2 Configuring Source and Library in Visual Studio .NET 2003

Chapter 2 - Configuring Source and Library in Visual Studio .NET 2003

Microsoft Visual Studio .NET 2003 is an advanced integrated development environment by Microsoft. It lets programmers to create programs that run on Microsoft Windows and the World Wide Web.

In chapter 2, you have generated C# code, exported database schema (DDL) and created persistent library. Now you can create a project for the generated C# code in Visual Studio .NET 2003.

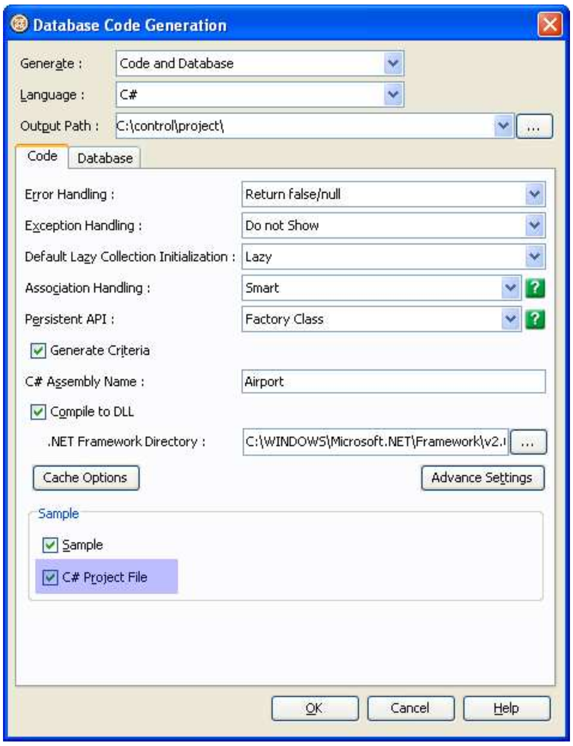
In this chapter:

- Copying Generated Source and Library to Visual Studio .NET Project
- Adding Reference of an Existing Project to DB-VA Generated C# Project
- Testing the Visual Studio .NET Project

Copying Generated Source and Library to Visual Studio .NET Project

You can create Visual Studio .NET project easily because DB-VA supports the generation of C# project file. The project file helps you to configure the generated C# classes, resources files and referenced libraries in Visual Studio .NET.

1. Select C# Project File in the Database Code Generation dialog box



Database Code Generation

Generate : Code and Database

Language : C#

Output Path : C:\control\project\

Code | Database

Error Handling : Return false/null

Exception Handling : Do not Show

Default Lazy Collection Initialization : Lazy

Association Handling : Smart

Persistent API : Factory Class

☑ Generate Criteria

C# Assembly Name : Airport

☑ Compile to DLL

.NET Framework Directory : C:\WINDOWS\Microsoft.NET\Framework\v2.(

Sample

☑ Sample

☑ C# Project File

Figure 2.1 - The C# Project File options

2. The C# Project File is created in the **Output Path\src** folder

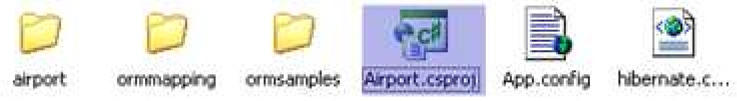


Figure 2.2 - The generated C# Project file

3. Open Visual Studio .NET 2003. From the menu bar, select **Open > Project.**

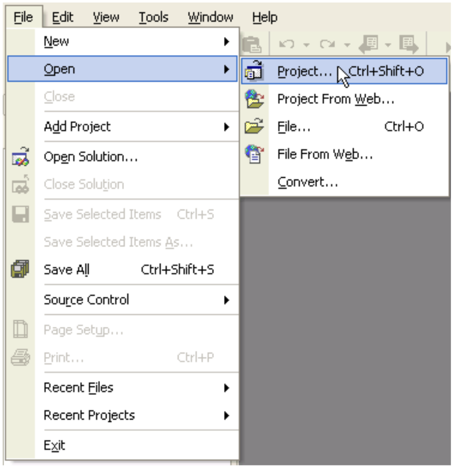
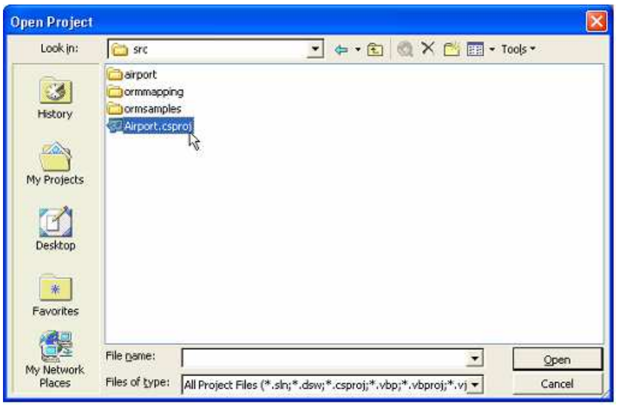


Figure 2.3 - Open a project files

4. Select the **Airport.csproj** and click **Open**



Look in: src

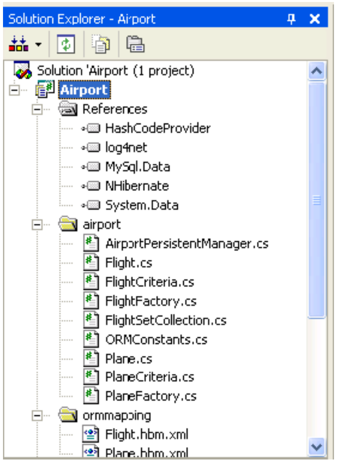
airport
ormmapping
ormsamples
Airport.csproj

File name:

Files of type: All Project Files (*.sln;*.dsw;*.csproj;*.vbp;*.vbproj;*.vj

Figure 2.4 - Select a project

5. All Libraries are added to References and C# classes are inside the project.



```
Solution 'Airport (1 project)
  Airport
    References
      HashCodeProvider
      log4net
      MySql.Data
      NHibernate
      System.Data
    airport
      AirportPersistentManager.cs
      Flight.cs
      FlightCriteria.cs
      FlightFactory.cs
      FlightSetCollection.cs
      ORMConstants.cs
      Plane.cs
      PlaneCriteria.cs
      PlaneFactory.cs
    ormmapping
      Flight.hbm.xml
      Plane.hbm.xml
```

Figure 2.5 - The project contains all sources and library

Adding Reference of an Existing Project to DB-VA Generated C# Project

You can also reference the generated C# Visual Studio .NET Project as a library to develop an application.

1. Open Visual Studio .NET Project. Select **File** > **New** > **Project** on menu bar.

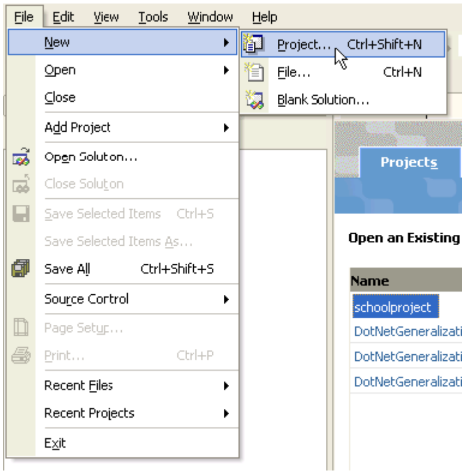
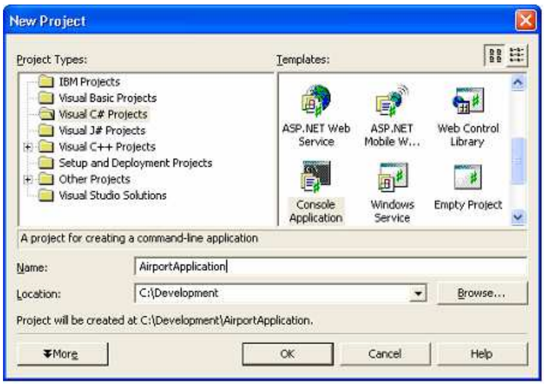


Figure 2.6 - To create a new project

2. Select Project Types as **Visual C# Projects** and Templates as **Console Application**. Enter the Project and Location for the new project and click **OK**.



A project for creating a command-line application

Name: AirportApplication

Location: C:\Development

Project will be created at C:\Development\AirportApplication.

Figure 2.7 - New Project dialog

3. The **AirportApplication** project is created.

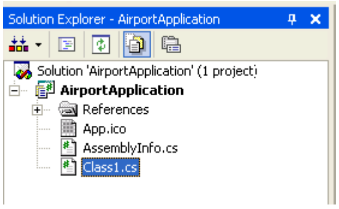
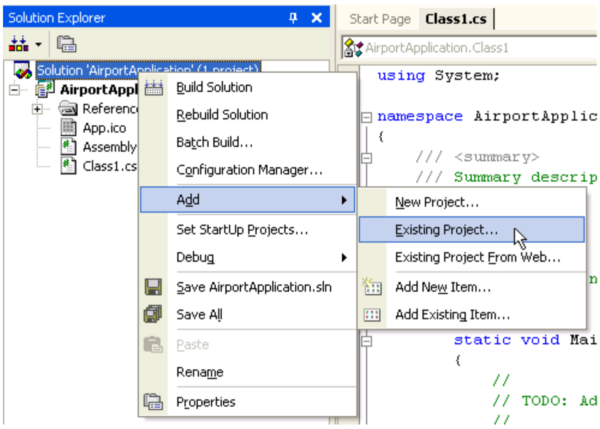

Solution Explorer - AirportApplication
Solution 'AirportApplication' (1 project)
AirportApplication
References
App.ico
AssemblyInfo.cs
Class1.cs

Figure 2.8 - Project created

You have an existing solution called AirportApplication and you may want to use the generated persistent code to develop a database application in AirportApplication.

4. Right click **Solution** > **Add** > **Existing Project...** to select the generated Project

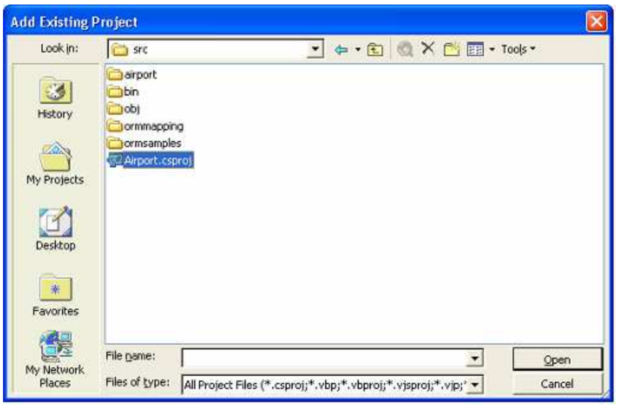


```
using System;

namespace AirportApplic
{
    /// <summary>
    /// Summary descrip
                                    n
        static void Mai
        {
            //
            // TODO: Ad
            //
```

Figure 2.9 - To add and existing project

5. Select the generated C# Project File **(Airport.csproj**) and add it to the existing Solution.



Look in: src

File name:

Files of type: All Project Files (*.csproj;*.vbp;*.vbproj;*.vjsproj;*.vjp;'

Figure 2.10 - select the project file

6. The generated C# project is added to **Solution**.

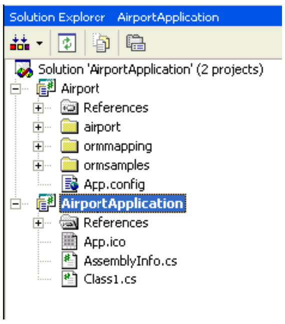


Figure 2.11 - Project imported

7. Right click **Airport** and select **Properties** on menu.

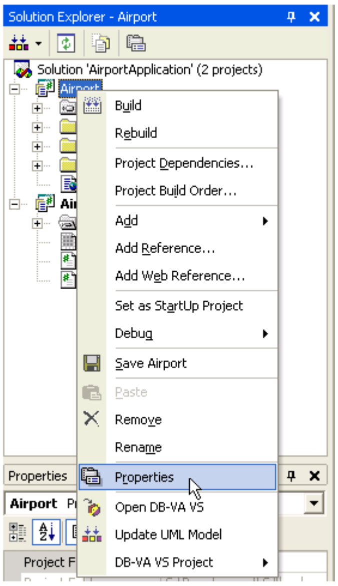
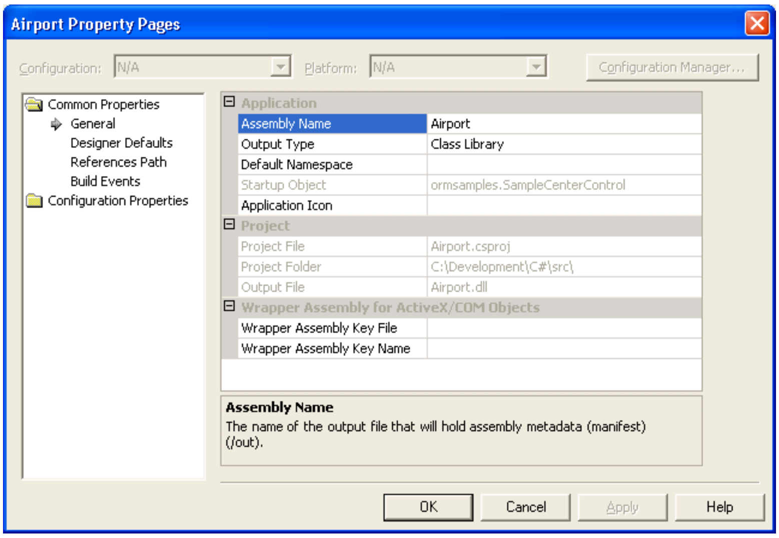


Figure 2.12 - To open the project properties

8. Change the **Output Type** from "**Windows Application**" to "**Class Library**". Click **OK**.



Application	
Assembly Name	Airport
Output Type	Class Library
Default Namespace	
Startup Object	ormsamples.SampleCenterControl
Application Icon	
Project	
Project File	Airport.csproj
Project Folder	C:\Development\C#\src\
Output File	Airport.dll
Wrapper Assembly for ActiveX/COM Objects	
Wrapper Assembly Key File	
Wrapper Assembly Key Name	

Assembly Name
The name of the output file that will hold assembly metadata (manifest) (/out).

Figure 2.13 - Project Property Page

9. From the menu bar, select **Build > Rebuild Solution**. The DLL file is generated.

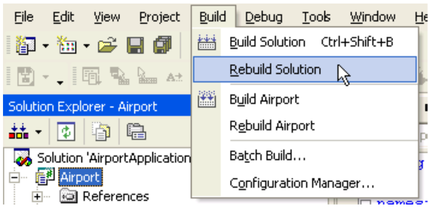


Figure 2.14 - To rebuild the solution

10. Right click the AirportApplication project and select **Add Reference...**on menu.

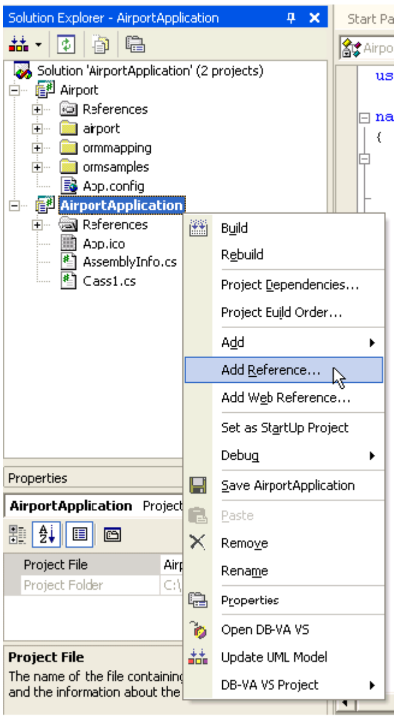
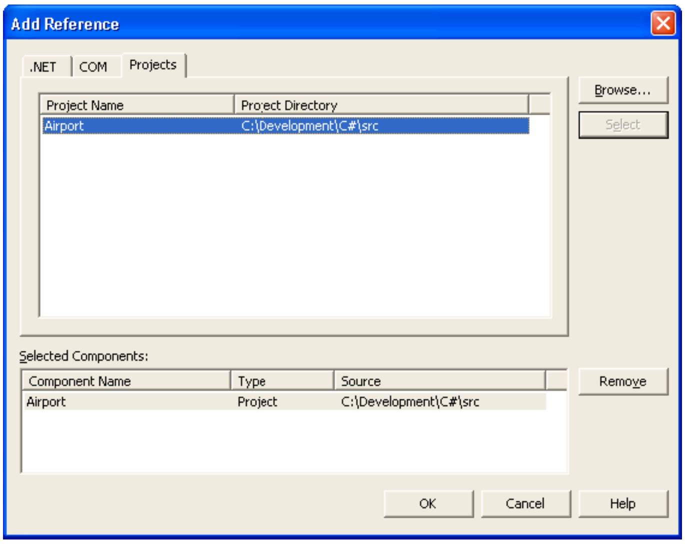


Figure 2.15 - Add project reference

11. Select Airport Project in **Projects** tab.

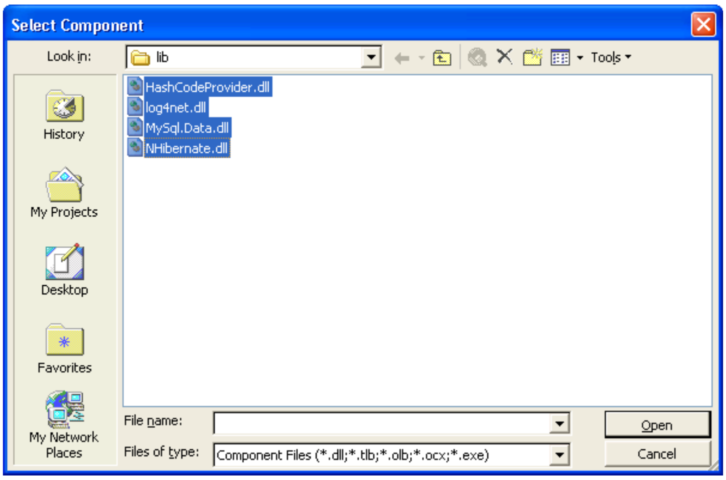


Project Name	Project Directory
Airport	C:\Development\C#\src

Component Name	Type	Source
Airport	Project	C:\Development\C#\src

Figure 2.16 - Add Reference dialog

12. Select **.NET** tab and add all the libraries (**HashCodeProvider.dll, log4net.dll, MySQL.Data.dll, NHibernate.dll**) in Airport project's **lib** folder.

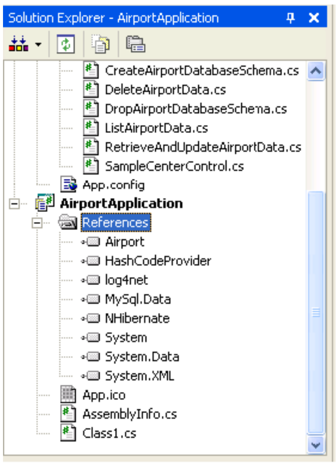


HashCodeProvider.dll
log4net.dll
MySql.Data.dll
NHibernate.dll

Files of type: Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe)

Figure 2.17 - Select the Component

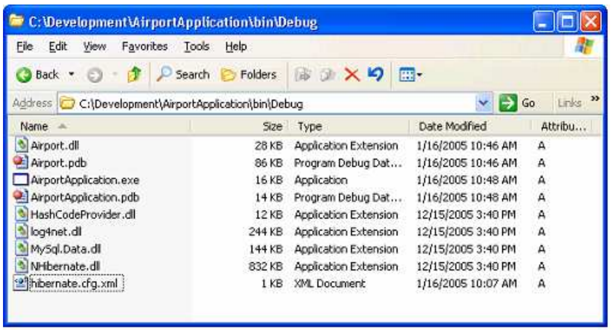
13. The Airport project and libraries are added to the AirportApplication project's Reference. You can develop application to call C# persistent classes in the AirportApplication project.



```
CreateAirportDatabaseSchema.cs
DeleteAirportData.cs
DropAirportDatabaseSchema.cs
ListAirportData.cs
RetrieveAndUpdateAirportData.cs
SampleCenterControl.cs
App.config
AirportApplication
  References
    Airport
    HashCodeProvider
    log4net
    MySql.Data
    NHibernate
    System
    System.Data
    System.XML
  App.ico
  AssemblyInfo.cs
  Class1.cs
```

Figure 2.18 - The reference added

14. Copy the **hibernate.cfg.xml** file from **Airport project\src** to **AirportApplication project\bin\Debug**. The **hibernate.cfg.xml** contains the information of database connections and other settings.



Address: C:\Development\AirportApplication\bin\Debug

Name	Size	Type	Date Modified	Attribu...
Airport.dll	28 KB	Application Extension	1/16/2005 10:46 AM	A
Airport.pdb	86 KB	Program Debug Dat...	1/16/2005 10:46 AM	A
AirportApplication.exe	16 KB	Application	1/16/2005 10:48 AM	A
AirportApplication.pdb	14 KB	Program Debug Dat...	1/16/2005 10:48 AM	A
HashCodeProvider.dll	12 KB	Application Extension	12/15/2005 3:40 PM	A
log4net.dll	244 KB	Application Extension	12/15/2005 3:40 PM	A
MySql.Data.dll	144 KB	Application Extension	12/15/2005 3:40 PM	A
NHibernate.dll	832 KB	Application Extension	12/15/2005 3:40 PM	A
hibernate.cfg.xml	1 KB	XML Document	1/16/2005 10:07 AM	A

Figure 2.19 - Copy the hibernate.cfg.xml to debug folder

Testing the Visual Studio .NET Project

You have created the AirportApplication project and referenced the generated Airport project. You can develop a simple program to test the project.

1. Open the Class1.cs file in AirportApplication.
2. Modify the Class1.cs file.

```
using System;
using airport;
using Orm;

namespace AirportApplication
{
        /// <summary>
        /// Summary description for Class1.
        /// </summary>
        class Class1
        {
                /// <summary>
                /// The main entry point for the application.
                /// </summary>
                [STAThread]
                static void Main(string[] args)
                {
                        PersistentTransaction t =
                        airport.AirportPersistentManager.Instance().GetSession().BeginTrans
                        action();
                        try
                        {
                                airport.Flight lairportFlight =
                                airport.FlightFactory.CreateFlight();
                                // Initialize the properties of the persistent object
                                lairportFlight.ArrivingAirport = "Hong Kong International
                                Airport";
                                lairportFlight.DepartingAirport = "Kansai International
                                Airport";
                                lairportFlight.DepartureTime = DateTime.Now;

                                airport.Plane lairportPlane =
                                airport.PlaneFactory.CreatePlane();
                                // Initialize the properties of the persistent object
                                lairportPlane.PlaneType = "747 plane";
                                lairportPlane.MaxSpeed = 967;
                                lairportPlane.MaxDistance = 8232;
                                lairportPlane.flights.Add(lairportFlight);
                                lairportPlane.Save();
                                // lairportPlane.Save();
                                t.Commit();
                        }
                        catch (Exception e)
                        {
                                t.RollBack();
                                Console.WriteLine(e);
                        }
                }
        }
}
```

3. From the menu bar, select **Build > Rebuild Solution**.

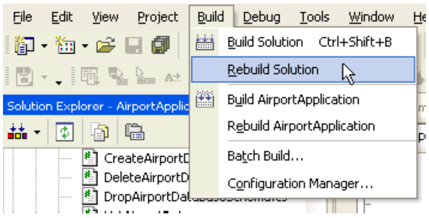


Figure 2.20 - To rebuild solution

4. Select **Debug > Start Without Debugging** to execute Class1.cs.

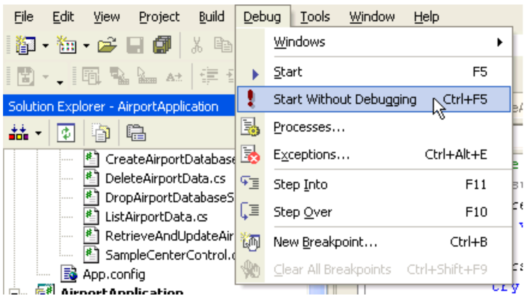
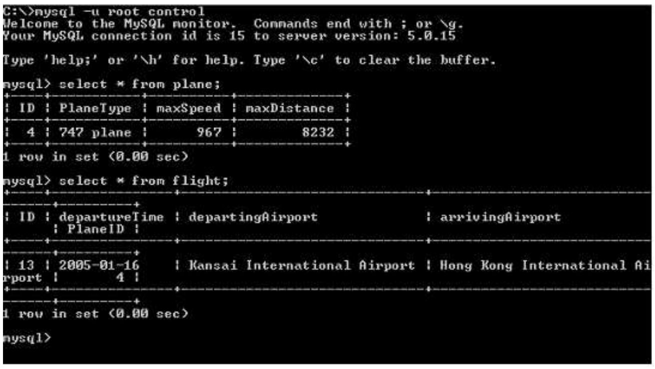


Figure 2.21 - To start without debugging

5. Check the MySQL database. The record is created.



```
C:\>mysql -u root control
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 15 to server version: 5.0.15

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> select * from plane;
+----+-----------+----------+-------------+
| ID | PlaneType | maxSpeed | maxDistance |
+----+-----------+----------+-------------+
|  4 | 747 plane |      967 |        8232 |
+----+-----------+----------+-------------+
1 row in set (0.00 sec)

mysql> select * from flight;
+----+---------------+------------------------------+---------------------------
-------+---------+
| ID | departureTime | departingAirport             | arrivingAirport
       | PlaneID |
+----+---------------+------------------------------+---------------------------
-------+---------+
| 13 | 2005-01-16    | Kansai International Airport | Hong Kong International Ai
rport  |       4 |
+----+---------------+------------------------------+---------------------------
-------+---------+
1 row in set (0.00 sec)

mysql>
```

Figure 2.22 - The record is created in the database