Delphi Tuts

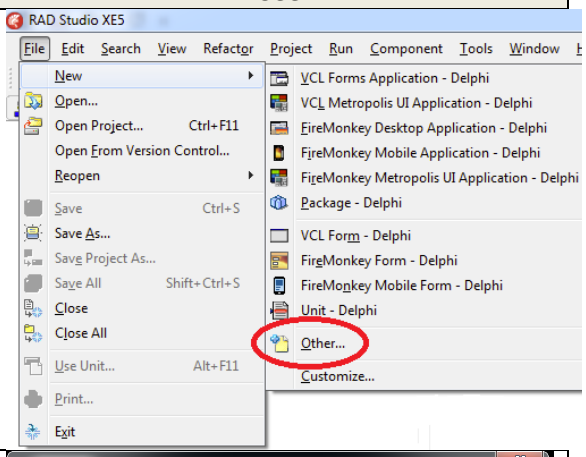
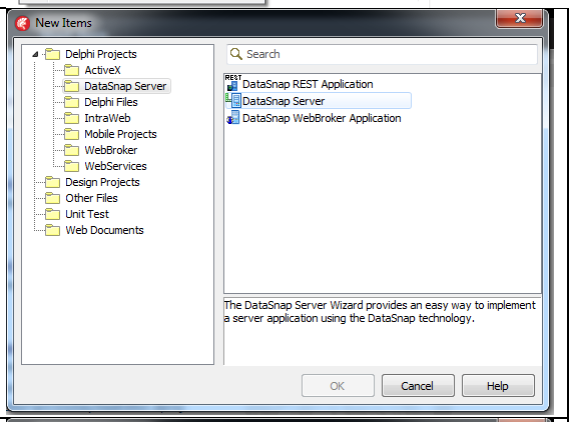
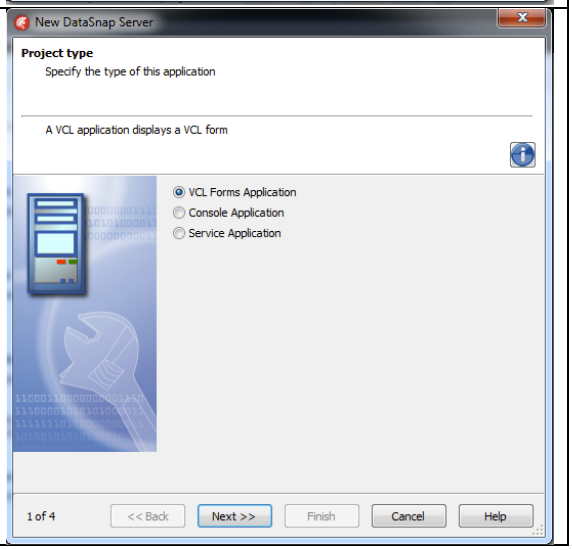
DataSnap “Hello World”

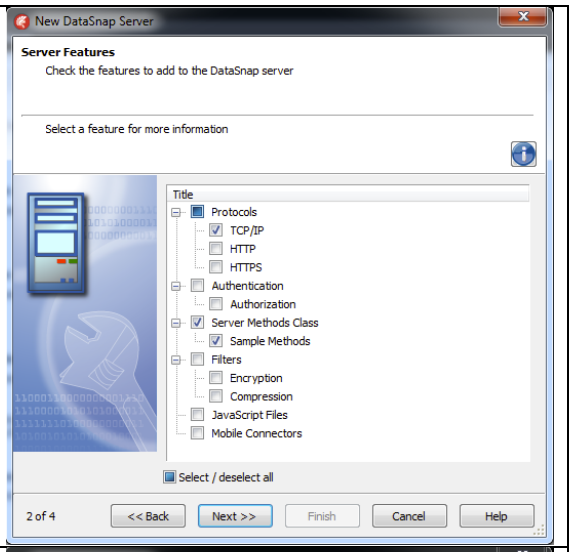
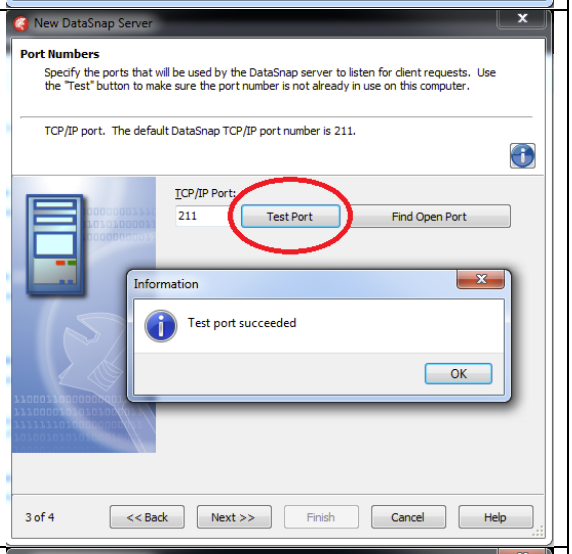
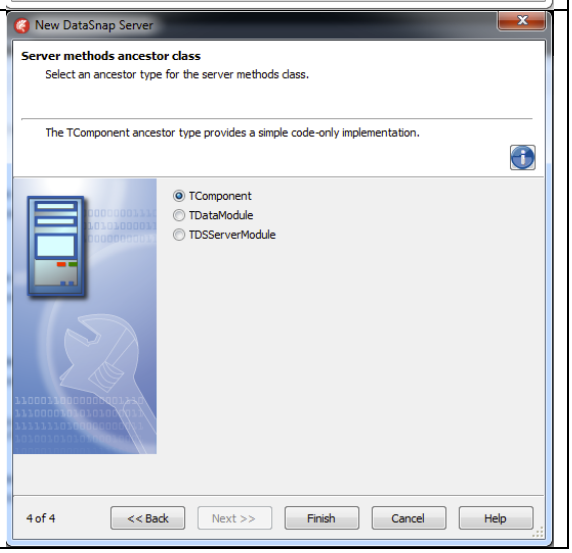
Paweł Głowacki - Embarcadero

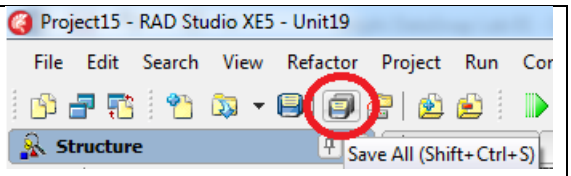
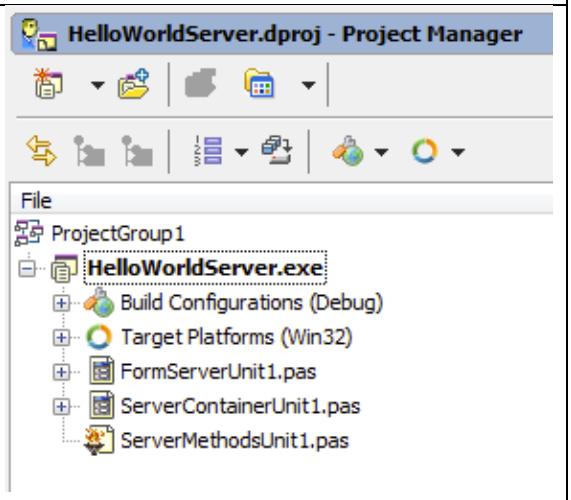
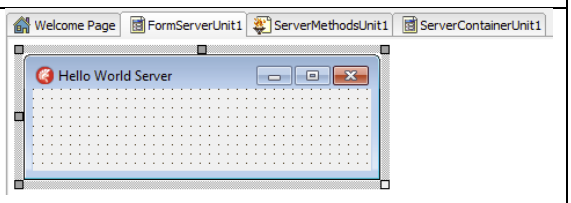
In this tutorial we are going to use Delphi XE5 to build the simplest possible DataSnap client/server system. The difficulty level of this lab is “Hello World”.

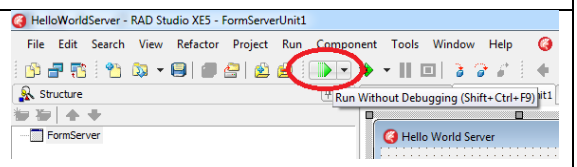
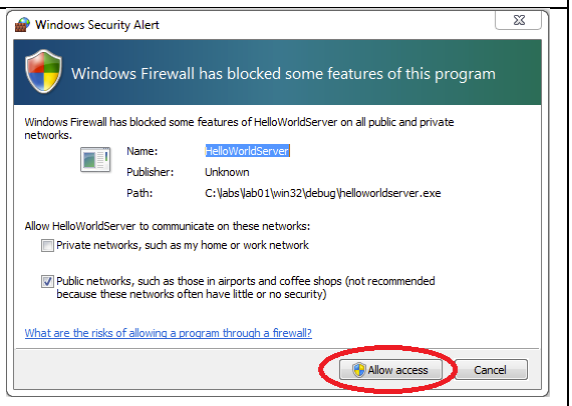
This step-by-step tutorial is intentionally very simple, so even not experienced Delphi programmers should be able to build projects described here. The objective of this lab exercise is to get familiar with basic steps needed for building DataSnap servers and clients.

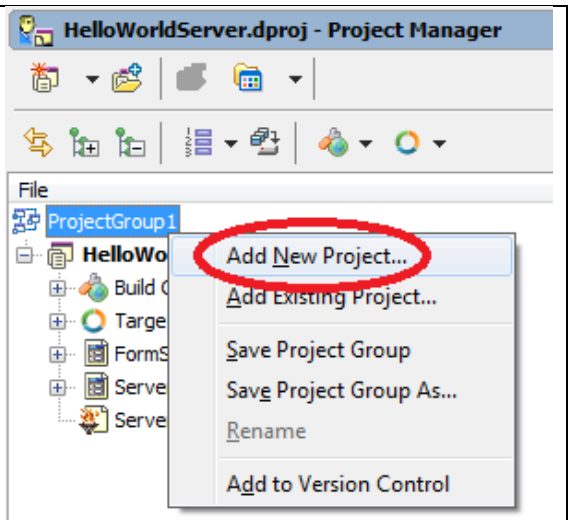
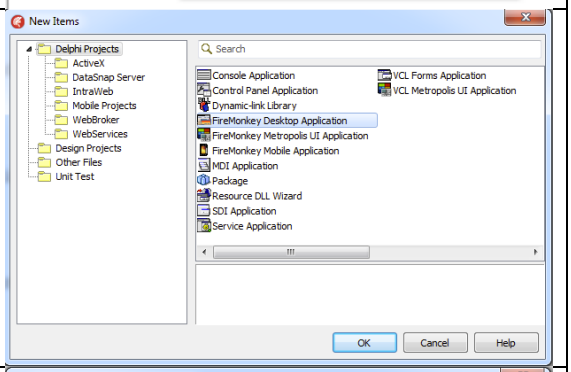
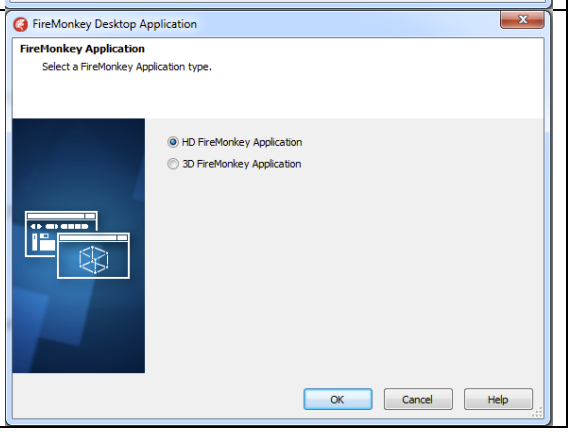
Our system will consist of a server and a client applications. They will use TCP/IP as the communication protocol. The server will be implemented as Delphi VCL Forms application and the client will be FireMonkey Desktop Application. The server will provide two methods that a client can invoke: “EchoString” and “ReverseString”. In this demo we are going to build a very simple client with just an edit and a button. When a user clicks on the button, the contents of the edit box will be sent to the server, reversed and the result will be displayed in the edit.

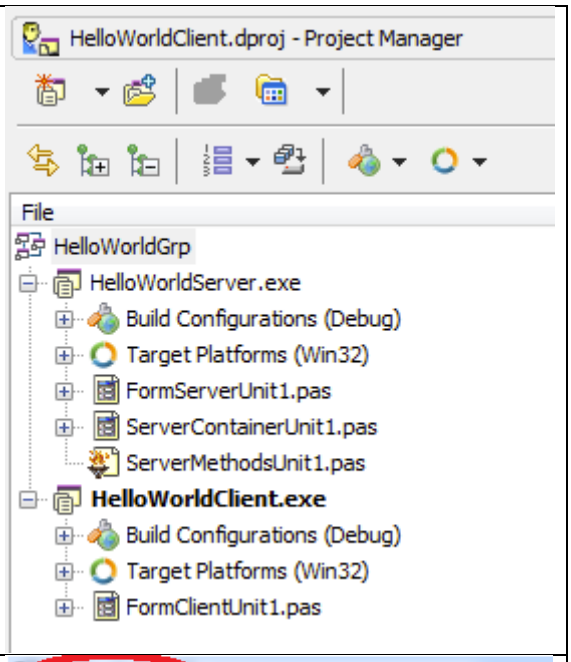
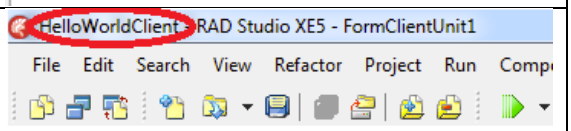
Do	Say	See
Start Delphi XE5. Select in the main menu “File -> New -> Other”.	The first step is to create a new DataSnap server application using “DataSnap Server” wizard.	
In the “New Items” dialog double-click on the “DataSnap Server” icon in the “Delphi Projects -> DataSnap Server” category.		
In the first tab keep the default DataSnap “Project type” which is “VCL Forms Application”.		

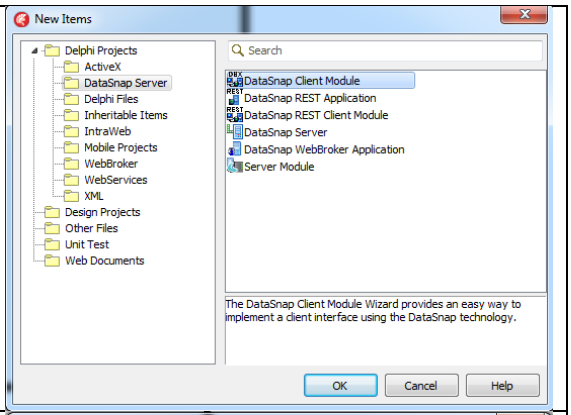
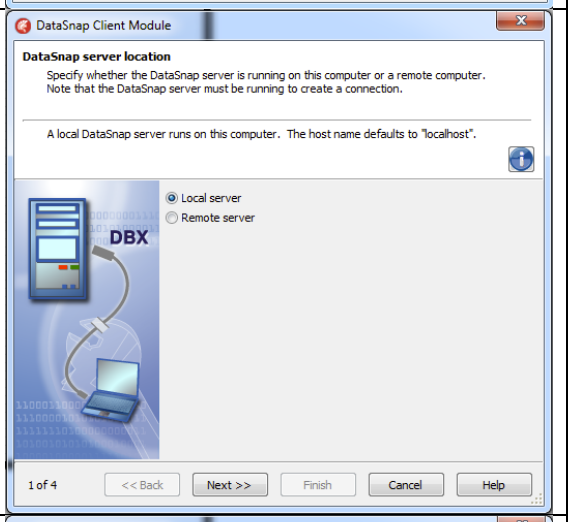
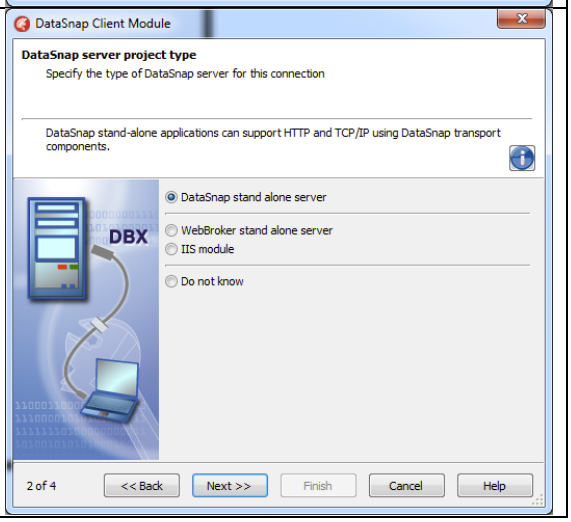
On the second tab keep all the default values. Later we are going to replace sample DataSnap server methods with our own implementation.		
On the third screen we keep the default value “211” for the TCP/IP Port. It is always a good idea to click on the “Test Port” to make sure that the selected port is available.		
On the last tab we are going to keep default server methods ancestor class which is “TComponent”. Click on “Finish” and the wizard should create a new project with three units.		

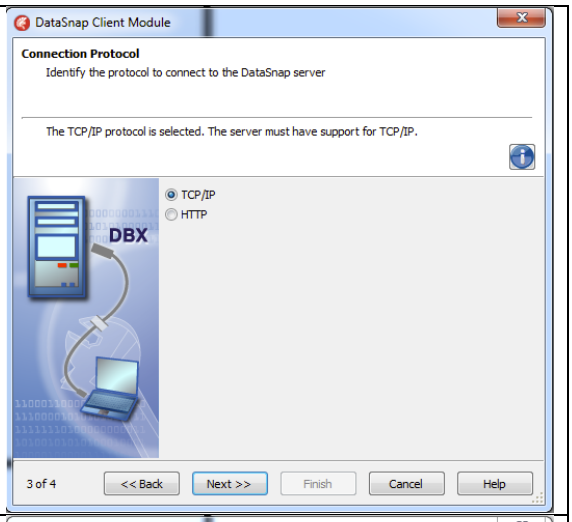
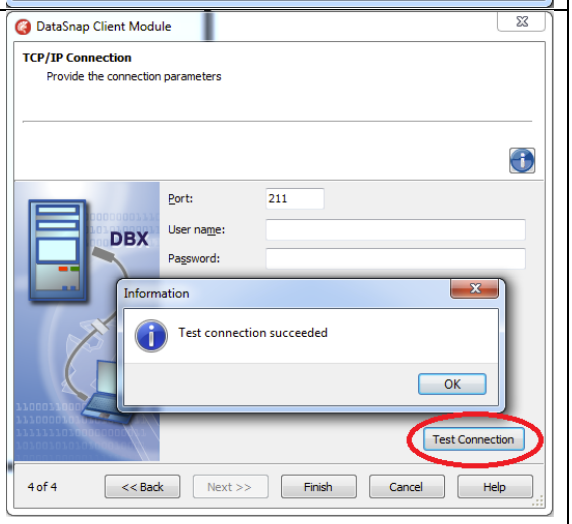
Save the whole project by clicking on “File -> Save All”. Alternatively you can click on the “Save All” icon or use “Shift+Ctrl+S” keyboard combination.		
Create a new directory for all files in this lab – for example “C:\Labs\Lab01\”. Save main application form as “FormServerUnit1” and keep default names for all other units – typically “ServerContainerUnit1” and “ServerMethodsUnit1” – and save project as “HelloWorldServer”.	At this stage you should see the following in the Delphi Project Manager	
Change the “Caption” property of the form to “Hello World Server”. Resize the form to make it smaller as it is going to be empty. Change the “Name” property of the form to “FormServer”. Save All.	Let’s make sure that all names are meaningful in the server project. This step is optional but making sure that everything has a proper name and caption is a good practice. Double-click on the “FormServerUnit1” form in the Project Manager.	
Double-click on the “ServerMethodsUnit1.pas” unit in the Project Manager and show its content in the editor.	We have chosen in the “DataSnap Server” wizard to add a server method class with sample methods. As this is a really simple demo we are not going to change anything in server methods unit. There are two public methods in the “TServerMethods1” class: “EchoString” and “ReverseString”. Both are taking one string	

	parameter and returning a string value. “EchoString” just returns the input parameter and “ReverseString” reverses the parameter and returns the result. Later on we are going to call the “ReverseString” method from our client application.	
Click on the “Run Without Debugging” icon to start the server. Alternatively you could select “Run -> Run Without Debugging” main menu option or press “Shift+Ctrl+F9” keyboard shortcut. Minimize the server window.	Our server is now fully implemented. In order to develop the client application, the server has to be running.	
Do not shut down the server until the end of this lab exercise.	The very first time that you start the server application, you may receive the Windows Firewall warning that some of the program features are blocked. Make sure that you click on the “Allow access” button to allow the server to communicate with clients.	

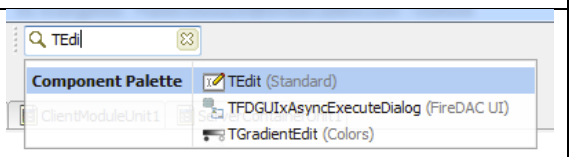
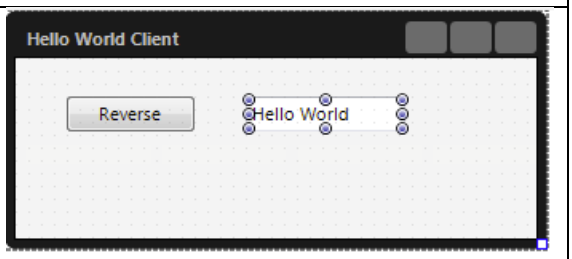
Right-click on the project group inside the Project Manager and select “Add New Project”.	Now it is time to create “Hello World” Client Application	
From the “New Items” dialog select “FireMonkey Desktop Application” from “Delphi Projects” category. Click “OK”.		
On the next scree keep “HD FireMonkey Application” as the application type to be created. “HD” stands for “High Definition” and is just a more fancy term for a traditional “2D” graphical user interface. Click “OK”.		

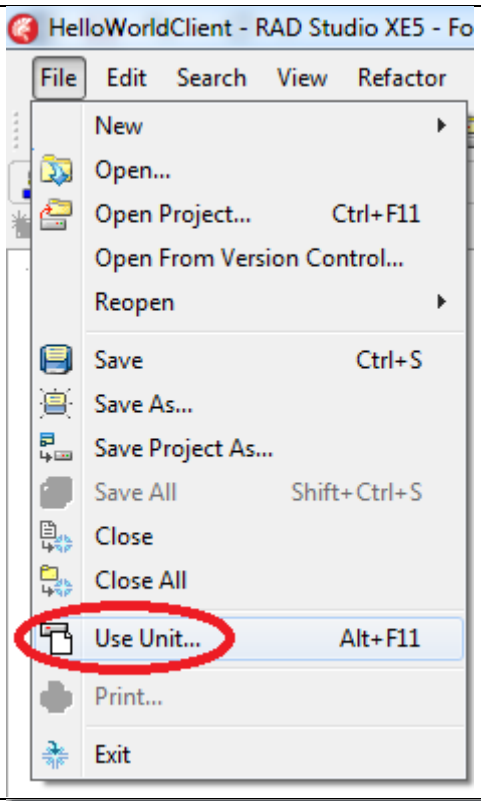
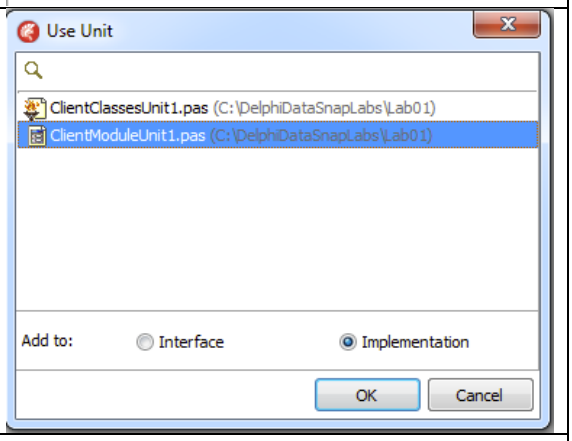
Now click on “Save All” icon. Browse to the folder where the server project has been saved (“C:\ Labs\Lab01”) and save there the main form unit of the client application as “FormClientUnit1”, the new project as “HelloWorldClient” and the project group as “HelloWorldGrp”.	A new project should be added to the existing project group. At this moment the Project Manager should look like this:	
	Our project contains now two projects: the server and the client. At any given time there could be only one “Active” project in the IDE (“Integrated Development Environment”). All commands that you choose in the IDE are applied to the “Active” project. In order to make sure that a given project is “active” you can double-click its name in the Project Manager. The name of the currently active project is displayed in “bold” in Project Manager and you can also see its name as part of the caption of Delphi IDE main window.	

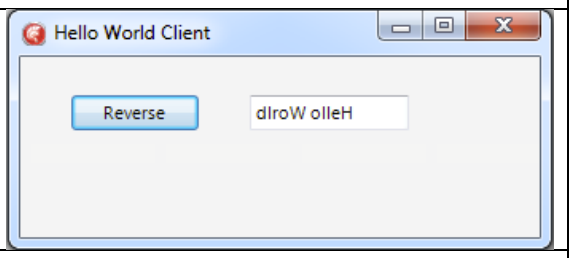
Make sure that the “HelloWorldClient” project is active. Click on “File -> New -> Other” and select “DataSnap Client Module” wizard. Click on “OK”.	Again we are going to keep all the default values in the wizard.	
Click on “Next”	On the first tab keep the “DataSnap server location” to “Local server”	
Click on “Next”	On the next tab we keep the default value “DataSnap stand alone server”	

Click on “Next”	Our server is using “TCP/IP” as the connection protocol, so keep the default selection on the next screen of the wizard.	
Click on “Test Connection”, verify that the “Test connection succeeded” information is displayed. Click on “OK” to close message dialog. Click on “Finish” to close the wizard.	On the last screen click on “Test Connection” to verify that the server listens on the default port 211 and click “Finish”. This is important the connection is OK. Otherwise the wizard would not be able to proceed. It needs access to running server in order to be able to query it for its functionality and generate appropriate source code after we hit “Finish”.	
Click on “Save All”	The wizard added two units to the client project: “ClientClassesUnit1” and “ClientModuleUnit1”.	
Double-click on the “ClientClassesUnit1” in the Project Manager and show its content in the editor.	. At the top of the unit you will see a comment that this unit was created by DataSnap proxy generator. In this unit there is “TServerMethods1Client” class, which has among other functions, the “EchoString” and “ReverseString” methods that look like their corresponding methods in	

	the server methods unit in the server project. This is why the server application has to be running why we generate client code using the wizard. The proxy generator checks what the functionality of the running server is and generates appropriate code.	
Double-click on the “ClientModuleUnit1” in the Project Manager.	Now let’s open the “ClientModuleUnit1”. This a data module. A container for non-visual components. The wizard added a “TSQLConnection” component to the form that has all information about the network address of the server, its communication protocol and port number used for communication. There is also convenient code generated, so the client application can just access the “ServerMethods1Client” property and just start invoking the methods of the “TServerMethod1Client” class that is instantiated on the first access to the property.	
Double-click on the “FormClientUnit1” in the Project Manager to display it.	The last task in this lab is to implement the user interface of the client application.	
Change the “Caption” property of the form to “Hello World Client” and its “Name” property to “FormClient”.	Let’s build a simple user interface for our client. We will need an edit box entering the string to be reversed and displaying the result, and a button to invoke the	

	“ReverseString” operation.	
“Ctrl” and “.” keys at the same time to focus “Search” box at the top of the IDE. Start typing “TEdit” and press “Enter” when it is selected. In the same way add a “TButton” component.	In the default layout, at the bottom right part of the IDE there is a Tool Palette with hundreds of reusable components. One way of adding a component to the form is to double-click its icon in the Tool Palette. However if we know what we want to add, probably the fastest way of doing so, is to use “Search” functionality. Alternatively we could use the “Search” box in the “Tool Palette” itself.	
Change the “Text” property of the “Button1” to “Reverse” and change the “Text” property of the “Edit1” component to “Hello World”.	The client’s user interface is ready. Now we need to add some code that will be executed when user clicks on the button.	

Select “File -> Use Unit”	The wizard has generated for us two units with everything that is necessary to connect to our server and invoke its operations. The first step is to make sure that this code is accessible from our client form. For this we need to add “ClientModuleUnit1” to the “uses” clause of the form. We could add this one line of code to the form, but it is faster to use the wizard	
In the “Use Unit” dialog select “ClientModuleUnit1” and click on “OK”	Now we are going to write some code that will be executed when the end user click on any of the button. This code will take the contents of the edit box, invoke the “ReverseString” server method and will display the result in the edit.	
Double click on the button component. In the code editor write one line of code: Edit1.Text := ClientModule1.ServerMethods1Client.ReverseString(Edit1.Text);	Double click on the button component. The IDE will move you from the form designer to the code editor, generate an empty “OnClick” event handler for the button and put a cursor in the first line of the event handler where we need to put the code that will be executed when end user will click on the button.	

	We just need to write one line of code.	
Save All and run the client application.	Now we just need to run the client application and click on the button. You should see that the contents of the edit box is reversed.	
	That’s it! Now you have got the basic skills to use Delphi and create powerful client/server systems with DataSnap!	
Summary	In this tutorial we have used Delphi XE5 for building a simple client/server system with DataSnap technology. We have been using wizards extensively to make our life simpler and build applications faster. Our server has been implemented as a Windows VCL Forms application and client was created as FireMonkey Desktop Application. Client and server were using TCP/IP as the communication protocol to invoke sample “ReverseString” server method.	
	The difficulty level of this lab was “Hello World” and its goal was to get familiar with using Delphi integrated development environment to work with multiple projects, invoke wizards, use form designer and even write some real Delphi code. In order to keep this tutorial simple we have	

	only focused on creating a client application for Windows, but we could easily recompile it for Mac OS X.	
	In the next “Delphi Tuts” step-by-step tutorial we are going to build DataSnap clients for all platforms supported in Delphi XE5: Windows, Mac OS X, iOS and Android!	