الدليل العملي لمقرر
الأنظمة الموزعة 1479
Distributed Systems

إعداد: د. م. سامر حسني جالودي
جامعة القدس المفتوحة - فرع نابلس

مقدمة

عزيزي الدارس، إن فلسفة النظم الموزعة هو أن يكون لديك شبكة من الحواسيب (العقد nodes) التي يعمل بعض منها على نظام تشغيل مختلف عن نظم التشغيل على باقي العقد (الحواسيب)، وتطبيقات بعضها مكتوب بلغة برمجة قد تكون نفسها أو مختلفة عن باقي العقد. فمثلا، شبكة ما، تحوي جهاز (عقدة) يعمل بنظام تشغيل ويندوز، يعمل عليه تطبيق مكتوب بلغة C# ويرتبط بقاعدة بيانات من مايكروسوفت. وجهاز آخر يعمل بأحد أنظمة لينوكس، يعمل عليه تطبيق مكتوب بلغة Java، ويرتبط بقاعدة بيانات MySQL. وجهاز ثالث، الخ.

يوجد في هذه الحالة العديد من المنصات platforms التي تعمل بنظم تشغيل وتطبيقات وقواعد بيانات تختلف عن بعضها البعض. هنا يأتي دور النظم الموزعة في توصيف بروتوكول يعمل على جسر الهوة بين هذه المنصات من خلال حل مشكلة التبادلية (Interoperability) ويوفر في نفس الوقت تقنية الاتصال بينها.

في هذا الدليل العملي، ستعمل عزيزي الدارس، وبمساعدة مشرفك الأكاديمي وفني المختبر، على إجراء مجموعة من التجارب العملية، من خلال نشاطات صفيّة. حيث ستقوم بتجارب تدعم تطبيقات النظم الموزعة باستخدام لغة جافا، من خلال بروتوكوليّ الاتصال RMI و CORBA. وستعمل أيضا على إجراء تجارب تدعم اكبر شبكة موزعة وهي الانترنت. بالتالي ستستخدم كل من لغة PHP وبروتوكول الاتصال XML-RPC.

قواعد البيانات التي ستستخدم في جميع الحالات، سواء كانت تطبيقات عادية أو تطبيقات الانترنت، هي MySQL. وبالتالي فإن تطبيقات النظم الموزعة وقواعد بياناتها ستكون الهدف الأساسي من هذا الدليل.

النشاط الصفي الأول

استدعاء الطرق عن بعد

Remote Method Invocation

أولا: مقدمة

تدعم لغة جافا العديد من بروتوكولات الاتصالات التي تعتمد على طبقة الشبكة، وتعتبر هذه البروتوكولات حلقة الوصل ما بين التطبيقات البرمجية وقواعد البيانات والمستخدم من جهة وما بين برتوكولات الاتصال في طبقة الشبكة. إحدى هذه البروتوكولات التي تدعمها جافا هي استدعاء الطرق عن بعد Remote Method Invocation، والتي تعتبر وريثة بروتوكول استدعاء الإجراء عن بعد Remote Procedure Call، إلا أن لغة جافا، تعرف طرق methods بدلا من الإجراءات procedures أو الدوال functions. أيضا تدعم لغة جافا معمارية وسيط طلب الأهداف المشتركة Common Object Request Broker Architecture أو المعروفة بالاختصار CORBA. تقدم لغة جافا مجموعة من المكتبات التي تيسّر عمل المبرمج بحيث يمكن كتابة برامج تعمل على الاتصال بين الحواسيب باستخدام CORBA أو RMI وفي نفس الوقت الربط مع قواعد البيانات، أو إنشاء GUI خاص بالمستخدمين. هذه التجربة، سيتم التعامل مع RMI، وفي التجربة الثانية سيتم التعامل مع CORBA.

عزيزي الدارس، هناك عدة طرق لتصنيف (Compile) وتنفيذ (Run) برامج جافا. منها:

1. باستخدام بيئة التطوير (eclipse) والذي يمكن تحميله (download) من الموقع التالي: http://www.eclipse.org/downloads/
2. باستخدام بيئة التطوير (JBuilder)، ولكن نسخة تجريبية، والذي يمكن تحميله من الموقع التالي: https://downloads.embarcadero.com/free/jbuilder
3. باستخدام بيئة التطوير (JCreator)، والذي يمكن تحميله من الموقع التالي: http://www.jcreator.com/
4. باستخدام بيئة التطوير (NetBeans) والذي يمكن تحميله من الموقع التالي: https://netbeans.org/downloads/
5. باستخدام موقع شركة أوراكل (Oracle)، يمكن تحميل معدات تطوير جافا (download JDK) : http://www.oracle.com/technetwork/java/javase/downloads/index.html ومن ثم بعد ذلك يمكن العمل على بيئة JDK بشكل مباشر أو من خلال برنامج (Textpad) للتسهيل أثناء التصنيف والتنفيذ من الموقع: http://www.textpad.com/download/

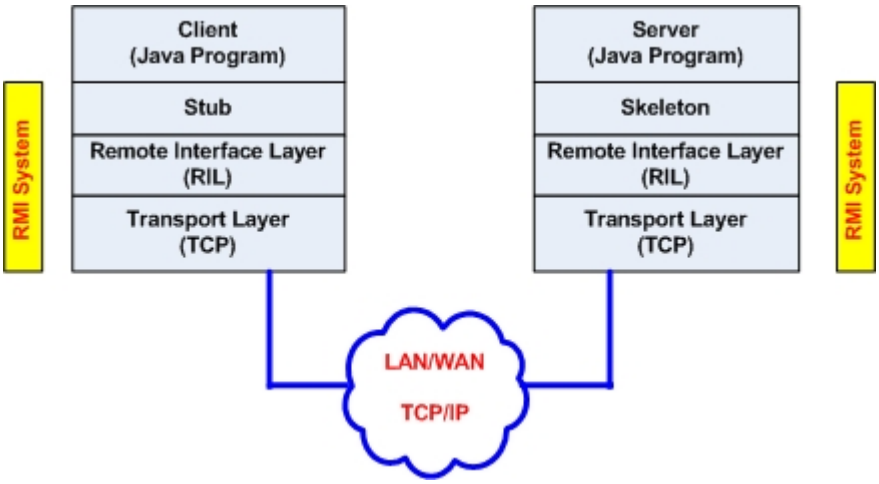
سيتم اعتماد الطريقة الأخيرة لتنفيذ التجربتين الأولى والثانية من هذا الدليل العملي وذلك لغايات تحقيق الأهداف المرجوة من المقرر. مع العلم أنه بإمكانك استخدام أي واجهة تطوير أخرى (IDE) تناسبك، وذلك بالتنسيق مع مشرفك الأكاديمي. في التجربة الأخيرة، سيتم استخدام بيئة التطوير المتكاملة NetBeans IDE، حيث ستقوم، عزيزي الدارس، بربط قاعدة البيانات مع خادم تقنيات الاتصال في جافا.

ثانيا: الأهداف

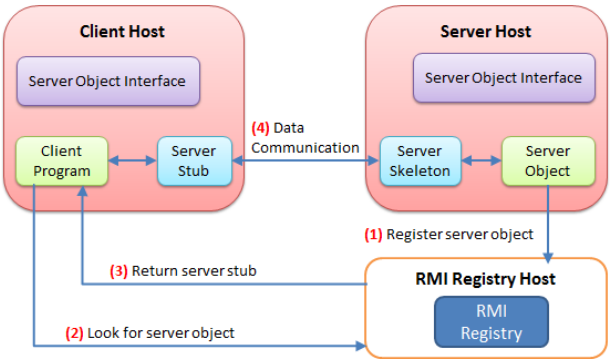
1. أن تتعرف على آلية عمل إحدى تقنيات الاتصال في جافا والتي تسمى "استدعاء الطرق عن بعد" RMI.
2. أن تعرف دور كل من stub و skeleton و Registry في تقنية الاتصال RMI.
3. أن تتعرف كيفية برمجة واستخدام تقنية الاتصال عن بعد باستخدام بروتوكول RMI، واستحضار الطرق عن بعد.
4. أن تحدد مشكلة تقنية الاتصال RMI الرئيسية، والتي تعد إحدى مساوئ RMI.

ثالثا: خطوات عمل التجربة

نبدأ بشرح فلسفة بروتوكول الاتصال RMI. حيث يعمل هذا البروتوكول بين طبقة النقل transport layer وطبقة التطبيقات، كما هو موضح في الصورة أدناه. فيكتب المبرمج برنامجا بلغة جافا لكل من الخادم server والزبون client، في طبقة التطبيقات application layer، ثم يتم استخدام بروتوكول RMI من أجل إنشاء اتصالا بين الخادم والزبون. تعمل تقنية الاتصال RMI في طبقة الشبكة حيث تستخدم بروتوكول TCP، لتبادل البيانات بين الخادم والزبون باستخدام شبكة LAN/WAN تعمل على حزمة بروتوكولات الانترنت وهي حزمة TCP/IP.



آلية إنشاء الاتصال وتبادل البيانات موضحة في الصورة أدناه. يبدأ الخادم كخطوة أولى بتسجيل منهاجه (طريقته method)، وهو المنهاج الذي سيتم مناداته من قبل الزبائن، في داخل سِّجِل Registry، خاص بنظام RMI والذي قد يتواجد على جهاز آخر طبعا. بعد ذلك وعندما يريد الزبون client، وكخطوة ثانية، بالبحث عن المنهاج، يبدأ بالبحث عنها في السجل Registry، فتعيد له أرْوِمَة (عقب) الخادم server stub، والتي تكون من جهة الزبون. فمن خلالها يمكن للزبون وكخطوة رابعة الاتصال مع مفتاح هيكلي (skeleton) جهة الخادم لتبادل البيانات.



https://camo.githubusercontent.com/fe96bc186fc00c31f975b687697565fc71343635/687474703a2f2f6c79636f672e636f6d2f77702d636f6e74656e742f75706c6f6164732f323031312f30332f6a6176612d726d692d6f766572766965772e706e67

يتبين عزيزي الدارس، أننا بحاجة إلى ثلاثة برامج، برنامج الواجهة remote interface، ويجب أن تحمل الاسم الذي سيتم مناداته. وبرنامج ثاني خاص بالخادم remote server. وبرنامج ثالث بسيط يمثل الزبون client. نبدأ بالهدفين الأول والثاني واللذين يمكن تحقيقهما من خلال البرامج التالية. أكتب كل برنامج من البرامج التالية في ملف منفصل إما باسم الصنف أو باسم الواجهة.

الملف الأول Hello.java

يجب استحضار مكتبة RMI كما يلي import java.rmi.*; أو تفصيل الأجزاء التي نريدها من تلك المكتبة كما هو موضح في البرنامج أدناه. حيث نعرف واجهة Hello، والتي تعرف بدورها المنهج الذي سيتم استحضاره عن بعد.

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Hello extends Remote
{
   String sayHello() throws RemoteException;
}
```

الملف الثاني Server.java

لكتابة برنامج الخادم، نتبع الخطوات التالية:

- يقوم الخادم بتنفيذ الواجهة Hello، وعليه يجب تعريف المنهج (الطريقة) التي سيقوم الزبون بمناداتها للتنفيذ وهي sayHello().
- يجب عمل مثيل من الخادم وليكن بإسم obj.
- لا يمكن عمل مثيل من واجهة في لغة جافا، ولكن يمكن الإشارة إلى هدف ينفذ الواجهة بالتحويل إلى نوع الواجهة. لذلك أنشئ عقِب من Hello وليكن بإسم stub، بحيث تعمل على تحويل نوع ناتج تنفيذ المنهج exportObject إلى نفس نوع الواجهة.
- أنشئ مثيل من السجل Registry من أجل تسجيل وحفظ مكان الطريقة method المراد مناداتها للتنفيذ.
- الآن أربط أسم الواجهة Hello مع أسم العقب stub. بالتالي أصبحت أسم الواجهة وما تحويه من طرق (مناهج) مسجلا في السجل.

```
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class Server implements Hello
{
   public Server() {}
   public String sayHello()
   {
      return "Hello, world!";
   }
   public static void main(String args[])
   {
      try
      {
         Server obj = new Server();
         Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);
         // Bind the remote object's stub in the registry
         Registry registry = LocateRegistry.getRegistry();
         registry.bind("Hello", stub);
         System.err.println("Server ready");
      }
      catch (Exception e)
      {
         System.err.println("Server exception: " + e.toString());
         e.printStackTrace();
      }
   }
}
```

البرنامج الثالث Client.java

هنا نطلب من مثيل السجل Registry أن يبحث عن الطريقة method المراد مناداتها في الواجهة Hello وهي الطريقة sayHello(). فنمرر عنوان IP المضيف الذي يحوي السجل، ليدلّنا على موقع الطريقة sayHello()، ثم نطبع ناتج التنفيذ.

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
public class Client
{
   private Client() {}
   public static void main(String[] args)
```

```
    {
        String host = "192.168.1.100";
        try
        {
            Registry registry = LocateRegistry.getRegistry(host);
            Hello stub = (Hello) registry.lookup("Hello");
            String response = stub.sayHello();
            System.out.println("response: " + response);
        }
        catch (Exception e)
        {
            System.err.println("Client exception: " + e.toString());
            e.printStackTrace();
        }
    }
}
```

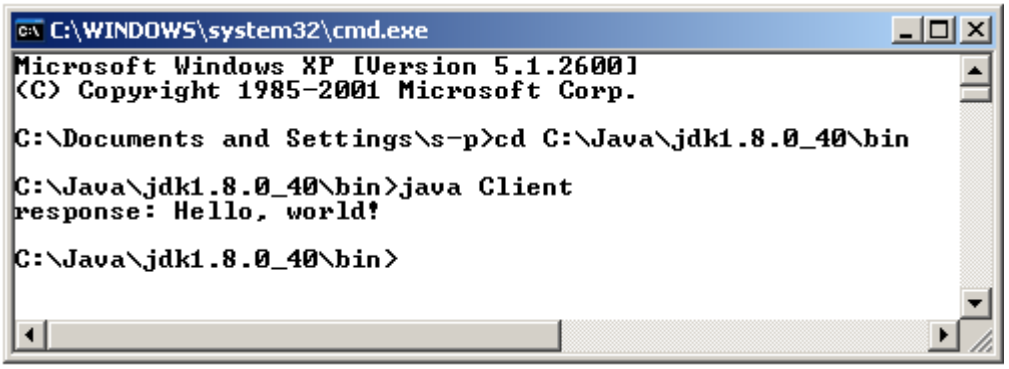
لتنفيذ المثال السابق، اتبع الخطوات التالية:

- ضع جميع الملفات السابقة (Hello.java, Cient.java, Server.java)، في مجلد bin، في بيئة جافا التطويرية JDK.
- افتح نافذة DOS، ثم انتقل إلى المجلد الذي يحوي الملفات الثلاث في ال bin.
- صنف compile، البرامج الثلاثة معا مستخدما الأمر: javac Hello.java Client.java Server.java
- افتح شاشة الدوس cmd، ثم ابدأ السِجل مستخدما الأمر start rmiregistry.
- من نفس نافذة DOS السابقة التي بدأت بها السجل، شغّل (ابدأ) برنامج الخادم server: مستخدما الأمر التالي:

"start java -Djava.rmi.server.codebase=file:C:/Java/jdk1.8.0_40/bin/ Server".

لاحظ أن هناك نافذتين يتم فتحهما عند تنفيذ أمر تشغيل كل من السجل registry، وكذلك الخادم server.

الآن ومن نافذة DOS مستقلة من على نفس الجهاز أو جهاز زميلك، شغل برنامج الزبون client، مستخدما الأمر java Client.



ملاحظة: قد لا يعمل الخادم على جهاز منفصل بسبب مشاكل في security، فحاول حلها مع مدير الشبكة إن حصلت تلك المشكلة.

تدريب 1: اشرح آلية عمل المثال السابق بعد الاطلاع على الرابط التالي من موقع أوراكل:
https://docs.oracle.com/javase/8/docs/technotes/guides/rmi/hello/hello-world.html

النشاط الصفي الثاني

معمارية وسيط طلب الأهداف المشتركة

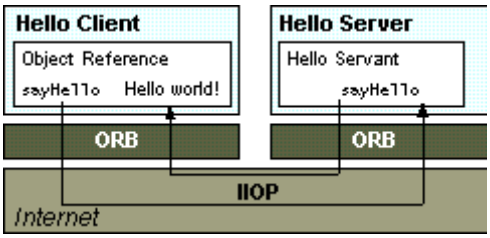
Common Object Request Broker Architecture (CORBA)

مقدمة

عزيزي الدارس، تعتبر لغة تعريف الواجهات Interface Definition Language (IDL) من تقنيات الأهداف الموزعة، حيث تتواصل مجموعة من الأهداف، متواجدة على نظم تشغيل مختلفة، من خلال شبكة. هذه اللغة تسهل من التواصل بين الأهداف بغض النظر عن اللغات المستخدمة سواء كانت Java أو C++ أو COBOL أو غيرها من اللغات. طبعا هذا ممكن لأن IDL لا تعتمد على لغة برمجة بعينها، وإنما تتبنى مبدأ عمل معمارية وسيط طلب الأهداف المشتركة Common Object Request Broker Architecture (CORBA). تم تطوير هذه المعمارية من قبل تجمع ضخم من الشركات الصانعة يسمى Object Management Group (OMG) من أجل التواصل بين الأهداف الموزعة (النظم الموزعة). فكل لغة من لغات البرمجة التي تدعم CORBA لديها طريقتها في النقل mapping من IDL إلى تلك اللغة. فمثلا Java IDL تدعم النقل إلى لغة جافا. للتواصل بين أهداف متواجدة على منصات مختلفة (اختلاف نظم التشغيل ولغات برمجة)، توفر معمارية CORBA وسيط يسمى ORB، وهو في لغة جافا عبارة عن مكتبة أصناف class library، تمكن التواصل بين تطبيقات Java IDL وتطبيقات CORBA.

هذه التجربة تعلمك، عزيزي الدارس، القواعد الأساسية لبناء تطبيقات موزعة قائمة على معمارية CORBA باستخدام Java IDL. حيث ستعمل على بناء مثال Hello World كنظام موزع، والذي له عملية واحدة فقط وهي إعادة سلسلة string للطباعة.

العلاقة التي تربط أي نظام موزع (أو أهداف موزعة) لها جانبان وهما الزبون client والخادم server. يوفر الخادم واجهة بعيدة remote interface، وعليه فإن الزبون يستدعي واجهة بعيدة. هذه العلاقة مشتركة مع جميع المعايير التي تتبنى الأهداف الموزعة مثل CORBA و RMI و RMI-IIOP. لاحظ أن الكلمات خادم وزبون، تعرفان هنا تواصل بين أهداف object-level interaction وليس تواصل بين تطبيقات application-level interaction، بحيث يصبح أي تطبيق لهذا الهدف عبارة عن خادم ولذاك الهدف عبارة عن زبون. أي أن التطبيق نفسه يمكن أن يحوي زبون وخادم في نفس الوقت. في الحقيقة، هدف واحد ممكن أن يكون زبون لواجهة تابعة لهدف بعيد، وفي نفس الوقت، ينفذ واجهة سيتم استدعائها عن بعد من طرف أهداف أخرى. الصورة أدناه توضح كيف أن هدف موزع بطريقة واحدة distributed object one-method مشترك بين خادم وزبون CORBA لتنفيذ صنف التطبيق Hello World.



https://docs.oracle.com/javase/8/docs/technotes/guides/idl/GShome.html

في جهة الزبون، يحتوي التطبيق على مرجع للهدف البعيد. المرجع الهدف له طريقة عقب stub، والتي مؤقتا تحلّ مكان (تستبدل) الطريقة التي سيتم استدعائها عن بعد. العقب stub هو في الحقيقة مبَرمَج في داخل ORB، وبالتالي استدعاؤه يستحضر أيضا إمكانيات اتصال ORB، والتي بدورها تحوِّل الاستحضار إلى الخادم.

في جهة الخادم، فإن ORB يستخدم كود المفتاح الهيكلي skeleton، لترجمة الاستدعاء عن بعد إلى استدعاء طريقة method على الهدف المحلي. يعمل المفتاح الهيكلي skeleton على ترجمة الاستدعاء call، وأية قيم parameters إلى النسق المناسب للتنفيذ ويستدعي الطريقة التي تم استحضارها. عندما يتم تنفيذ الطريقة method، وتعيد قيمة، فإن كود المفتاح الهيكلي skeleton يحوّل نتائج أو أخطاء، ويعيد إرسالها إلى الزبون عن طريق ORB. ما بين ORBs فإن الاتصال يتم عن طريق بروتوكول IIOP وهو اختصار Internet Inter ORB Protocol. بروتوكول IIOP، والذي يعتمد على بروتوكولات TCP/IP، يعرف بدوره كيفية قيام ORBs بتمرير معلومات فيما بينها. كما هو الحال في كل من CORBA و IDL، فإن IIOP هو معيار اتصال معرف من قبل تجمع OMG.

الأهداف

1. أن تتعرف على لغة تعريف الواجهات IDL.
2. أن تتعرف على معمارية وسيط طلب الأهداف المشتركة CORBA، وبيئتها.
3. أن تبني مثالا يوضح طريقة الاتصال بين الأهداف مستخدما برامج جافا وتقنية CORBA.
4. أن تذكر الفرق بين تقنيتيّ الاتصال RMI و CORBA.

خطوات عمل التجربة

هذه التجربة تعلمك عزيزي الدارس أساسيات بناء نظام موزع باستخدام تقنية الاتصال CORBA، حيث تستخدم لغة جافا في بناء برامج المثال. سيتم بناء المثال الكلاسيكي Hello World، كتطبيق موزع، له طريقة واحدة فقط تعيد سلسلة للطباعة. بالرغم من بساطة هذا المثال، إلا أنه يعلمك بناء معظم برامج CORBA التي تستخدم الاستحضار الثابت static invocation. يجب بناء ثلاثة برامج، الأول برنامج الواجهة باستخدام لغة IDL، والثاني برنامج الخادم server باستخدام لغة جافا، والثالث برنامج الزبون client باستخدام لغة جافا أيضا.

الملف الأول Hello.idl

الخطوة الأولى على طريق إنشاء تطبيق CORBA هو أن تحدد جميع الأهداف و واجهاتها باستخدام لغة IDL، والتي لها تركيب مشابه للغة C++، والتي يمكنها تعريف وحدات modules، واجهات interfaces، تراكيب بيانات data structures، وأكثر. الكود التالي مكتوب بلغة IDL، ويصف هدف من نوع CORBA، والذي طريقته ()sayHello تعيد سلسلة عند مناداتها وطريقته ()shutdown تغلق الوسيط ORB.

```
module HelloApp
{
	interface Hello
	{
		string sayHello();
		oneway void shutdown();
	};
};
```

الملف الثاني HelloServer.java

يتكوّن ملف الخادم من صنفين two classes، المستخدَم servant والخادم server. المستخدَم HelloImpl هو تنفيذ لملف IDL المسمى Hello الذي يعرف واجهة interface في داخل وحدة module. كل مثيل Hello ينفَّذ بوساطة مثيل HelloImpl. المستخدَم هم مثيل صنف مشتق من HelloPOA، والذي تم توليده بشكل أوتوماتيكي عند تصنيف ملف IDL بوساطة الأمر idlj. المستخدَم يحتوي على طريقة واحدة لكل عملية IDL، وفي هذا المثال هما الطريقتان ()sayHello و ()shutdown. في الحقيقة، فإن طرق المستخدَم هي نفس طرق جافا الاعتيادية. الكود الآخر للتعامل مع ORB، لقيادة وترتيب marshalling المتغيرات والنتائج، يتم تزويدها بوساطة المفتاح الهيكلي skeleton.

يحوي الصنف HelloServer على منهج ()main، والذي:

- ينشئ بقيم مبدئية مثيل ORB.
- يجلب مرجع للجذر Portable Object Adapter (POA) ويفعّل POAManager.
- إنشاء مثيل من المستخدَم servant، وإخبار وسيط ORB بذلك. يتم إنشاء المثيل من خلال تنفيذ هدف Hello واحد.
- الحصول على مرجع هدف CORBA لسياق التسمية naming context حيث سيتم تسجيل هدف CORBA الجديد فيه.
- الحصول على سياق التسمية الجذر.
- تسجيل register الهدف الجديد في سياق التسمية بمسمى Hello.
- انتظار استحضار invocation الهدف الجديد من الزبون.

ملاحظة: إن مكيّف الهدف object adapter، عبارة عن وسيلة لربط طلب request، باستخدام مرجع الهدف بكود معين، مع خدمة الطلب. مكيّف الهدف المحمول POA (قابلة للعمل)، هو نوع من أنواع object adapter، والذي:

- يسمح للمبرمجين بإنشاء أهداف تنفيذية قابلة للعمل بين منتجات ORB مختلفة.
- يوفر الدعم لأهداف بمعرّف مستمر persistent، كما هو الحال في المثال السابق والذي يمثل persistent example.
- يوفر الدعم لتفعيل شفاف transparent activation للأهداف.

- السماح لمستخدّم معين بدعم العديد من معرفات الأهداف بشكل متزامن.

```
import HelloApp.*;

import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;

import java.util.Properties;

class HelloImpl extends HelloPOA
{
	private ORB orb;
	public void setORB(ORB orb_val) {    orb = orb_val;  }
	// implement sayHello() method
	public String sayHello() {  return "\nHello world !!\n";  }
	 // implement shutdown() method
	public void shutdown() {    orb.shutdown(false);  }
}

public class HelloServer
{
	public static void main(String args[])
	{
		try
		{
			 // create and initialize the ORB
			ORB orb = ORB.init(args, null);
			// get reference to rootpoa & activate the POAManager
			POA rootpoa = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
			rootpoa.the_POAManager().activate();
			 // create servant and register it with the ORB
			HelloImpl helloImpl = new HelloImpl();
			helloImpl.setORB(orb);
			// get object reference from the servant
			org.omg.CORBA.Object ref = rootpoa.servant_to_reference(helloImpl);
			Hello href = HelloHelper.narrow(ref);
			// get the root naming context. NameService invokes the name service
			org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
			// Use NamingContextExt which is part of the Interoperable
			// Naming Service (INS) specification.
			NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
			// bind the Object Reference in Naming
			String name = "Hello";
			NameComponent path[] = ncRef.to_name( name );
			ncRef.rebind(path, href);
			System.out.println("HelloServer ready and waiting ...");
			// wait for invocations from clients
			orb.run();
		}
		catch (Exception e)
		{
			System.err.println("ERROR: " + e);
			e.printStackTrace(System.out);
		}
		System.out.println("HelloServer Exiting ...");
	}
}
```

الملف الثالث HelloClient.java

يتم في تطبيق الزبون ما يلي:

- إنشاء ORB بقيم مبدئية initializations.
- الحصول على مرجع إلى جذر سياق التسمية root naming context.
- البحث عن Hello في سياق التسمية naming context، واستقبال مرجع إلى ذلك الهدف من CORBA.
- استحضار طرق هدف وهي sayHello() و shutdown() وطباعة النتيجة.

```
import HelloApp.*;

import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;

public class HelloClient
{
	static Hello helloImpl;
	public static void main(String args[])
	{
		try
		{
			// create and initialize the ORB
			ORB orb = ORB.init(args, null);
			// get the root naming context
			org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
			// Use NamingContextExt instead of NamingContext.
			// This is part of the Interoperable naming Service.
			NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
			// resolve the Object Reference in Naming
			String name = "Hello";
			helloImpl = HelloHelper.narrow(ncRef.resolve_str(name));
			System.out.println("Obtained a handle on server object: " + helloImpl);
			System.out.println(helloImpl.sayHello());
			helloImpl.shutdown();
		}
		catch (Exception e)
		{
			System.out.println("ERROR : " + e) ;
			e.printStackTrace(System.out);
		}
	}

}
```

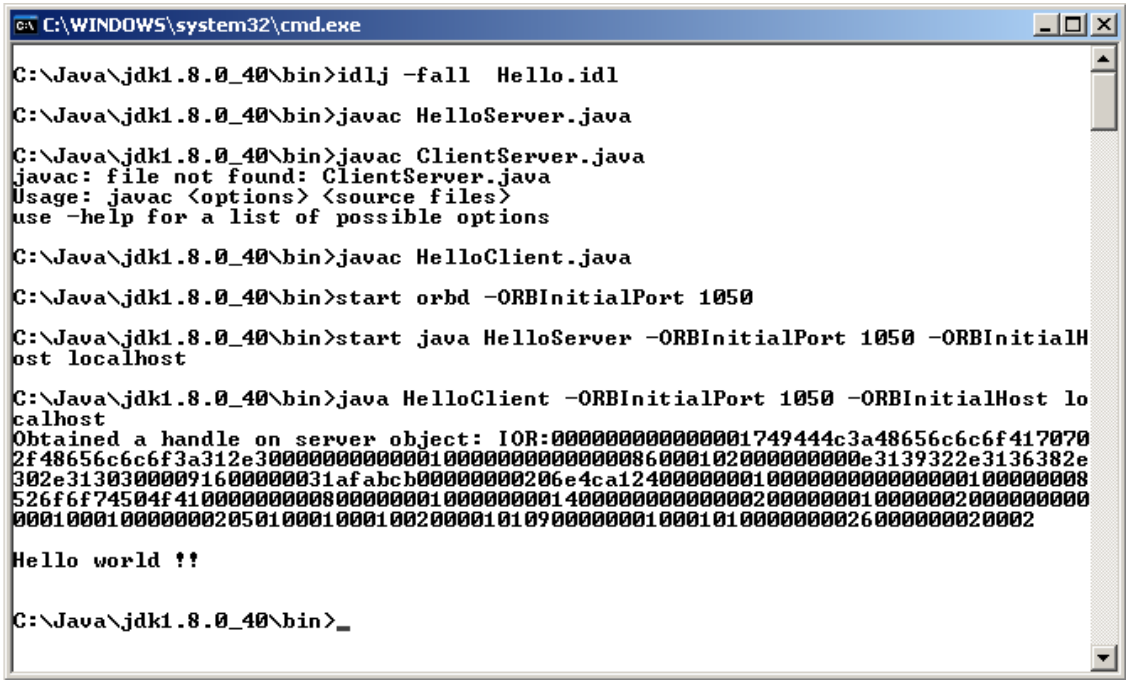
لتنفيذ المثال السابق، ضع جميع ملفاتك في مجلد bin، ثم افتح نافذة DOS. كما هو موضح في الصورة أدناه:

- صنف compile ملف Hello.idl، باستخدام الأمر: idlj -fall Hello.idl
- صنف compile ملف HelloServer.java، باستخدام الأمر javac HelloServer.java
- صنف compile ملف HelloClient.java، باستخدام الأمر javac HelloClient.java
- شغل وسيط ORB على منفذ port رقم 1050 باستخدام الأمر: start orbd –ORBInitialPort 1050
- الآن شغل الخادم على نفس المنفذ من على منصة الجهاز المحلّي باستخدام الأمر التالي:

```
start java HelloServer -ORBInitialPort 1050 -ORBInitialHost localhost
```

- يمكنك الآن، عزيزي الدارس، أن تشغل الزبون باستخدام الأمر التالي:

```
java HelloClient -ORBInitialPort 1050 -ORBInitialHost localhost
```



عند تنفيذ المثال، يتم إعطاء رقم على شكل سلسلة من القيم بالنظام السداسي عشري Hexadecimal يسمى Interoperable Object Reference (IOR)، وهذا عبارة عن محدد مكان لمرجع الهدف. يحوي في داخله مثلا عنوان IP، ورقم منفذ port، ومرجع هدف للخادم servant، الخ.

تدريب: بمساعدة مشرفك الأكاديمي، اشرح آلية عمل المثال السابق بعد الاطلاع على الرابط التالي من موقع أوراكل:

https://docs.oracle.com/javase/8/docs/technotes/guides/idl/jidlExample.html

النشاط الصفي الثالث

مقدمة في لغة برمجة الويب بي اش بي

Introducing PHP

أولا: الأهداف

1. التعرف على بيئة عمل التطبيق XAMPP.
2. التعرف على مجال عمل خادم الويب Apache.
3. التعرف على بيئة عمل لغة الويب PHP.
4. تقديم نبذة عن برامج لغة PHP، وخصوصا المتغيرات والدوال.
5. كتابة برامج بسيطة بلغة PHP وتنفيذها من خلال خادم Apache على جهاز الحاسوب المحلي.

ثانيا: مقدمة

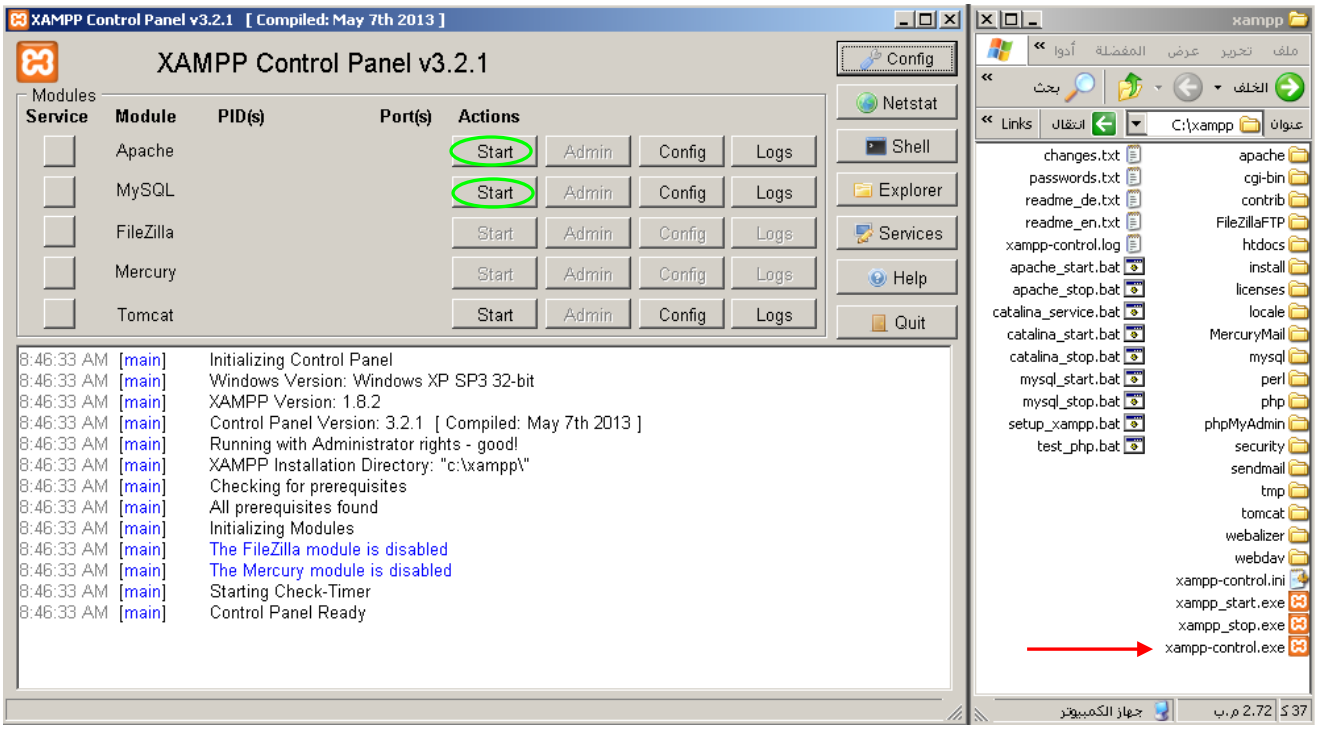
إن لغة برمجة الويب بي إش بي PHP لغة مفتوحة المصدر open source (يمكن تحميلها مجانا من الموقع www.php.net)، والاختصار هو للمصطلح "Hypertext Preprocessor". هي من لغات التعليمات scripting language أي أنها لا تحتاج إلى تصنيف compile مثل لغات Java, C, C++. أما ملفاتها فيجب أن يكون امتدادها ".php" ويمكن أن تحتوي نصوص أو كود إحدى لغات الويب مثل JavaScript, CSS, HTML، طبعا بالإضافة إلى كود اللغة نفسها PHP، حيث يتم تنفيذ هذه الملفات على الخادم server، ونتائج التنفيذ تعود إلى المستخدم على جهاز الحاسوب الزبون client في صفحة HTML، ويمكنها إنتاج ملفات نوع صور أو PDF أو XML، أو حتى أفلام فلاش. يمكن للغة PHP أن تعمل على إنشاء/حذف ملفات على خادم، قراءة/كتابة ملفات على خادم، إنشاء/حذف/تغيير قواعد بيانات على خادم، ويمكنها كذلك التحكم بوصول المستخدمين، وتشفير البيانات.

هذه اللغة مدعومة من قبل العديد من خوادم الويب مثل IIS و Apache، ويمكن أن تنفذ على معظم نظم التشغيل مثل الويندوز واللينوكس واليونكس والماكنتوش. كما أنها تدعم العديد من قواعد البيانات وفي مقدمتها MySQL التي أصبحت تحمل الاسم MariaDB. لذلك ومن أجل العمل بهذه اللغة، يجب تحميل وتنصيب (download & install) خادم ويب web server ، ثم بعد ذلك يمكن تحميل كل من PHP و MySQL. ولكن وبدلا من كل هذه الخطوات، يمكن تحميل تطبيق XAMPP من موقع أصدقاء أباتشي https://www.apachefriends.org/download.html، والذي يحوي واجهة تحكم بالإضافة إلى ما يلي:

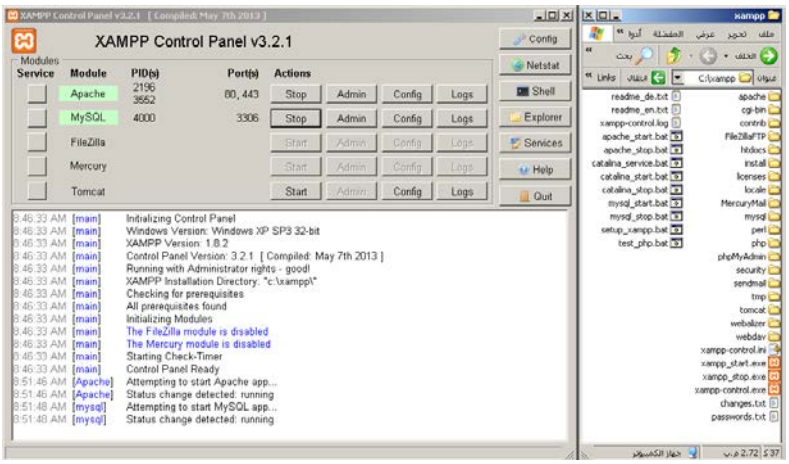
Apache 2.4.18, MariaDB 10.1.13, PHP 7.0.8, phpMyAdmin 4.5.1, OpenSSL 1.0.2, XAMPP Control Panel 3.2.2, Webalizer 2.23-04, Mercury Mail Transport System 4.63, FileZilla FTP Server 0.9.41, Tomcat 7.0.56 (with mod_proxy_ajp as connector), Strawberry Perl 7.0.56 Portable.

كما تلاحظ عزيزي الدارس، فإن الفرق الأساس في الإصدار الحالي من XAMPP والإصدار السابق هو أن الإصدار السابق يحوي قواعد البيانات باسم MySQL، والإصدار الحالي يحمل أسم قواعد البيانات MariaDB. في الحقيقة فإنه وبالنسبة للمستخدم لا فرق في الأوامر أو التعامل بينهما. إلا أن مبرمجي مجتمع أباتشي Apache Community، قد قرروا أن يصمموا MariaDB بسب مشاكل مع الشركة الأم أوراكل التي اشترت MySQL من شركة صن مسبقا، ولم تهتم بتطويره بسبب وجود قواعد بياناتها أصلا. لذلك تم تصميم MariaDB من قِبل مبرمجي مجتمع أباتشي بدون أي فروق، وبطريقة تحاكي نفس نمط MySQL.

بالضغط على الملف التنفيذي xampp-content.exe، المؤشر بالسهم الأحمر، يتم فتح لوحة التحكم والتي تسمح للمستخدم بتشغيل خادم الويب Apache وبالتالي تجزئة وتعريب parse كود لغة PHP، وكذلك تطبيق MariaDB أو MySQL. أنقر زر start المشار إليه بدائرة باللون الأخضر لتشغيل خادم الويب Apache وكذلك زر تشغيل قواعد البيانات MariaDB.



والناتج يجب أن يكون كما في الصورة التالية:



للتأكد من أن الخادم يعمل بصورة صحيحة، من خلال متصفح الانترنت (مثلا Google Chrome)، اطلب الموقع localhost، ثم اختار اللغة الانجليزية، فيجب أن تظهر معك الصفحة التالية:

ثالثا: البرنامج الأول

لكتابة البرنامج الأول بلغة PHP، لا بد من فهم مبادئ هذه اللغة. إن تعليمات script كود مكتوب بلغة PHP، يمكن أن توضع في أي موضع في المستند، ويجب أن يبدأ السكريبت بالإشارة **<?php** وينتهي بالإشارة **?>**. وهذا من أجل المتصفح الذي سيفهم أن سكريبت الكود القادم هو من نوع PHP. الشكل العام لسكريبت PHP في المستند، يظهر كما يلي:

```
<?php
// PHP code goes here
?>
```

يحتوي ملف PHP عادة على إشارات HTML وتعليمات كود PHP، كما يظهر في المثال التالي الذي يستخدم دوال داخلية جاهزة built-in، واسم الأولى echo ليخرج نص hello world في صفحة الويب، واسم الثانية print ولها نفس عمل الأولى. يمكن أن يتم إدخال إشارات HTML لإخراجها في صفحة الويب للمستخدم أيضا مثل سطر جديد وطباعة الجملة Hello Everybody بخط غامق، وإدخال سطر جديد، الخ.

ملف ex0.php

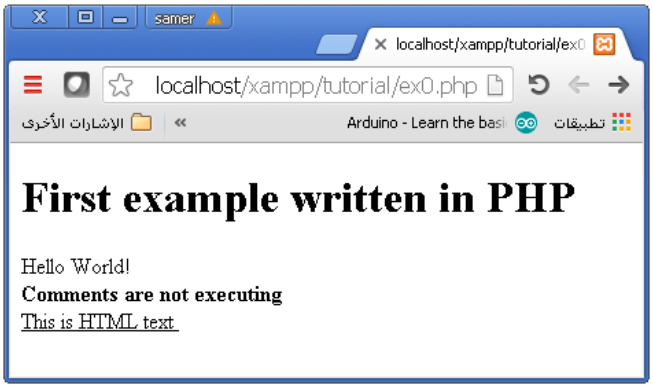
```
<!DOCTYPE html>
<html> <body> <h1>First example written in PHP</h1>

<?php
        echo "Hello World!";
?>

<?php
        print "<br> <b> Comments are not executing </b>";
        # This is a single-line comment
?>

<br> <u> This is HTML text  </u> <br>
</body> </html>
```

خزّن هذا الملف على مجلد جديد باسم tutorial داخل مجلد xampp وذلك على مجلد htdocs كما يلي:

C:\xampp\htdocs\xampp\tutorial

ملاحظة: عزيزي الدارس، إن لم يكن التطبيق xampp مفتوح، افتحه ثم شغل خادم الويب Apache.

الآن ومن خلال المتصفح نفذ الملف ex0.php، وتأكد من أن الناتج هو كما يلي:



يحتوي ملف PHP عادة على إشارات HTML وتعليمات كود PHP، كما يظهر في المثال التالي الذي يستخدم دوال داخلية جاهزة built-in، واسم الأولى echo ليخرج نص hello world في صفحة الويب، واسم الثانية print ولها نفس عمل الأولى. يمكن أن يتم إدخال إشارات HTML لإخراجها في صفحة الويب للمستخدم أيضا مثل سطر جديد وطباعة جملة بخط غامق، وإدخال سطر جديد، الخ . ففي المثال السابق، وكما تلاحظ، تنتهي جمل PHP بفاصلة منقوطة semicolon، ويمكن دمج إشارات HTML، داخل كود PHP. حيث تم إظهار النص Comments are not executing بخط غامق وتم أيضا إظهار الجملة في سطر جديد.

يستعمل التعليق في البرمجة من أجل توضيح تعليمات معينة أو جمل معينة داخل الكود، وهذه التعليقات لا تنفذ طبعا. في لغة PHP، التعليقات يمكن أن تتم من خلال الشرطتين "//" و السلم "#" واللتان تسمحان بكتابة سطر واحد من التعليقات، أو من خلال الشرطة مع النجمة والتي تسمح بكتابة أكثر من سطر من التعليقات كما في لغة C، ولغة C++

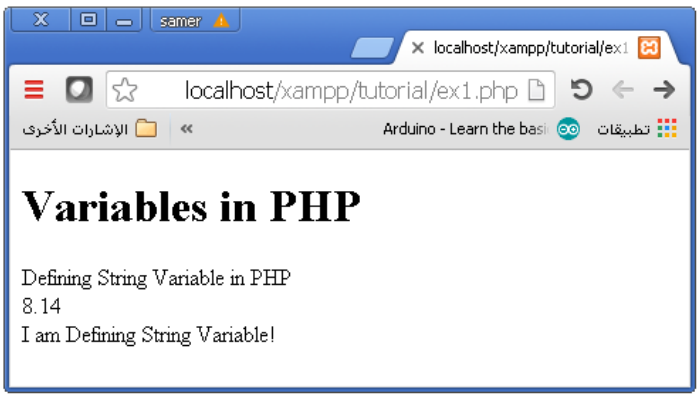
رابعا: المتغيرات Variables

في لغة PHP، فإن جميع الكلمات المحجوزة keywords، والأصناف والدوال هي not case-sensitive، أي أن أسم الدالة Print أو PrinT أو PRINT، هي جميعها تستدعي الدالة print السابقة. أما المتغيرات فهي case-sensitive. فمثلا المتغير x والذي يحمل نوع البيانات صحيح integer، يختلف عن المتغير X والذي يحمل نوع البيانات نقطة عائمة float.

ملف ex1.php

```
<!DOCTYPE html>
<html><body> <h1>Variables in PHP</h1>

<?php
        $str = "Defining String Variable";
        $x = 5;
        $X = 3.14;
        prinT "$str in PHP<br>";
        ecHo $x + $X;
        echo "<br>";
        echo "I am " . $str . "!";
?>

</body> </html>
```

الآن ومن خلال المتصفح نفذ الملف ex1.php، وتأكد من أن الناتج هو كما يلي:



كما تلاحظ عزيزي الدارس، يمكن تعريف متغير في لغة PHP باستخدام إشارة $ متبوعة باسم المتغير والذي يجب أن يبدأ بحرف أو underscore. ويمكن لصق أكثر من سلسلة باستخدام العامل (.) كما في التعليمة الأخيرة من المثال السابق. كذلك فإننا لم نخبر

لغة PHP بنوع البيانات الذي يحمله المتغير، وذلك لأن لغة PHP تقوم بتحويل المتغير بشكل أوتوماتيكي إلى نوع البيانات المناسب بناء على قيمته. فمثلا الكود التالي يمكن استخدامه لمعرفة نوع البيانات الذي يحمله المتغير من خلال الدالة الجاهزة var_dump:

ملف ex2.php

```
<!DOCTYPE html>
<html><body>
<?php
	$x = 379.999;
	var_dump($x);
	echo "<br>";
	$x = 29;
	var_dump($x);
?>
</body></html>
```

وتكون النتيجة كما يلي:

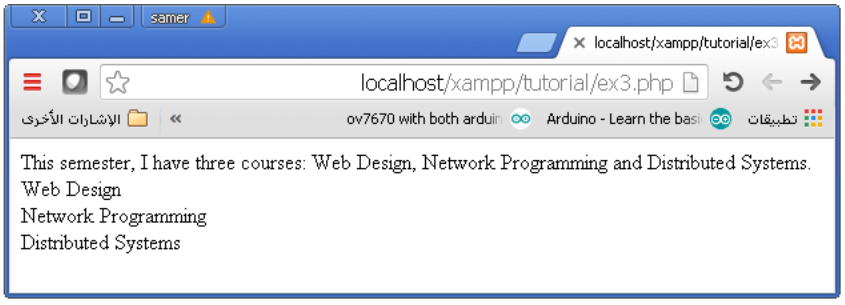
```
float(379.999)
int(29)
```

لتعريف متغير من نوع مصفوفة، نستخدم الكلمة المحجوزة array، والتي تمكننا من تعريف مجموعة من العناصر تحمل نفس نوع البيانات. المثال التالي يوضح تعريف مصفوفة تحمل ثلاثة عناصر من نوع سلسلة string، حيث يتم طباعة عناصر المصفوفة ووصل تلك العناصر بعضها ببعض بالإضافة إلى سلاسل أخرى من خلال العامل (.) كما تم توضيحه سابقا.

ملف ex3.php

```
<!DOCTYPE html>
<html><body>
<?php
	$courses = array("Web Design", "Network Programming", "Distributed Systems");
	echo "This semester, I have three courses:  " . $courses[0] . ", " . $courses[1] . " and " . $courses[2] . ".";
	echo "<br>";
	$arrayLength = count($courses);
	for($i = 0; $i < $arrayLength; $i++)
	{
		echo $courses[$i];
		echo "<br>";
	}
?>
</body></html>
```

وتكون النتيجة كما يلي:



هناك نوع من المصفوفات وهي المصفوفات المتزاملة (من زميل) تسمى associative arrays، تحوي مجموعة من العناصر المتزاملة مع قيمها، أي أن لكل عنصر في المصفوفة زميل واحد من القيم المقابلة له. هناك طريقتين لتعريف المصفوفة المتزاملة، الأولى باستخدام الكلمة المفتاح array:

```
$price = array("BMW"=>"35000", "VW"=>"32000", "Opel"=>"24000");
```

والثنية باستخدام التعريف المنفرد لكل عنصر:

```
$price['BMW'] = "35000";
$price['VW'] = "32000";
$price['Opel'] = "24000";
```

الآن يمكن استخدام أحد العناصر وما يقابله من قيمة باستخدام السكريبت التالي:

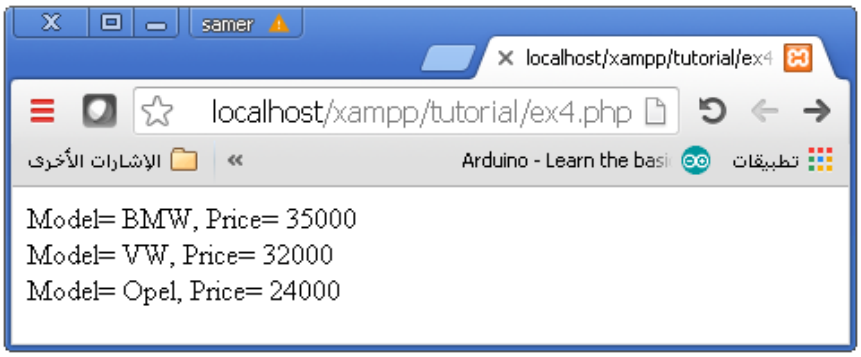
```
echo "Price of Opel is " . $price['Opel'] . " USD.";
```

أو من خلال جملة دوران، حيث يتم تعريف متغير مثل x يمثل العنصر، ومتغير آخر مثل x_value يمثل القيمة المقابلة (الزميلة)، كما هو موضح في المثال التالي:

ملف ex4.php

```
<!DOCTYPE html>
<html><body>
<?php
	$price = array("BMW"=>"35000", "VW"=>"32000", "Opel"=>"24000");
	foreach($price as $x => $x_value)
	{
		echo "Model= " . $x . ", Price= " . $x_value;
		echo "<br>";
	}
?>
</body></html>
```

وتكون النتيجة كما يلي:



خامسا: الدوال Functions

لقد تعرفنا على مجموعة من الدوال الجاهزة أو المبنية built-in في لغة PHP، ومنها print و echo. في هذا الجزء، سنتعرف على الدوال الرياضية المبنية في اللغة والتي تختص بالرياضيات، وكذلك على طريقة إنشاء واستدعاء الدوال من قبل المستخدم. المثال التالي يوضح طريقة استخدام بعض دوال الرياضيات. فيمكن إيجاد القيمة المطلقة لعدد معين باستخدام الدالة abs، وأسس الأعداد الصحيحة باستخدام الدالة pow، والدالة الأسية باستخدام exp، ولوغاريتم عدد باستخدام الدالة log10، والتقريب باستخدام round، وأقل عدد صحيح لعدد كسري باستخدام floor، وأكبر عدد صحيح لعدد كسرية باستخدام ceil. أما جيب الزاوية sin وجيب تمامها cos، فيأخذان زاوية بال radians ويعيدان قيمتها على شكل قيمة نقطة عائمة floating point value.

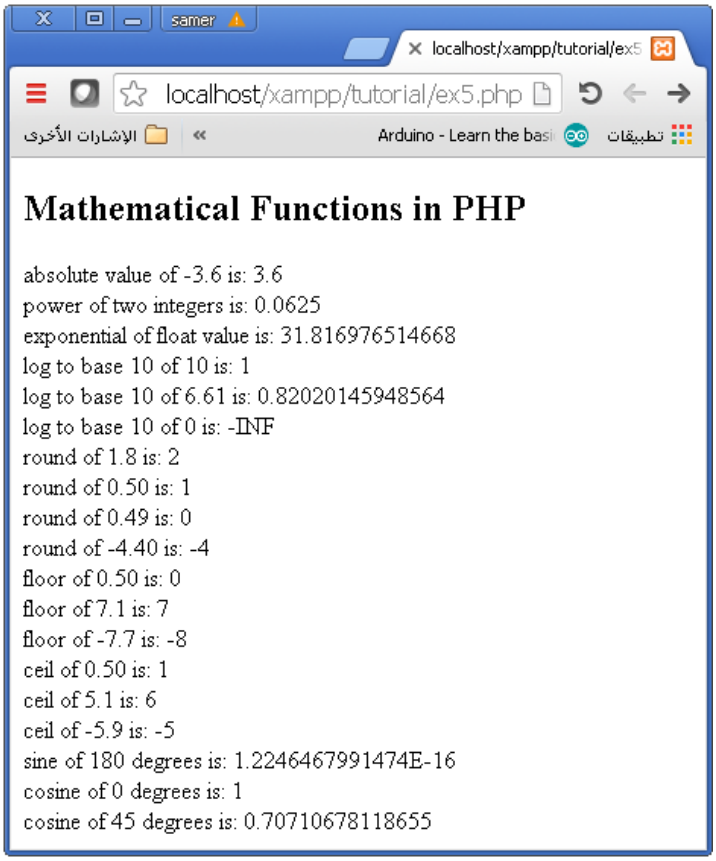
ملف ex5.php

بمساعدة مشرفك الأكاديمي، نفذ المثال التالي على جهاز الحاسوب، وحاول أن تتعرف على آلية عمل الدوال الرياضية.

```
<!DOCTYPE html>
<html><body><h2>Mathematical Functions in PHP</h2>
```

```
<?php
	echo "absolute value of -3.6 is: ";
	echo(abs(-3.6) ."<br>");
	echo "power of two integers is: ";
	echo(pow(2,-4) . "<br>");
	echo "exponential of float value is: ";
	echo(exp(3.46) . "<br>");
	echo "log to base 10 of 10 is: ";
	echo(log10(10) . "<br>");
	echo "log to base 10 of 6.61 is: ";
	echo(log10(6.61) . "<br>");
	echo "log to base 10 of 0 is: ";
	echo(log10(0) . "<br>");
	echo "round of 1.8 is: ";
	echo(round(1.80) ."<br>");
	echo "round of 0.50 is: ";
	echo(round(0.50) ."<br>");
	echo "round of 0.49 is: ";
	echo(round(0.49) ."<br>");
	echo "round of -4.40 is: ";
	echo(round(-4.40) ."<br>");
	echo "floor of 0.50 is: ";
	echo(floor(0.50) . "<br>");
	echo "floor of 7.1 is: ";
	echo(floor(7.1) . "<br>");
	echo "floor of -7.7 is: ";
	echo(floor(-7.7) . "<br>");
	echo "ceil of 0.50 is: ";
	echo(ceil(0.50) . "<br>");
	echo "ceil of 5.1 is: ";
	echo(ceil(5.1) . "<br>");
	echo "ceil of -5.9 is: ";
	echo(ceil(-5.9) . "<br>");
	echo "sine of 180 degrees is: ";
	echo(sin(M_PI) . "<br>");
	echo "cosine of 0 degrees is: ";
	echo(cos(0) . "<br>");
	echo "cosine of 45 degrees is: ";
	echo(cos(M_PI/4.0) . "<br>");
?>

</body></html>
```

ويكون ناتج التنفيذ كما يلي:



بجانب احتواء لغة PHP على آلاف الدوال المبنية مسبقا built-in، يمكن للمبرمج أن يعرف الدالة الخاصة به لتقوم ببعض المهمات. فالدالة في لغات البرمجة بشكل عام هي بلوك يحوي مجموعة من الجمل البرمجية، وهذا البلوك يتم استدعاؤه للتنفيذ عدة مرات أثناء عمل البرنامج وذلك بحسب الحاجة. فبدلا من كتابة نفس الكود عدة مرات، يتم وضع ذلك الكود في داخل بلوك يسمى function، ويتم استدعاؤه عند الحاجة للتنفيذ. تعليمات كود الدالة لا يتم تنفيذها عند تحميل الصفحة، ولكن عندما يتم استدعاء الدالة. في الشكل العام التالي لتعريف الدالة، وكما ترى، عزيزي الدارس، فإن التصريح بالدالة يتم من خلال الكلمة المفتاحية function. اسم الدالة يبدأ بحرف أو (_)، ولكن لا يمكن أن يبدأ برقم.

```
function functionName()
{
   //code to be executed;
}
```

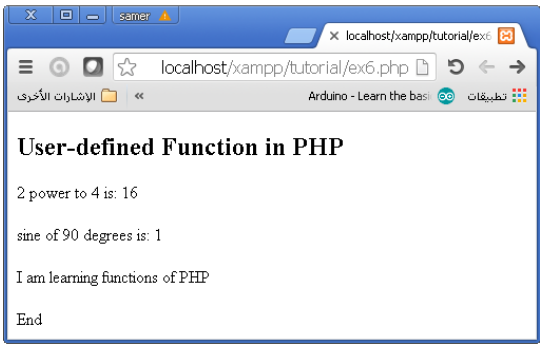
في المثال التالي، ستكتب دالة، تعمل على طباعة جملة في المتصفح.

ملف ex6.php

```
<!DOCTYPE html>
<html><body><h2>User-defined  Function in PHP</h2>

<?php
        echo “2 power to 4 is: “;
        echo(pow(2, 4) . “<br>”);
        function writeMessageToBrowser()
        {
                echo “<br>I am learning functions of PHP <br>”;
        }
        echo “<br>sine of 90 degrees is: “;
        echo(sin(M_PI/2.0) . “<br>”);
        writeMessageToBrowser(); // call the function
```

```
	echo “<br>End”;
?>

</body></html>
```

وتكون النتيجة كالتالي:

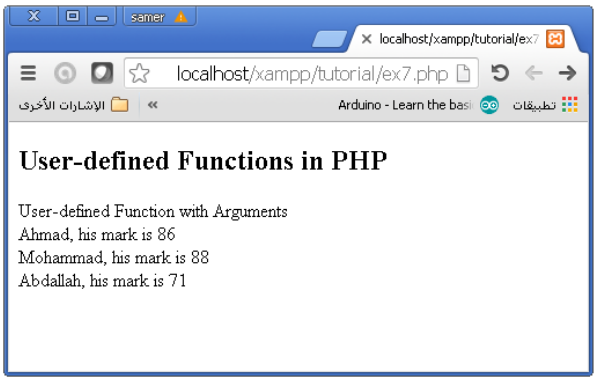


وكما ترى عزيزي الدارس، يجب استدعاء الدالة من أجل التنفيذ. فقد تم تنفيذ تعليمات script ناتج 2 للقوة 4، ولكن لم يتم تنفيذ تعليمات الدالة، فقفز عن الأسطر الخاصة بها، حيث قام بتنفيذ تعليمات إيجاد جيب زاوية 90 درجة، بعد ذلك صادف وجود تعليمة تستدعي الدالة، فقام بتنفيذ تعليمات الدالة، ثم عاد ينفذ أخر تعليمة في البرنامج وهي طباعة الكلمة End.

ملف ex7.php

في المثال التالي، عزيزي الدارس، سوف تكتب دالة، تأخذ متغيرين two parameters، المتغير الأول أسم العائلة familyName، والمتغير الثاني العلامة Mark، حيث تعمل هذه الدالة على كتابة معلومات عن الطالب في المتصفح.

```
<!DOCTYPE html>
<html><body><h2>User-defined  Functions in PHP</h2>

<?php
	echo "User-defined Function with Arguments<br>";
	function studentInformation($FamilyName, $Mark)
	{
		echo "$FamilyName, his mark is $Mark <br>";
	}

	studentInformation("Ahmad","86");
	studentInformation("Mohammad","88");
	studentInformation("Abdallah","71");
?>

</body></html>
```

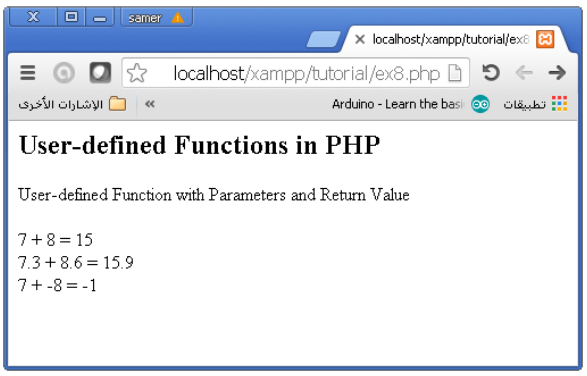


لاحظ عزيزي الدارس، أنه تم مناداة الدالة باسمها مع تمرير معاملين two arguments، في كل مرة. وهما اسم العائلة والعلامة.

ملف ex8.php

في المثال التالي، عزيزي الدارس، سوف تكتب دالة، تأخذ متغيرين two parameters، من نوع integers، وتعيد الناتج إلى المنادي ليعمل على كتابة حاصل جمع العددين في المتصفح.

```
<!DOCTYPE html>
<html><body><h2>User-defined  Functions in PHP</h2>

<?php
	echo "User-defined Function with Parameters and Return Value<br><br>";
	function Addition($number1, $number2)
	{
		$result = $number1 + $number2;
		return $result;
	}

	echo "7 + 8 = " . Addition (7, 8)."<br>";
	echo "7.3 + 8.6 = " . Addition (7.3, 8.6)."<br>";
	echo "7 + -8 = " . Addition (7, -8)."<br>";
?>

</body></html>
```



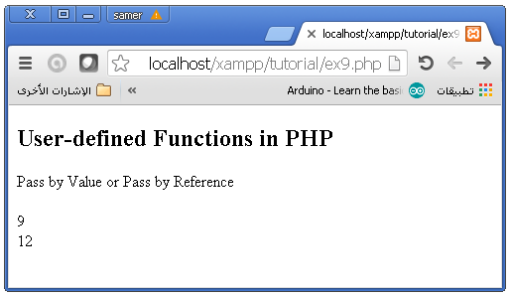
لاحظ عزيزي الدارس، أنه تم مناداة الدالة باسمها مع تمرير معاملين two arguments، ثلاث مرات. في المرة الأولى وفي المرة الثالثة، كان المعملان عددين صحيحين، وفي المرة الثانية كانا بفاصلة عشرية floating point numbers.

ملف ex9.php

في المثال التالي، عزيزي الدارس، سوف تكتب دالتين، الأولى تأخذ متغير واحد one parameter، والثانية أيضا. الفرق بينهما، هو أن نوع التمرير، تمرير بالقيمة وتمرير بالمرجع call by value or call by reference.

```
<!DOCTYPE html>
<html><body><h2>User-defined  Functions in PHP</h2>

<?php
	echo "Pass by Value or Pass by Reference<br><br>";
	function add7($number1)
	{
		$number1 = $number1 + 7;
	}
	function add3(&$number2)
	{
		$number2 = $number2 + 3;
	}
```

```
        $result = 9;
        Add7($result);
        echo "$result<br>";
        Add3($result);
        echo "$result<br>";
?>

</body></html>
```



لاحظ عزيزي الدارس، أنه تم مناداة كل دالة باسمها مع تمرير معامل لها. ولكن في المرة الأولى، كانت المناداة بتمرير القيمة، فلم تتغير القيمة الأصلية، وفي المرة الثانية كانت المناداة بتمرير المرجع، فحصل التغيير على الرقم الأساس.

سادسا: مجال المتغيرات Scope of Variables

إن مجال المتغيرات scope of variables، هو الجزء من التعليمات الذي يمكنه رؤية ذلك المتغير، ويمكن استعمال ذلك المتغير من قبل تلك التعليمات. ولغة PHP تعرف 3 مجالات وهي المحلي local، والعام global، والثابت static. فالمتغير الذي يعرف خارج الدالة له مجال رؤية من نوع Global، ولا يمكن استخدامه من داخل الدالة إلا إذا تم التصريح بعموميته من داخل تلك الدالة. بالتالي، فإن تنفيذ البرنامج التالي يؤدي إلى خطأ بسبب محاولة استخدام متغير في داخل دالة، مع العلم أن تم تعريفه عام global.

```
<!DOCTYPE html>
<html><body><h2>Global variable in PHP</h2>
<?php
        $x = 3; // global scope
        function func1()
        {
                echo "<p>Error using variable x inside function: $x</p>";
        }
        func1();
        echo "<p>No Error: $x</p>";
?>
</body></html>
```

متغير تم تعريفه داخل الدالة، له مجال محلي local scope، ويمكن استخدامه في داخل تلك الدالة فقط. لذلك فإن تنفيذ سكريبت PHP التالي، سيؤدي إلى خطأ بسبب أن متغير تم تعريفه داخل دالة، ولكن المبرمج يعمل على محاولة استخدامه من خارجها.

```
<!DOCTYPE html>
<html><body><h2>Local variables in PHP</h2>
<?php
        function func2()
        {
                $x = 3; // local scope
                echo "<p>No Error printing variable x inside the function: $x</p>";
        }
        func2();
        echo "<p>Error accessing variable x outside the function: $x</p>";
?>
</body></html>
```

متغير تم تعريفه خارج الدالة، يمكن استخدامه في داخل الدالة، ولكن يجب أن يوصف بأنه عام من خلال الكلمة المفتاحية global، وذلك في داخل الدالة التي ستستخدمه. لا ينتج أية أخطاء عند تنفيذ السكريبت التالي:

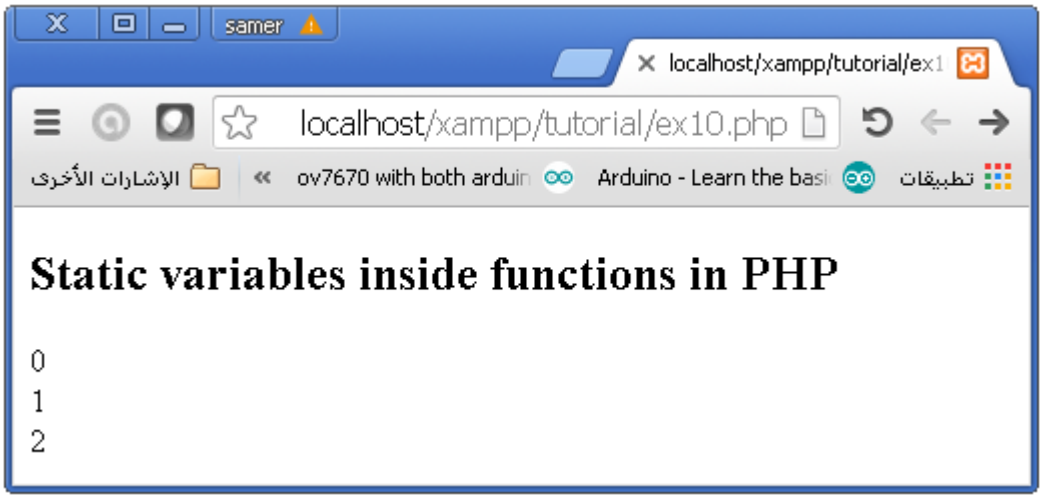
```
<!DOCTYPE html>
<html><body><h2>Global variables inside functions in PHP</h2>
<?php
	$x = 5; $y = 7; $z = 0;
	function func3()
	{
		global $x, $y, $z;
		$z = $x + $y;
	}
	func3();
	echo $z; // result is 12
?>
</body></html>
```

عند الانتهاء من تنفيذ دالة قمنا بكتابتها، فإن جميع متغيراتها المحلية تحذف. أحيانا، نريد الاحتفاظ بآخر قيمة لمتغير محلي معين، ولا نريد أن يتم حذفه، لأننا نريد استخدامه لاحقا. يتم التصريح بهذا المتغير على أنه static عند تعريفه، كما يلي:

ملف ex10:

```
<!DOCTYPE html>
<html><body><h2>Static variables inside functions in PHP</h2>
<?php
	function func4()
	{
		static $x = 0;
		echo $x;
		echo "<br>";
		$x++;
	}
	func4 ();
	func4 ();
	func4 ();
?>
</body></html>
```

ويكون ناتج تنفيذ البرنامج كما يلي:



فكما تلاحظ، تم الاحتفاظ بالقيمة القديمة ثم أضاف القيمة الجديدة إليها، فكانت الزيادة تظهر في كل مرة. نفذ عزيزي الدارس، المثال السابق من غير الكلمة المفتاحية static، ثم لاحظ الفرق.

النشاط الصفي الرابع

أساسيات قواعد البيانات باستخدام MySQL

مقدمة

قاعدة البيانات database، هي عبارة عن تجميع منظم من البيانات، والتي قد تكون على شكل جداول tables، أو استعلامات queries، أو تقارير reports، أو حتى مخططات schemas. يتم تنظيم البيانات بحيث تعمل على نمذجة بعض التطبيقات الحية مثل المساعدة في حجز غرفة فارغة في فندق. فقد تشكل قواعد البيانات جزءاً أساسياً من نظام اتصالات، أو نظام بنكي، أو ألعاب حاسوب، أو حتى أية جهاز إلكتروني. إن نظام إدارة قواعد البيانات database management system (DBMS)، هو عبارة عن تطبيق برمجي software حاسوبي، والذي يتعامل مع مستخدم أو تطبيقات أو نفس قاعدة البيانات من أجل جلب وتحليل البيانات. يتم تصميم نظام قواعد بيانات عام المهمات (general-purpose DBMS)، من أجل القيام بمهمات عامة مثل تعريف قاعدة بيانات، إنشاء قاعدة بيانات، الاستعلام عن محتويات قاعدة بيانات، تحديث قاعدة بيانات، بالإضافة إلى إدارة قواعد البيانات. بعض الأمثلة على نظم قواعد البيانات MySQL, PostgreSQL, Oracle, MS SQL Server, Sybase, SAP HANA, IBM DB2.

ليس بالضرورة أن تعمل قاعدة بيانات معينة على جميع نظم إدارة قواعد البيانات DBMS، ولكن يمكن استخدام معايير عالمية للتواصل مع تلك النظم وفيما بينها أيضا مثل معايير SQL, JDBC, ODBC. بالتالي نسمح لأية تطبيق أن يتواصل مع قاعدة البيانات من غير معرفة الكثير عن تصميمها أو نوعها. من أكثر أنواع البيانات شهرة قواعد البيانات العلائقية Relational DBMS أو (RDBMS)، والذي يعتبر MySQL أشهرها خصوصا في تطبيقات الويب والانترنت.

الأهداف

1. أن تتعرف على بيئة قواعد بيانات MySQL، وطريقة تشغيلها والعمل عليها.
2. أن تعمل على إنشاء قاعدة بيانات خاصة بالزبائن في بيئة MySQL، وتنشئ جدول يحوي بياناتهم الشخصية وآخر يحوي طلبياتهم.
3. أن تتعرف على طريقة استعراض كافة محتويات الجدول أو أن تختار جزء منها.
4. أن تتعرف على طريقة الربط بين الجدولين في نفس قاعدة البيانات، بحيث تربط كل زبون في الجدول الأول بطلبياته في الجدول الثاني.

خطوات عمل الجزء الأول

سيتم في هذا الجزء التعرف على طريقة الدخول إلى قاعدة البيانات. شغّل، عزيزي الدارس، شاشة DOS، ثم أدخل إلى مجلد bin الموجود في مجلد mysql، والموجود في مجلد xampp عن طريق أمر DOS وهو CD. من هنا يمكن إطلاق عمل MySQL من خلال الأمر mysql –u root –p، والذي يحدد اسم المستخدم "root"، وكلمة المرور. عندما يطلب كلمة المرور، لا نكتب شيء لعدم وجود كلمة مرور أصلا على المستخدم root. الآن يمكنك البدء بإعطاء أو كتابة أوامر sql في نافذة mysql.

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\exam>cd C:\xampp\mysql\bin

C:\xampp\mysql\bin>mysql -u root -p
Enter password:
mysql: Unknown OS character set 'cp720'.
mysql: Switching to the default character set 'latin1'.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.5.25a MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

نريد إنشاء قاعدة بيانات للزبائن بأسم customers. يم ذلك من خلال الأمر ;CREATE DATABASE customers. وتكون النتيجة إيجاباً إن تم إنشاء قاعدة البيانات. عندها يمكن البدء باستخدامها من خلال الأمر ;USE customers. فيمكن إنشاء جدول بأسم Persons، ويحتوي على خمسة أعمدة columns كما يلي:

1. العمود الأول معرِّف identifier للشخص person بعنوان p_id، من نوع عدد صحيح int، ويجب أن لا تكون قيمته فراغ، وهو مفتاح أساسي PRIMARY KEY.
2. العمود الثاني أسم العائلة أو الأسم الأخير للشخص بعنوان LastName، من نوع حروف متغيرة varchar، ويجب أن لا تكون قيمته فراغ، وبحدّ أقصى 255 حرف.
3. العمود الثالث الأسم الأول للشخص بعنوان FirstName، من نوع حروف متغيرة varchar، وبحدّ أقصى 255 حرف.
4. العمود الرابع للعنوان البريدي Address، من نوع حروف متغيرة varchar، وبحدّ أقصى 255 حرف.
5. العمود الخامس للمدينة City، من نوع حروف متغيرة varchar، وبحدّ أقصى 255 حرف.

```
mysql> CREATE DATABASE customers
    -> ;
Query OK, 1 row affected (0.00 sec)

mysql> USE customers;
Database changed
mysql> CREATE TABLE persons
    -> (
    -> p_id int NOT NULL,
    -> LastName varchar(255) NOT NULL,
    -> FirstName varchar(255),
    -> Address varchar(255),
    -> City varchar(255),
    -> PRIMARY KEY (P_ID)
    -> );
Query OK, 0 rows affected (0.06 sec)
```

الآن يمكن إدخال القيم إلى الجدول باستخدام الصفوف rows، صفّ يليه صفّ. كل صف يحمل قيم الأعمدة السالفة الذكر التي تم إنشاؤها في الخطوة السابقة. هذا يتم من خلال الأمر INSERT INTO Persons، والذي يعمل على إدخال قيم للأعمدة بنفس ترتيب الأعمدة، حيث تأخذ القيم VALUES بالشروط المحددة مسبقا عند إنشاء الجدول. يتم تكرار نفس الأمر لكل زبون، ولكن باختلاف القيم طبعا.

```
mysql> INSERT INTO Persons (p_id, LastName, FirstName, Address, City)
    -> VALUES (0, 'Ahmad', 'Ayman', 'Rafedia 10', 'Nablus');
Query OK, 1 row affected (0.03 sec)

mysql> INSERT INTO Persons (p_id, LastName, FirstName, Address, City)
    -> VALUES (1, 'Mohamed', 'Wael', 'Qabatia 18', 'Jenin');
Query OK, 1 row affected (0.08 sec)

mysql> INSERT INTO Persons (p_id, LastName, FirstName, Address, City)
    -> VALUES (2, 'Ali', 'Radi', 'Betonya 167', 'Ramallah');
Query OK, 1 row affected (0.03 sec)
```

من أجل استعراض محتويات الجدول، نستخدم أمر SQL التالي: SELECT * FROM Persons;، فعلامة النجمة تشير إلى جميع محتويات الجدول.

```
mysql> SELECT * FROM Persons;
+------+----------+-----------+-------------+----------+
| p_id | LastName | FirstName | Address     | City     |
+------+----------+-----------+-------------+----------+
|    0 | Ahmad    | Ayman     | Rafedia 10  | Nablus   |
|    1 | Mohamed  | Wael      | Qabatia 18  | Jenin    |
|    2 | Ali      | Radi      | Betonya 167 | Ramallah |
+------+----------+-----------+-------------+----------+
3 rows in set (0.00 sec)
```

لاستعراض جزء من محتويات الجدول وليس جميع المحتويات، نختار محتويات الأعمدة التي نريد عرضها لكل زبون. فمثلا إن أردنا أن نعرف معرف الزبون p_id، واسمه الأخير فقط، نستخدم الأمر SELECT p_id, LastName FROM Persons;.

```
mysql> SELECT p_id, LastName FROM Persons;
+------+----------+
| p_id | LastName |
+------+----------+
|    0 | Ahmad    |
|    1 | Mohamed  |
|    2 | Ali      |
+------+----------+
3 rows in set (0.00 sec)
```

كما يمكننا أن ضع شرط على الاختيارات، فمثلا لاستعراض كل البيانات التي تتعلق بشخص معين، يمكن استخدام الأمر التالي:

```
mysql> SELECT * FROM Persons WHERE p_id=1;
+------+----------+-----------+------------+-------+
| p_id | LastName | FirstName | Address    | City  |
+------+----------+-----------+------------+-------+
|    1 | Mohamed  | Wael      | Qabatia 18 | Jenin |
+------+----------+-----------+------------+-------+
1 row in set (0.01 sec)
```

خطوات عمل الجزء الثاني

في هذا الجزء، يتطلب منك عزيزي الدارس إنشاء جدول آخر لطلبات الزبائن بأسم Orders. سيتم ربط هذا الجدول بالجدول الأول، والذي يتعلق بالبيانات الشخصية للزبون، من خلال المفتاح الأجنبي FOREIGN KEY. سيحتوي الجدول الجديد على معرف طلبية بأسم o_id، ورقم طلبية OrderNo، بالإضافة طبعا إلى رقم الزبون صاحب الطلبية. هذا الحقل سيستخدم كوسيط للربط ما بين الجدولين باستخدام أمر SQL التالي: FOREIGN KEY (p_id) REFERENCES Persons (p_id). كما ترى، عزيزي الدارس، فإن المرجع reference هو معرِّف الزبون في جدول Persons.

```
mysql> CREATE TABLE Orders
    -> (o_id int NOT NULL,
    -> OrderNo int NOT NULL,
    -> p_id int,
    -> PRIMARY KEY (o_id),
    -> FOREIGN KEY (p_id) REFERENCES Persons (p_id)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

بنفس الطريقة السابقة، أدخل القيم التالية إلى الجدول الجديد.

```
mysql> INSERT INTO Orders (o_id, OrderNo, p_id)
    -> VALUES (0, 22222, 1);
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO Orders (o_id, OrderNo, p_id)
    -> VALUES (1, 33333, 1);
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO Orders (o_id, OrderNo, p_id)
    -> VALUES (2, 101010, 0);
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO Orders (o_id, OrderNo, p_id)
    -> VALUES (3, 151516, 0);
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO Orders (o_id, OrderNo, p_id)
    -> VALUES (4, 445522, 1);
Query OK, 1 row affected (0.02 sec)
```

للتأكد من أن البيانات المدخلة صحيحة، استعرض كافة محتويات الجدول:

```
mysql> SELECT * FROM Orders;
+------+---------+------+
| o_id | OrderNo | p_id |
+------+---------+------+
|    0 |   22222 |    1 |
|    1 |   33333 |    1 |
|    2 |  101010 |    0 |
|    3 |  151516 |    0 |
|    4 |  445522 |    1 |
+------+---------+------+
5 rows in set (0.00 sec)
```

يمكنك أيضا استعراض بعض محتويات الجدول، مثلا رقم الطلبة ورقم الزبون فقط، لجميع الزبائن، كما يلي:

```
mysql> SELECT OrderNo, p_id FROM Orders;
+---------+------+
| OrderNo | p_id |
+---------+------+
|   22222 |    1 |
|   33333 |    1 |
|  101010 |    0 |
|  151516 |    0 |
|  445522 |    1 |
+---------+------+
5 rows in set (0.01 sec)
```

أو أن تستعرض جميع الطلبات التي تتعلق بزبون معين من خلال رقم التعريف (المعرِّف):

```
mysql> SELECT * FROM Orders WHERE p_id=1;
+------+---------+------+
| o_id | OrderNo | p_id |
+------+---------+------+
|    0 |   22222 |    1 |
|    1 |   33333 |    1 |
|    4 |  445522 |    1 |
+------+---------+------+
3 rows in set (0.00 sec)
```

كما يمكنك استعراض بعض محتويات الجدولين في نفس الوقت. تذكر عزيزي الدارس، أننا قمنا بربط الجدولين من خلال المفتاح الأجنبي FOREIGN KEY. فيمكن أن نستعرض أسم العائلة لزبون من جدول Persons، ورقم الطلبية من الجدول Orders، بشرط أن يتساوى الرقم المعرف (p_id) في كلا الجدولين. لاستعراض طلبيات زبون معين نشترط أن يكون الرقم المعرف p_id، صفر.

```
mysql> SELECT Persons.LastName, Orders.OrderNo FROM Persons, Orders WHERE
    -> Persons.p_id=Orders.p_id AND Orders.p_id=0;
+----------+---------+
| LastName | OrderNo |
+----------+---------+
| Ahmad    |  101010 |
| Ahmad    |  151516 |
+----------+---------+
2 rows in set (0.00 sec)
```

بعد الانتهاء من العمل على MySQL، يمكنك الإنهاء باستخدام الأمر quit، كما يلي:

```
mysql> quit
Bye

C:\xampp\mysql\bin>
```

النشاط الصفي الخامس

ربط قواعد البيانات بلغة برمجة الويب

PHP and MySQL

مقدمة

عزيزي الدارس، لا يكاد يخلو موقع ويب مختص بالتسوق online shopping من قواعد بيانات، وكثير من مواقع الشركات الكبرى، تستخدم قواعد البيانات بشكل ملفت. ناهيك عن مواقع التواصل الاجتماعي التي تعتمد بشكل كلي تقريبا على قواعد البيانات. ولكن سرّ قوة قواعد البيانات يكمن في إنشاء واجهة رسومات تخاطبية GUI مع المستخدم ليسهل التعامل ما بين المستخدم وقواعد البيانات. وهنا يأتي دور لغة البرمجة في تقديم واجهة التخاطب GUI تسهيل التعامل مع محرك قواعد البيانات Database Engine. في هذه التجربة، ستعمل بمساعدة مشرفك الأكاديمي وفني المختبر، على إنشاء واجهت تخاطب GUI مع المستخدم في المتصفح Browser، من خلال لغة برمجة الويب PHP، بحيث يكون المستخدم قادرا على إضافة بيانات إلى قاعدة البيانات أو حذف بيانات من تلك القاعدة.

الأهداف

1. أن تتعرف على طريقة ربط لغة البرمجة PHP مع قواعد البيانات MySQL.
2. أن تتعرف على الدوال الجاهزة في لغة PHP، والتي تختص بمعالجة قواعد بيانات MySQL.
3. أن تتعرف على طريقةجلب fetch بيانات معينة بواسطة لغة PHP من قاعدة بيانات تم إنشائها مسبقا في MySQL.
4. أن تعمل على إدخال بيانات إلى قاعدة بيانات تم إنشائها مسبقا من خلال صفحة PHP.

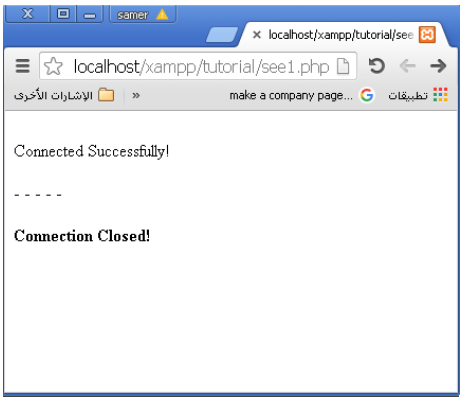
خطوات عمل الجزء الأول

هذا الجزء يحقق الهدفين الأول والثاني، وهما طريقة ربط لغة البرمجة PHP مع قواعد بيانات MySQL، والدوال التي تعالج هذا الأمر في لغة PHP. لنتذكر أن منفذ port الخاص بنظام إدارة قواعد بيانات MySQL، هو 3306 وبالتالي، فإننا نعرّف الجهاز المضيف في متغير dbhost بحيث يحمل القيمة "localhost:3306". أمل أسم المستخدم فهو root، وبدون كلمة المرور لأننا لم نحددها مسبقا. البرنامج التالي يربط مع قاعدة البيانات من غير أن ينفذ أية أمر من أوامر MySQL. يتم الربط من خلال استدعاء الدالة الجاهزة mysql_connect، والتي تأخذ ثلاثة متغيرات: أسم المضيف والمنفذ كمتغير أول، والمتغير الثاني الذي يحمل أسم المستخدم، والمتغير الثالث الذي يحمل كلمة المرور. في حال نجح الربط تعيد هذه الدالة true، وإلا فإنها تعيد false. نفحص إن لم ينجح الاتصال، نوقف العملية عن طريق الدالة الجاهزة die، حيث نستدعي الدالة التي تحمل الخطأ وهي mysql_error. عند الانتهاء من فحص الربط، نغلقه باستدعاء الدالة mysql_close()، ونمرر لها أسم الرابط.

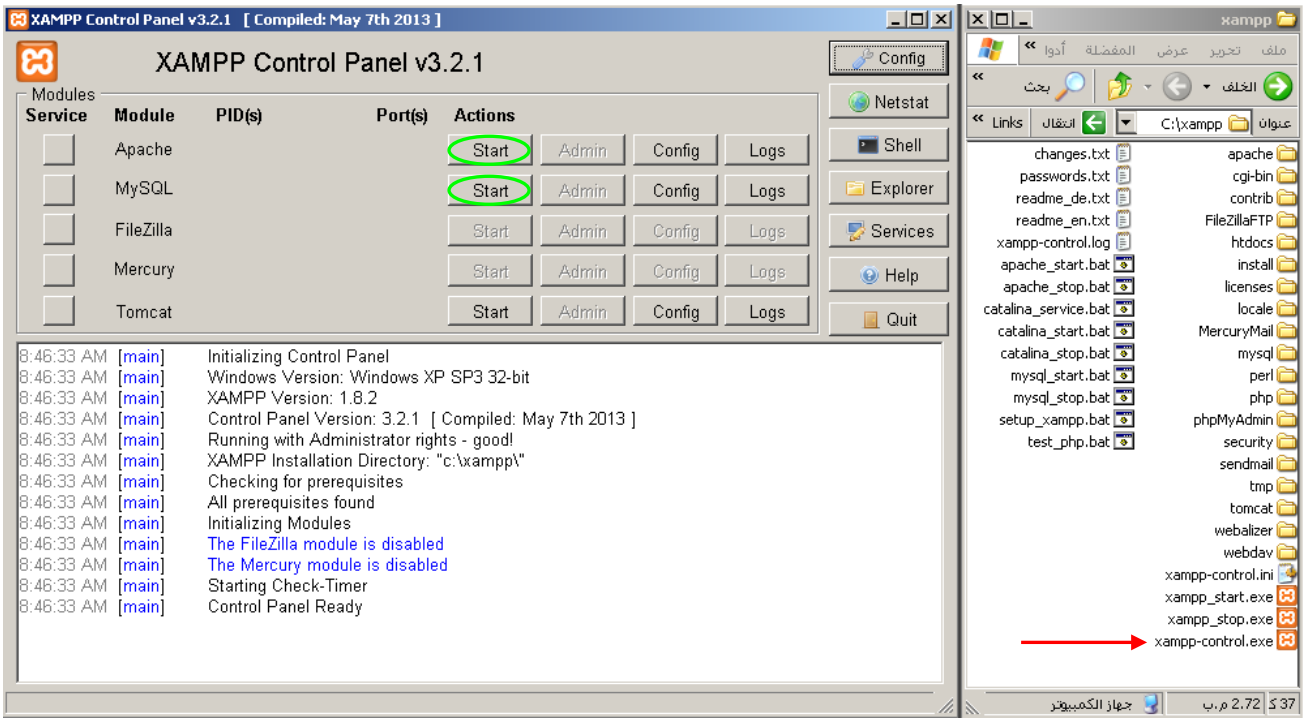
المثال الأول see1.php

```
<?php
  $dbhost = 'localhost:3306'; # check with the XAMPP control panel for the working port
  $dbuser = 'root';
  $dbpass = '';
  $conn = mysql_connect($dbhost, $dbuser, $dbpass);
  if(! $conn )
  {
    die( 'Could not connect: ' . mysql_error() );
  }
  echo '<br>Connected Successfully!<br>';
  echo '<br> - - - - - <br>';
  mysql_close($conn);
```

```
  echo '<br><b>Connection Closed!</b><br> ';
?>
```



ملاحظة: عزيزي الدارس، يجب أن يكون كل من MySQL و Apache في حالة التشغيل من أجل تنفيذ أمثلة هذه التجربة.



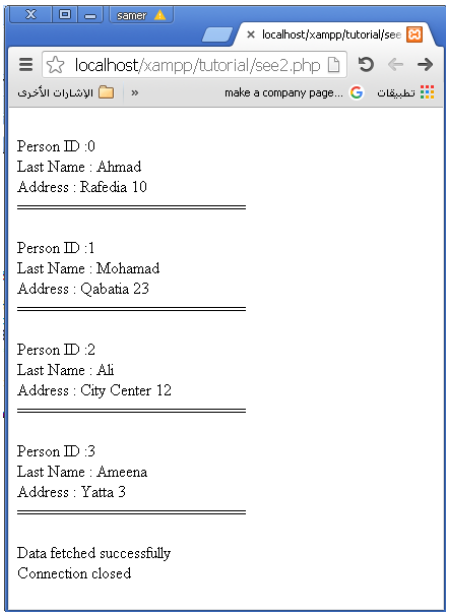
خطوات عمل الجزء الثاني

هذا الجزء يوضح جلب fetch بيانات معينة من قاعدة البيانات التي تم إنشائها مسبقا وهي customers، وبالتالي يحقق الهدف الثاني. نفس الطريقة التي استخدمتها للربط مع قواعد البيانات في المثال السابق، سوف تستخدمها في هذا المثال أيضا. ولكن الجديد في هذا المثال، هو انك ستعمل على تنفيذ جمل SQL لجلب fetch بيانات الزبائن من قاعدة البيانات customers. دوال PHP الجاهزة التي نحتاجها لهذا الهدف هي mysql_select_db() والتي تنفذ أمر sql الخاص باختيار قاعدة البيانات من أجل العمل عليها، والدالة mysql_query() والتي تنفذ أوامر sql الخاصة بالاستعلام، واخيرا الدالة mysql_fetch_array() والتي تعمل على جلب مكونات ومحتويات ناتج تنفيذ أمر الاستعلام.

المثال الثاني see2.php

```
<?php
        $dbhost = 'localhost:3306'; /* or use the default: $dbhost = 'localhost';*/
        $dbuser = 'root';
        $dbpass = '';
```

```
	$conn = mysql_connect($dbhost, $dbuser, $dbpass);
	if(! $conn )
	{
		die('Could not connect: ' . mysql_error());
	}
	$sql_statement = 'SELECT P_Id, LastName, Address FROM persons';
	mysql_select_db('customers');
	$retval = mysql_query( $sql_statement, $conn );
	if(! $retval )
	{
		die('Could not get data: ' . mysql_error());
	}
	while($row = mysql_fetch_array($retval, MYSQL_ASSOC))
	{
		echo "<br> Person ID :{$row['P_Id']} ";
		echo "<br> Last Name : {$row['LastName']} ";
		echo "<br> Address : {$row['Address']} ";
		echo "<br> ========================== <br> ";
	}
	echo "<br> Data fetched successfully\n";
	mysql_close($conn);
	echo "<br> Connection closed \n";
?>
```



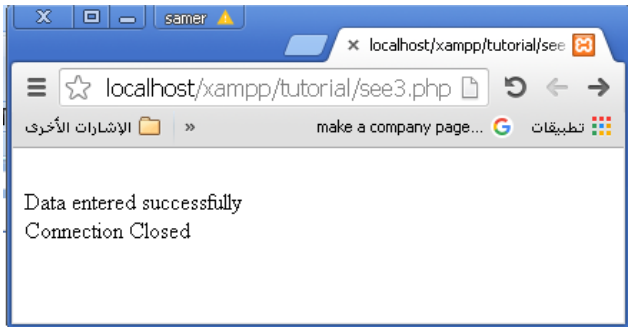
خطوات عمل الجزء الثالث

يحقق هذا الجزء، الهدف الثالث وهو إدخال بيانات إلى قاعدة بيانات تم إنشائها مسبقا من خلال صفحة PHP، وذلك من غير واجهة رسومات تخاطبية GUI. الفرق الوحيد عن المثال السابق هو في جملة sql التي ستنفذ. في هذا المثال، يتم إدخال بيانات جديدة إلى قاعدة البيانات ولا يتم جلب بيانات. بالتالي الدوال المستخدمة في المثال السابق، سوف تستخدم في هذا المثال أيضا.

المثال الثالث see3.php

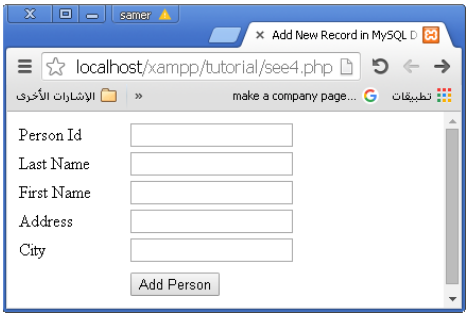
```
<?php
  $dbhost = 'localhost:3306';
  $dbuser = 'root';
  $dbpass = '';
  $conn = mysql_connect($dbhost, $dbuser, $dbpass);
  if(! $conn )
```

```
  {
    die('Could not connect: ' . mysql_error());
  }
  $sql_statement = 'INSERT INTO Persons (P_Id, LastName, FirstName, Address, City) VALUES (4, "Ameena",
"Mahmoud", "Yatta 3", "Hebron")';
  mysql_select_db('customers');
  $retval = mysql_query( $sql_statement, $conn );
  if(! $retval )
  {
    die('Could not enter data: ' . mysql_error());
  }
  echo "<br> Data entered successfully\n";
  mysql_close($conn);
  echo "<br> Connection Closed \n";
?>
```



خطوات عمل الجزء الرابع

في هذا الجزء سيتم إنشاء واجهة رسومات تخاطبية GUI مع المستخدم من خلال نموذج HTML Form، كما هو موضح في الصورة أدناه. يتم إنشاء الواجهة التخاطبية GUI باستخدام سكريبت كود HTML، حيث ننشئ حقل لرقم الشخص (الزبون)، وحقل لأسمه الأخير، وحقل لأسمه الأول وحقل للعنوان وآخر للمدينة. من اجل رفع upload المعلومات إلى خادم قاعدة البيانات، نضيف زر button باسم add وليكن معرفه id أيضا بنفس أسمه. الدالة الجديدة التي سنستخدمها من PHP هي isset وهذه الدالة تعمل على فحص إن كان الزر add قد تم ضغطه من قبل المستخدم فإنها تعيد true. عندها يبدأ العمل على فتح رابط مع قاعدة البيانات على الخادم وتنفيذ اوامر sql كما حصل في الأمثلة السابقة.



المثال الرابع see4.php

```
<html>
  <head>
    <title>Add New Record in MySQL Database</title>
  </head>
  <body>
    <?php
      if(isset($_POST['add']))
      {
        $dbhost = 'localhost:3306';
```

```
  $dbuser = 'root';
  $dbpass = '';
  $conn = mysql_connect($dbhost, $dbuser, $dbpass);
  if(! $conn )
  {
    die('Could not connect: ' . mysql_error());
  }
  $id1 = $_POST['id1'];
  $LN = $_POST['LN'];
  $FN = $_POST['FN'];
  $C  = $_POST['C'];
  $A  = $_POST['A'];
  $sql="INSERT INTO Persons(P_Id,LastName,FirstName,City,Address) VALUES('$id1','$LN','$FN','$C','$A')";
  mysql_select_db('customers');
  $retval = mysql_query( $sql, $conn );
  if(! $retval )
  {
    die('Could not enter data: ' . mysql_error());
  }
  echo "<br> Entered data successfully\n";
  mysql_close($conn);
  echo "<br> Connection closed\n";
}
else
{
  ?>
    <form method="post" action="<?php $_PHP_SELF ?>">
      <table width="400" border="0" cellspacing="1" cellpadding="2">
        <tr>
          <td width="100">Person Id</td>
          <td><input name="id1" type="text" id="id1"></td>
        </tr>
        <tr>
          <td width="100">Last Name</td>
          <td><input name="LN" type="text" id="LN"></td>
        </tr>
        <tr>
          <td width="100">First Name</td>
          <td><input name="FN" type="text" id="FN"></td>
        </tr>
        <tr>
            <td width="100">Address</td>
            <td><input name="A" type="text" id="C"></td>
        </tr>
        <tr>
            <td width="100">City</td>
            <td><input name="C" type="text" id="A"></td>
        </tr>
        <tr>
          <td width="100"> </td>
          <td> </td>
        </tr>
        <tr>
          <td width="100"> </td>
          <td>
            <input name="add" type="submit" id="add" value="Add Person">
          </td>
        </tr>
      </table>
    </form>
  <?php
```

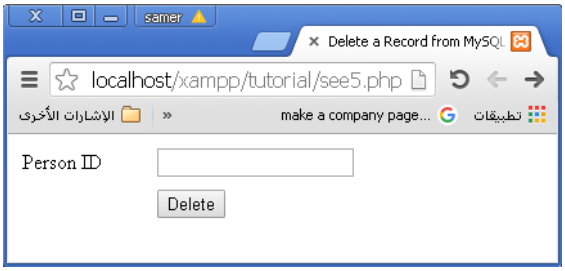
```
    }
  ?>
</body></html>
```

خطوات عمل الجزء الخامس

لا دوال جديدة هنا، والمبدأ الذي تم استخدامه في المثال السابق، سيتم تطبيقه نفسه هنا أيضا.

المثال الخامس see5.php

في هذا المثال ستعمل، عزيزي الدارس، على إنشاء النموذج form التالي، والذي يطلب من المستخدم تحديد رقم الشخص أو (الزبون) من اجل إلغاء بياناته كافة من قاعدة البيانات.




```
<html>
  <head>
    <title>Delete a Record from MySQL Database</title>
  </head>
  <body>
    <?php
      if(isset($_POST['delete'])) {
        $dbhost = 'localhost:3306';
        $dbuser = 'root';
        $dbpass = '';
        $conn = mysql_connect($dbhost, $dbuser, $dbpass);
        if(! $conn ) {
          die('Could not connect: ' . mysql_error());
        }
        $id1 = $_POST['id1'];
        $sql = "DELETE FROM Persons WHERE P_Id = $id1" ;
        mysql_select_db('customers');
        $retval = mysql_query( $sql, $conn );
          echo "<br> $retval\n";
        if(! $retval )
        {
          die('Could not delete data: ' . mysql_error());
        }
          else
                  echo "<br> Data deleted successfully\n";
        mysql_close($conn);
      }
      else {
        ?>
          <form method = "post" action = "<?php $_PHP_SELF ?>">
            <table width = "400" border = "0" cellspacing = "1" cellpadding = "2">
              <tr>
                <td width = "100">Person ID</td>
                <td><input name = "id1" type = "text" id = "id1"></td>
              </tr>
              <tr>
                <td width = "100"> </td>
                <td> </td>
              </tr>
```

```
                <tr>
                  <td width = "100"> </td>
                  <td>
                    <input name = "delete" type = "submit" id = "delete" value = "Delete">
                  </td>
                </tr>
              </table>
            </form>
          <?php
        }
      ?>
    </body>
</html>
```

النشاط الصفي السادس

استدعاء الإجراء عن بعد بلغة بي اش بي

XML RPC with PHP

مقدمة

الربط مع الإجراءات عن بعد Remote Procedure Call (RPC)، هي تقنية اتصال سهلة وبسيطة تمكن المبرمج من الاتصال بين خادم ومجموعة من الزبائن يعملون على منصات platform مختلفة (نظم تشغيل مختلفة) بحيث توفر بيئة محوسبة موزعة متعددة المنصات (Cross–Platform Distributed Computing) بالاعتماد على بروتوكول نقل النصوص في الانترنت HTTP. وبما أن لغة PHP تستخدم بروتوكول HTTP للنقل، ولغة XML للشيفرة encoding، فإن هذا يسمح نقل تراكيب بيانات معقدة complex data structures بين العقد في الشبكة وهذه الطريقة تسمى XML-RPC في لغة PHP. بالتالي يمكن لزبون client يعمل على جهازه من نوع Dell، بمنصة نظام ويندوز، أن يتصل مع خادم يعمل على جهاز نوع HP، بمنصة نظام لينوكس Linux، ويتبادل البيانات مع الخادم أيضا بفضل وسيلة الاتصال XML-RPC باستخدام بروتوكول HTTP. ولكن هذا يتطلب وجود شبكة تدعم بروتوكول الاتصال HTTP مثل الانترنت أو الانترانت Internet/Intranet.

الأهداف

1. أن تتعرف على تقنية الاتصال RPC في لغة PHP والتي تغلفها بلغة XML، أي XML-RPC.
2. أن تتعرف على الدوال المكتوبة بلغة PHP والتي تسهل على المبرمج الاتصال بتقنية XML-RPC.
3. أن تكتب بلغة PHP برنامج يمثل خادم وبرنامج آخر يمثل زبون، تنفذ البرنامجين من خلال جهازيّ حاسوب اثنين، وسيلة الاتصال بينهما هي XML-RPC.
4. أن تعمل على تطوير البرنامجين السابقين بحيث يقوم الخادم بالربط مع قاعدة بيانات ليلبي احتياجات طلب الزبون.

خطوات عمل الجزء الأول

في هذا الجزء، ستتعرف، عزيزي الدارس، على تقنية الاتصال RPC في لغة PHP والتي تغلفها بلغة XML، أي XML-RPC. ثم ستتعرف على الدوال المكتوبة بلغة PHP والتي تسهل الاتصال بتقنية XML-RPC. وأخيراً ستكتب بلغة PHP برنامج يمثل خادم وبرنامج آخر يمثل زبون، تنفذ البرنامجين من خلال جهازيّ حاسوب اثنين، بوساطة تقنية XML-RPC. بالتالي سوف يتم تحقيق الأهداف الثلاثة الأولى المذكورة أعلاه.

الملف الأول: service.php

يمثل هذا البرنامج خادما بسيطا يعمل على الرد على استفسار من زبون. فيستقبل رقما من الزبون ويقوم بضربه بالرقم "8" ثم يعيد الناتج إلى شاشة الزبون.

```
<?php
$request_xml = file_get_contents("php://input"); #inputstream
function databases($method_name, $args, $app_data)
{
   $username = $args[0];
   $password = $args[1];
   $x1       = $args[2];
   /*Perform authentication and authorization here. If failed return a fault code or exit*/
  /* You may connect to databases and retrieve data here */
   return $x1*6;
}
$xmlrpc_server = xmlrpc_server_create();
xmlrpc_server_register_method($xmlrpc_server, "databases", "databases");
header('Content-Type: text/xml');
print
xmlrpc_server_call_method($xmlrpc_server, $request_xml, array());
?>
```

تم إنشاء الدالة التي سيقوم الزبون بمناداتها وهي بأسم databases، حيث يتم تمرير ثلاث متغيرات لها، يمثل المتغير الأول أسم الدالة method_name، والمتغير الثاني عبارة عن مصفوفة من المتغيرات تحمل أسم المستخدم username، وكلمة السر password، والمتغير الثالث يحمل الرقم الذي سيتم ضربه ب "8".
تسهل لغة PHP على المبرمج إنشاء خادم من خلال استدعاء الدالة xmlrpc_server_create() والتي تعيد خادم على شكل مصدر resource. وبما أننا نستخدم بروتوكول HTTP للنقل، فإن طرق هذا البروتوكول هي التي سوف تستخدم، ومنها get في حالة الخادم لاستقبال الطلب request من خلال الدالة file_get_contents، وكذلك post في حالة الزبون من اجل إرسال الطلب request. لذلك تم تعريف متغير بأسم $request_xml يعمل على تحديد get لاستقبال البيانات من خلال استدعاء الدالة xmlrpc_encode_request والتي تعمل على قراءة البيانات المرسلة وفك شفرة XML للحصول على تلك البيانات.

من أجل التعرف على الدالة التي تنفذ عن بعد، لا بدّ من تسجيل تلك الدالة في قاعدة بيانات الخادم من خلال استدعاء الدالة xmlrpc_server_register_method(resource $server, string $method_name, string $function); والتي تأخذ ثلاثة معاملات، أسم الخادم الذي تم إنشاؤه، وأسم الطريقة التي توافق اسم الدالة المناداة وهي نفسها المعامل الأخير.

يتم الرد على الزبون باستدعاء الدالة print لتطبع على البروتوكول نتيجة تنفيذ الدالة xmlrpc_server_call_method($xmlrpc_server, $request_xml, array());، والتي تأخذ ثلاث معاملات، مصدر الخادم resource $server، الطلب على شكل شيفرة xml والمعاملات التي سيتم ارجاعها على شكل مصفوفة.

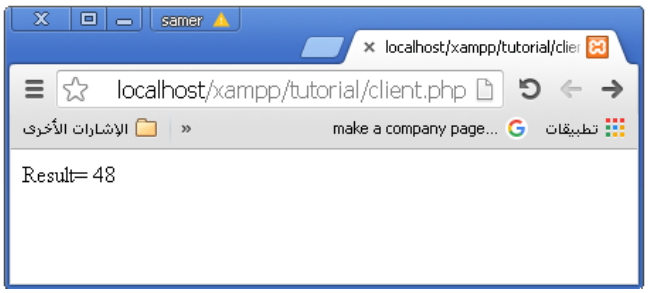
سؤال 1: أذكر خطوات إنشاء خادم XML-RPC، في لغة PHP؟

البرنامج الثاني client.php

من جهة الزبون، يجب مناداة الدالة التي تعمل على إنشاء دفق stream للمحتوى context وهي الدالة stream_context_create(array('http' => array()))، والتي تأخذ معامل على شكل مصفوفة، تحدد كل من بروتوكول الاتصال وهو HTTP و كذلك معاملات هذا البروتوكول. فطريقة الارسال هي POST كما ذكر أعلاه، والرأسية header، توضح أن نوع المحتوى هو نصوص xml أو text/xml، وأن عامل المستخدم User-Agent هو تقنية الاتصال RPC من خلال لغة PHP أو PHPRPC.

```
<?php
$request = xmlrpc_encode_request("databases", array('username', 'password', 8));
$context = stream_context_create(array('http' => array(
   'method' => "POST",
   'header' => "Content-Type: text/xml\r\nUser-Agent: PHPRPC/1.0\r\n",
   'content' => $request
)));

/*This URL may not exist. Replace it with your server URL*/
$server = 'http://localhost/xampp/tutorial/service.php';
$file = file_get_contents($server, false, $context);
$response = xmlrpc_decode($file);
echo "Result= " . $response;
?>
```

الآن يجب طباعة الناتج على شاشة الزبون كما هو مبين في الصورة أدناه. يتم ذلك من خلال مناداة الدالة $xmlrpc_decode() والتي تأخذ معامل على شكل ملف يحدد طريقة الحصول على المحتوى جهة الخادم وهي get. يتم تعريف هذا المعامل من خلال الدالة get_file_contents()، والتي تعيد ما تقرأه من بيانات الملف على شكل سلسلة string، ولكنها تأخذ بدورها ثلاث معاملات، المعامل الأول هو أسم الخادم، والمعامل الثاني false لعدم تحديد المسار، والثالث المحتوى context الذي تم إنشاؤه سابقا.



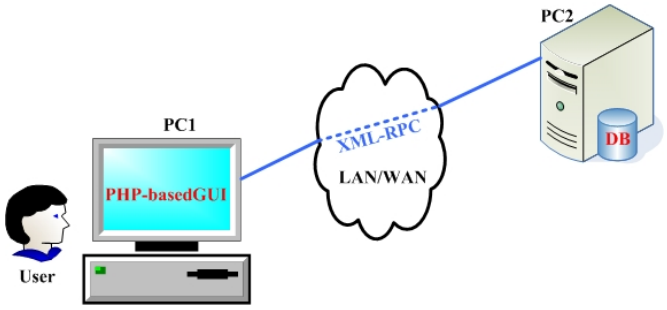
ملاحظة: استبدل اسم الجهاز المحلي localhost، باسم الجهاز الذي يوجد عليه الخادم. يمكنك الاستعانة بمشرف المختبر ليعطيك اسم الجهاز.

سؤال 2: أذكر خطوات إنشاء زبون XML-RPC، في لغة PHP؟

خطوات عمل الجزء الثاني

في هذا الجزء، ستعمل عزيزي الدارس من خلال التدريب التالي على ربط الخادم server مع قاعدة بيانات بحيث يطلب الزبون client بيانات من الخادم، فيقوم الخادم بإعادة تلك البيانات من قاعدة بيانات تم إنشاؤها مسبقا.

تدريب 1: اعمل على إضافة الجزء البرمجي الخاص بك لكل من برنامج الخادم وبرنامج الزبون بحيث يطلب الزبون، طالب جامعي فرضا، جلب علاماته من الخادم. انظر الصورة أدناه.



النشاط الصفي السابع

ربط خوادم RMI و CORBA بقواعد البيانات MySQL

مقدمة

إن مصدر قوة بروتوكول الاتصال سواء كان RMI أو CORBA أو RPC، تكمن في ربط خادمه بقاعدة بيانات. في هذه التجربة، ستقوم عزيزي الدارس باستخدام نفس البرامج التي كتبتها في التجربتين الأولى والثانية من أجل ربط الخادم بقاعدة البيانات customers. ولكن لا بد من أن تتعرف أولا على طريقة الربط بين لغة البرمجة جافا، وقاعدة البيانات customers في MySQL.

الأهداف

1. أن تتعرف على طريقة جافا في الربط مع قواعد البيانات MySQL.
2. أن تتعرف على الطرق methods والأصناف classes التي تعرفها جافا للربط مع قواعد البيانات MySQL.
3. أن تكتب بلغة Java برنامجا يمثل خادم وبرنامجا آخر يمثل زبون، تنفذ البرنامجين من خلال جهازيّ حاسوب اثنين، وسيلة الاتصال بينهما هي RMI، حيث يرتبط خادم RMI مع قاعدة البيانات.
4. أن تكتب بلغة Java برنامجا يمثل خادم وبرنامجا آخر يمثل زبون، تنفذ البرنامجين من خلال جهازيّ حاسوب اثنين، وسيلة الاتصال بينهما هي CORBA، حيث يرتبط خادم CORBA مع قاعدة البيانات.

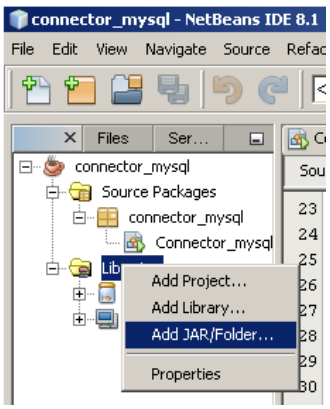
خطوات عمل الجزء الأول

في هذا الجزء، ستتعرف، عزيزي الدارس، على طريقة ربط جافا مع قاعدة البيانات التي تم إنشاؤها مسبقا باسم customers في MySQL.

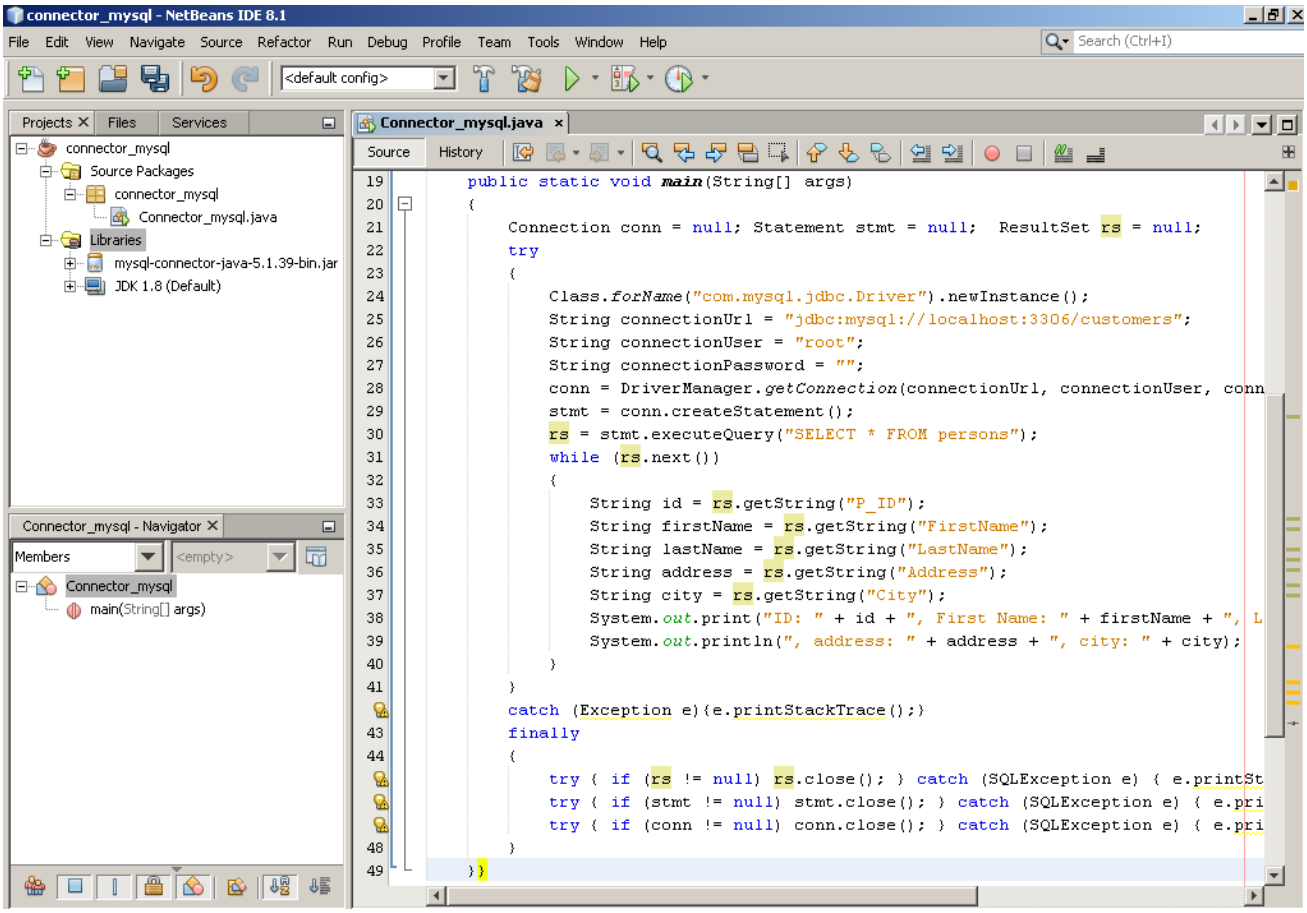
الملف الأول: Connector_mysql.java

يوضح هذا البرنامج طريقة ربط جافا مع MySQL، باستخدام طريقة الربط المعيارية وهي Connector/J. يجب تحميل هذا الملف المضغوط من الموقع التالي http://dev.mysql.com/downloads/file/?id=462850. قم بفك ضغط الملف، على مجلد معين لحين الخطوة التالية.

ثم وبعد ذلك افتح برمجية NetBeans IDE، أغلق جميع المشاريع المفتوحة إن وجد، واعمل على إنشاء مشروع جديد. أعط أسم للحزمة connector_mysql، مبقيا على الافتراضات ومنها أسم الصنف إن أردت Connector_mysql ومنهج main. انسخ البرنامج التالي في داخل الصفحة أن أبقيت على المسميات السابقة. الآن اعمل على إضافة مكتبة إلى المكتبات libraries من خلال جلب ملف jar.



ثم اختار ملف mysql-connector-java-5.1.39-bin.jar من داخل المجلد الذي قمت بفك ضغطه.




```
package connector_mysql;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

/**
 * @author Dr. Eng. Samer Jaloudi
 * QOU, Nablus, Palestine
 */
public class Connector_mysql
{
   public static void main(String[] args)
   {
          Connection conn = null;
          Statement stmt = null;
          ResultSet rs = null;
          try
          {
                Class.forName("com.mysql.jdbc.Driver").newInstance();
                String connectionUrl = "jdbc:mysql://localhost:3306/customers";
                String connectionUser = "root";
                String connectionPassword = "";
                conn = DriverManager.getConnection(connectionUrl, connectionUser, connectionPassword);
                stmt = conn.createStatement();
                rs = stmt.executeQuery("SELECT * FROM persons");
                while (rs.next())
```

```
                {
                        String id = rs.getString("P_ID");
                        String firstName = rs.getString("FirstName");
                        String lastName = rs.getString("LastName");
                        String address = rs.getString("Address");
                        String city = rs.getString("City");
                        System.out.print("ID: " + id + ", First Name: " + firstName + ", Last Name: " + lastName);
                        System.out.println(", address: " + address + ", city: " + city);
                }
        }
         catch (Exception e){e.printStackTrace();}
         finally
         {
                try { if (rs != null) rs.close(); } catch (SQLException e) { e.printStackTrace(); }
                try { if (stmt != null) stmt.close(); } catch (SQLException e) { e.printStackTrace(); }
                try { if (conn != null) conn.close(); } catch (SQLException e) { e.printStackTrace(); }
        }
   }
}
```

الخطوات التي يجب إتباعها في الربط مع قاعدة بيانات لجلب بيانات منها:

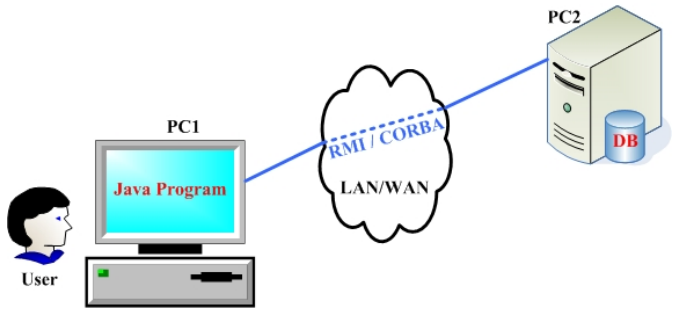
1. أنشئ مثيل من أداة الربط مع قاعدة البيانات Connection بإسم conn، ومثيل من جملة قاعدة البيانات Statement بإسم stmt، ومثيل آخر من مجموعة النتيجة ResultSet بأسم rs لحفظ النتيجة في هذا المتغير.
2. أنشئ مثيل من سائق الربط Driver مع MySQL، باستخدام الكود البرمجي
   Class.forName("com.mysql.jdbc.Driver").newInstance();
3. من خلال سلسلة String، عرف ثلاثة متغيرات، متغير يحمل عنوان مضيف قاعدة البيانات ومنفذها (3306)، ومتغير آخر يحمل اسم المستخدم لقاعدة البيانات "root"، ومتغير ثالث يحمل كلمة المرور لقاعدة البيانات "". طبعا نتركها فارغة هنا لأننا لم نحدد كلمة مرور مسبقا على customers.
4. الآن يمكن تنفيذ الربط مع قاعدة البيانات من خلال المنهج getConnection التابعة للصنف DriverManager، حيث تأخذ هذه الطريقة ثلاثة متغيرات وهي التي تم تعريفها في النقطة السابقة (نقطة 3).
5. ثم نفذ جملة استعلام SQL من خلال استدعاء المنهج executeQuery، وهذا المنهج يأخذ متغير من نوع سلسلة، يحمل جملة الاستعلام Query، التي نريد تنفيذها في قاعدة البيانات. نتيجة التنفيذ توضع في المتغير الذي تم انشاؤه مسبقا باسم قس كنوع من مجموعة النتيجة ResultSet.
6. مستخدما جملة دوران، اطبع جميع المحتويات التي تم جلبها من القاعدة.
7. إغلق جميع منافذ الربط والتعريفات التي تتعلق بقاعدة البيانات.
8. هذا المثال يستخدم مكتبة sql، فيمكن استيراد ما نحتاجه من المكتبة من خلال جملة واحدة ;*.java.sql.import.

خطوات عمل الجزء الثاني

في هذا الجزء، ستعمل عزيزي الدارس من خلال التدريبين التاليين على ربط الخادم server مع قاعدة بيانات بحيث يطلب الزبون client بيانات من الخادم، فيقوم الخادم بإعادة تلك البيانات من قاعدة بيانات تم إنشاؤها مسبقا. ستطبق ذلك في RMI وكذلك CORBA.

تدريب 1: عدل على كل من برنامج خادم RMI وبرنامج الزبون RMI بحيث يطلب الزبون، طالب جامعي فرضا، جلب علاماته من الخادم. انظر الصورة أدناه.

تدريب 2: عدل على كل من برنامج خادم CORBA وبرنامج الزبون CORBA بحيث يطلب الزبون، طالب جامعي فرضا، جلب علاماته من الخادم. انظر الصورة أدناه.



<u>تدريب 3:</u> عدل على كل من برنامج خادم XML-RPC وبرنامج زبون XML-RPC في التجربة السابقة بحيث يطلب الزبون، طالب جامعي فرضا، جلب علاماته من خادم عليه بياناته الشخصية، وهذا الخادم بدوره يجلب العلامات من خادم آخر بنفس تقنية الاتصال المستخدمة. انظر الصورة أدناه. يمكنك استخدام تقنية الاتصال RMI أو تقنية CORBA.