView" />
</RelativeLayout>

```

• ويكون ملف الكود الموافق :activity_enrollments.java

```

package com.example.basel.webapi;

import android.content.Intent;
import android.os.AsyncTask;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;

```



```
import android.view.Menu;

import android.view.MenuItem;

import android.widget.AdapterView;

import android.widget.EditText;

import android.widget.ListView;

import android.widget.TextView;

import android.widget.Toast;

import org.apache.http.HttpResponse;

import org.apache.http.NameValuePair;

import org.apache.http.client.ClientProtocolException;

import org.apache.http.client.HttpClient;

import org.apache.http.client.entity.UrlEncodedFormEntity;

import org.apache.http.client.methods.HttpGet;

import org.apache.http.client.methods.HttpPost;

import org.apache.http.impl.client.DefaultHttpClient;

import org.apache.http.message.BasicNameValuePair;

import org.json.JSONArray;

import org.json.JSONObject;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.util.ArrayList;

import java.util.List;
```

```
public class activity_enrollments extends ActionBarActivity {  
    TextView textView;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_enrollments);  
  
        int studentID = getIntent().getIntExtra("StudentID", 0);  
        String url = "http://10.0.2.2/api/Enrollments/" + studentID;  
        new HttpAsyncTask().execute(url);  
  
        textView = (TextView)findViewById(R.id.textView);  
        textView.setText(textView.getText().toString() + studentID);  
    }  
  
    private String getEnrollments(String url) {  
        String result = "";  
        try {  
            HttpClient httpClient = new DefaultHttpClient();  
            HttpGet httpGet = new HttpGet(url);  
            HttpResponse httpResponse = httpClient.execute(httpGet);  
            InputStream inputStream =  
                httpResponse.getEntity().getContent();  
  
            BufferedReader bufferedReader = new BufferedReader(new
```

```
        InputStreamReader(inputStream));

    String line = "";

    while ((line = bufferedReader.readLine()) != null) {

        result += line;

    }

    inputStream.close();

    } catch (ClientProtocolException e) {

        Toast.makeText(activity_enrollments.this, "Connection Error",
            Toast.LENGTH_LONG).show();

    } catch (IOException e) {

        Toast.makeText(activity_enrollments.this, "Connection Error",
            Toast.LENGTH_LONG).show();

    }

    return result;

}

private class HttpAsyncTask extends AsyncTask<String, Void, String>
{

    @Override

    protected String doInBackground(String... params) {

        return getEnrollments(params[0]);

    }

    @Override

    protected void onPostExecute(String result) {
```

```
Toast.makeText(activity_enrollments.this, "DONE",
Toast.LENGTH_LONG).show();

try {

    // here we should parse the json text we should have and display it
    // using some control

    // we'll use a ListView to display items

    // Converting the returned json String into a list of
    //String objects to be displayed

    // or display an error message when there is an error

    // Define an empty list of Strings

    ArrayList<String> items2Display = new ArrayList<String>();

    // Convert the returned string into a list of json objects
    JSONArray enrollments = new JSONArray(result);

    for (int i = 0; i < enrollments.length(); i++) {
        // Parse each json object and put it into one String
        JSONObject enrollment = enrollments.getJSONObject(i);
        String item2Display = "";
        item2Display += enrollment.getString("SubjectName");
        item2Display += ": ";
        item2Display += enrollment.getDouble("Mark");

        // Add the String object to the list of Strings above
        items2Display.add(i, item2Display);}

    // Create an adapter of String objects to fill the listView
    // that displays the results

    // this adapter uses the standard layouts provided by Android system
```

```
// i.e. android.R.layout.simple_list_item_1 THAT
// CONTAINS android.R.id.text1

// Developer can create a custom adapter to display the results
// in a more complicated and advanced layout

// but the data should be filled within a custom class like (Enrollment)

// this class should be defined within this java code

// and a list of its objects should be passed to the custom adapter

ArrayAdapter<String> adapter = new
    ArrayAdapter<String>(activity_enrollments.this,
        android.R.layout.simple_list_item_1, android.R.id.text1,
        items2Display);

// Get the listView and assign the previously defined adapter to it

ListView listView = (ListView)findViewById(R.id.listView);

    listView.setAdapter(adapter);

} catch (Exception e) {

Toast.makeText(activity_enrollments.this, "Data Error",
    Toast.LENGTH_LONG).show(); }}}
```

- لإظهار نتائج الطالب يتم العمل وفق التسلسل التالي:
- تقوم الإجرائية onCreate باستدعاء الطريقة execute للصف AsyncTask مع تمرير عنوان واجهة الوب الموافق مع رقم الطالب:

```
int studentID = getIntent().getIntExtra("StudentID", 0);
String url = "http://10.0.2.2/api/Enrollments/" + studentID;
new AsyncTask().execute(url);
```

- يؤدي استدعاء الطريقة execute أولاً إلى استدعاء الإجرائية doInBackground للصف :HttpAsyncTask

```
protected String doInBackground(String... params) {

    return getEnrollments(params[0]);

}
```

- تقوم الإجرائية doInBackground باستدعاء الطريقة getEnrollments والتي تأخذ معامل وحيد هو عنوان واجهة الوب.

- نقوم في الإجرائية getEnrollments بتنفيذ طلب HTTP من النمط GET والذي يُعيد في مثالنا سلسلة JSON تحوي نتائج الطالب (مادة/علامة).

```
private String getEnrollments(String url) {  
    String result = "";  
    try {  
  
        HttpClient httpClient = new DefaultHttpClient();  
        HttpGet httpGet = new HttpGet(url);  
        HttpResponse httpResponse =  
            httpClient.execute(httpGet);  
        InputStream inputStream =  
            httpResponse.getEntity().getContent();  
  
        BufferedReader bufferedReader = new  
            BufferedReader(new InputStreamReader(inputStream));  
        String line = "";  
        while ((line = bufferedReader.readLine()) != null) {  
            result += line;  
        }  
        inputStream.close();  
    } catch (ClientProtocolException e) {  
        Toast.makeText(activity_enrollments.this, "Connection  
Error", Toast.LENGTH_LONG).show();  
    } catch (IOException e) {  
        Toast.makeText(activity_enrollments.this, "Connection  
Error", Toast.LENGTH_LONG).show();  
    }  
    return result;  
}
```

- بعد الانتهاء من تنفيذ الإجراءات السابقة في الخلفية. ينتقل التحكم إلى الإجراءات `onPostExecute` للصف `HttpAsyncTask` والتي تقوم بمعالجة سلسلة `JSON` ومن ثم إظهار النتائج في عنصر التحكم `ListView`.

```
protected void onPostExecute(String result) {

    Toast.makeText(activity_enrollments.this, "DONE",
    Toast.LENGTH_LONG).show();

    try {

        ArrayList<String> items2Display =
            new ArrayList<String>();

        // Convert the returned string into a list of json objects

        JSONArray enrollments = new JSONArray(result);

        for (int i = 0; i < enrollments.length(); i++) {

            // Parse each json object and put it into one String

            JSONObject enrollment =
                enrollments.getJSONObject(i);

            String item2Display = "";

            item2Display +=
                enrollment.getString("SubjectName");

            item2Display += ": ";

            item2Display += enrollment.getDouble("Mark");

            // Add the String object to the list of strings above

            items2Display.add(i, item2Display);}

        // Create an adapter of String objects to fill
        // the ListView that displays the results

        // this adapter uses the standard layouts provided by Android system

        // i.e. android.R.layout.simple_list_item_1
        // THAT CONTAINS android.R.id.text1

        ArrayAdapter<String> adapter = new
        ArrayAdapter<String>(activity_enrollments.this,
            android.R.layout.simple_list_item_1, android.R.id.text1,
```

```
items2Display);

// Get the listView and assign the previously defined adapter to it

    ListView listView =
        (ListView)findViewById(R.id.ListView);

    listView.setAdapter(adapter);

} catch (Exception e) {

    Toast.makeText(activity_enrollments.this, "Data Error",
    Toast.LENGTH_LONG).show();

}

}
```