

8

Programming in C++.NET

Chapter 8 - Programming in C++ .NET

DB Visual ARCHITECT (DB-VA) can generate C#.NET source code so you can implement your application by C# programming language directly but you can also choose another language (VB.NET or C++) based on your taste in the .NET Framework. DB-VA generates DLL file and persistent libraries that can be referenced by .NET projects of language other than C#.

In this chapter:

- Introduction
- Generating DLL file
- Creating C++ Project
- Adding Referenced Project
- Working with the Generating Code and Persistent Library
- Running the Application

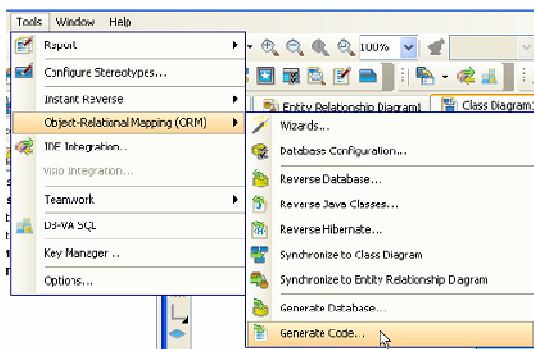
Introduction

C++ is an Object-Oriented Programming (OOP) language that is viewed by many as the best language for creating large-scale applications. The .NET Framework contains a virtual machine called Common Intermediate Language (CIL). Simply put, programs are compiled to produce CIL and the CIL is distributed to user to run on a virtual machine. C++, VB.NET, C# compilers are available from Microsoft for creating CIL. In DB-VA you can generate C# persistent source code and DLL file, so you can reference the DLL file and persistent library in Visual Studio .NET 2003 and develop the C++ application.

In the following section, you will develop a C++ application. The application is exactly same as the one in Chapter 4 – Developing Standalone .NET Application sample, but this time you use C++ instead of C# for development. You need to download the Chapter 4 sample application because it contains DLL file and persistent libraries for your C++ project.

Generating DLL File

1. From the menu bar, select **Tools > Object-Relational Mapping (ORM) > Generate Database...** to open the Database Code Generate dialog box.



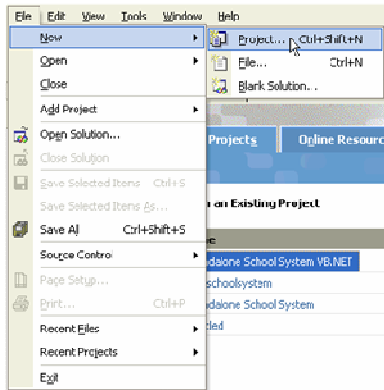
2. Check the **Compile to DLL** option to generate the DLL file.
 - **Compile to DLL**

By checking this option, DB-VA will generate DLL files which can be referenced by .NET projects of language other than C#. DB-VA generates batch file for the DLL file at the same time. You can modify the configuration file (hibernate.cfg.xml) manually and use the batch file to recompile and build an up-to-date DLL file for referenced project.

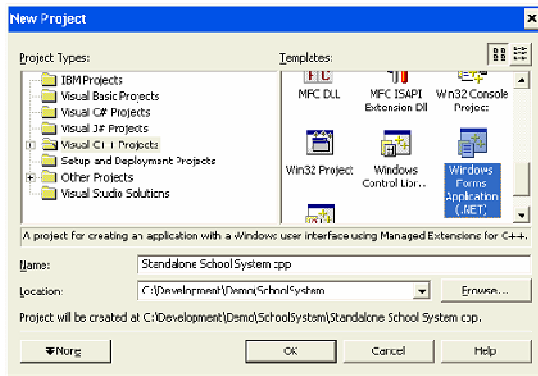


Creating C++ Project

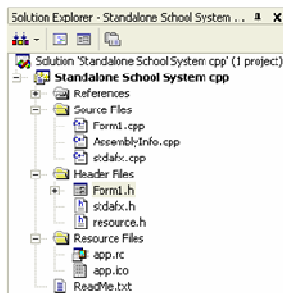
1. Open Microsoft Visual Studio .NET 2003.
2. Select **File > New > Project ...** from the menu.



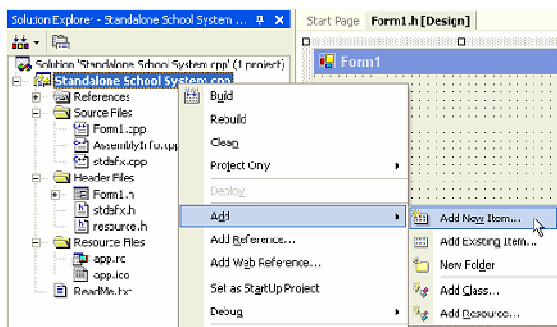
3. Select Project Types as Visual C++ Projects and Templates as Windows Form Application (.NET) and Location for the new application.



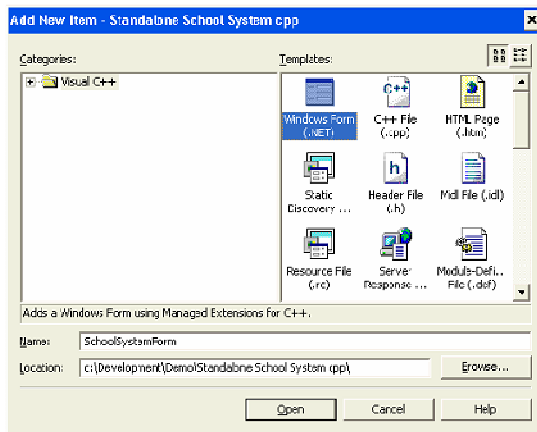
4. The School System Project is created.



5. Right click Standalone School System.cpp Project, select **Add > Add New Item...** from the popup menu.



6. Select Category as Visual C++, Template Windows Form (.NET) and enter the name for the form called "SchoolSystemForm". This is the start point for the application.



7. Append the following content to the SchoolSystemForm.cpp file (Source files/SchoolSystemForm.cpp). This is the main method for C++ Application to execute.

```
#include <windows.h>

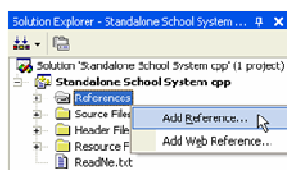
using namespace StandaloneSchoolSystemcpp;

int APIENTRY _tWinMain(HINSTANCE hInstance,
                      HINSTANCE hPrevInstance,
                      LPTSTR lpCmdLine,
                      int nCmdShow)
{
    System::Threading::Thread::CurrentThread->ApartmentState =
    System::Threading::ApartmentState::STA;
    Application::Run(new SchoolSystemForm());
    return 0;
}
```

8. Remove the Form1.cpp and Form1.h files.

Adding Referenced Project

1. Right click References under the Standalone School System C++ project and select **Add Reference...**. Reference the C# example DDL file and persistent libraries for developing the C++ application.



3. After that, the system creates the new Student Persistent object.

```
private: System::Void okButton_Click(System::Object *sender,
System::EventArgs *e)
{
    if(loginIDTextBox->Text->Length == 0 || passwordTextBox->Text-
>Length == 0){
        MessageBox::Show("Login id or password missing");
        return;
    }
    PersistentTransaction *t =
SchoolSystemPersistentManager::Instance()->GetSession()-
>BeginTransaction();
    try{
        if(userType == CREATE_TEACHER){
            createdUser = TeacherFactory::CreateTeacher();
        }else{
            createdUser = StudentFactory::CreateStudent();
        }
    }
}
```

Source File : Standalone School System cpp \RegisterDialog.h

4. Set the student information from the text fields value to the Student Object

```
createdUser->set_Name(userNameTextBox->Text);
createdUser->set_Password(passwordTextBox->Text);
createdUser->set_LoginID(loginIDTextBox->Text);
if(dynamic_cast<Teacher*>(createdUser)){
    dynamic_cast<Teacher*>(createdUser)->set_Email(emailTextBox->Text);
}
}
```

Source File : Standalone School System cpp \RegisterDialog.h

5. Call Save() method of Student Persistent Object and Commit() method of PersistentTransaction, then the new Student object will be saved in database. If there are any errors occurred during the transaction, you can call the Rollback() method to cancels the proposed changes in a pending database transaction

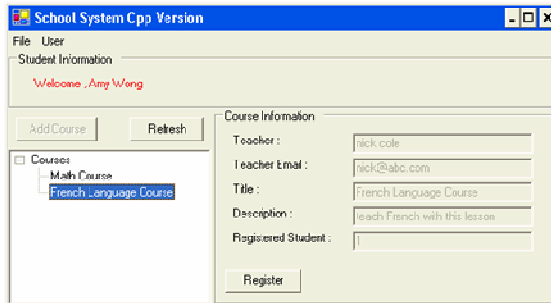
```
createdUser->Save();
this->set_CreatedUser(createdUser);
DialogResult = DialogResult::OK;
t->Commit();
Close();
}catch(Exception *ex){
    Console::WriteLine(ex->InnerException->Message);
    DialogResult = DialogResult::Cancel;
    t->RollBack();
}
```

Source File : Standalone School System cpp \RegisterDialog.h

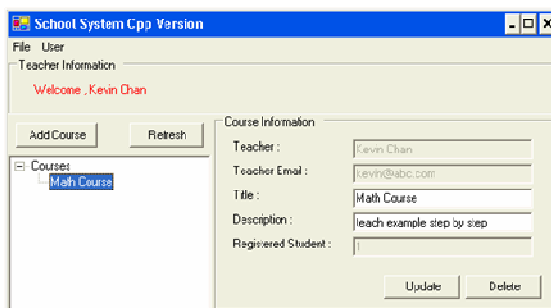
- Querying Object from Database

After the user login the School System, the system queries different Course objects from the database according to user role. If the user is a student, the system shows all the available courses. The student can select and register the course. If the user is teacher, the system shows the courses that are created by that teacher. The teacher can update or delete the course information in system.

Student Login:



Teacher Login:



1. Query the course objects when user login. When Student login, the system will call **ListCourseByQuery()** method in **CourseFactory** to get all available courses. When Teacher login, the system will call **courses** collection variable in Teacher object.

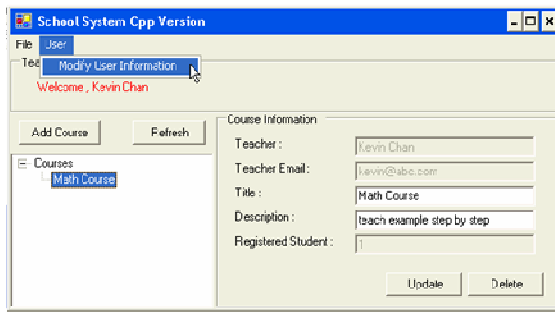
```
void updateTreeView(void) {
    Course *courses[];
    if(dynamic_cast<Student*>(currentUser)) {
        courses = CourseFactory::ListCourseByQuery(NULL, NULL);
    } else {
        courses = dynamic_cast<Teacher*>(currentUser)->courses-
>ToArray();
    }
    ...
}
```

Source File : Standalone School System cpp \SchoolSystemForm.h

- Updating Object and Saving to Database

You can modify the user information and update the record in database. After the user login, the User object is stored in the application, so you can set new information in the user object and update the database record.

1. From the menu bar, select **User > Modify User Information** to open the Modify User Information dialog box.



2. Enter new user information and click **OK** to update the User record.



3. Update the information for the User object includes password, name and email address.

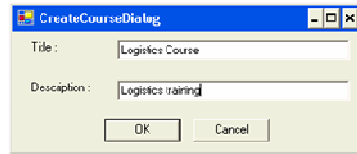
```
private: System::Void okButton_Click(System::Object *sender,
System::EventArgs *e)
{
    if(nameTextBox->Text->Length == 0 || passwordTextBox->Text->Length ==
0){
        MessageBox::Show("Missing name or password");
        return;
    }else{
        PersistentTransaction *t =
SchoolSystemPersistentManager::Instance()->GetSession()-
>BeginTransaction();
        try{
            user->set_Name(nameTextBox->Text);
            user->set_Password(passwordTextBox->Text);
            if(dynamic_cast<Teacher*>(user)){
                (dynamic_cast<Teacher*>(user))->set_Email(emailTextBox-
>Text);
            }
            user->Save();
            DialogResult = DialogResult::OK;
            t->Commit();
        }catch(Exception *ex){
            DialogResult = DialogResult::Cancel;
            t->RollBack();
        }
    }
}
```

Source File : Standalone School System cpp\ModifyUserDialog.h

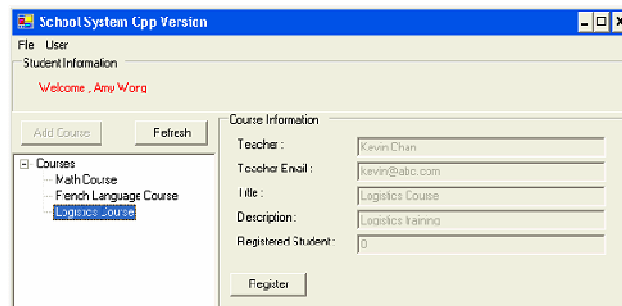
- Deleting Object in Database

Teacher can create courses for students to register and they can cancel the course in the system. They only need to click delete button of the course then the course information will be deleted in the database and all its relationships with register students will be removed.

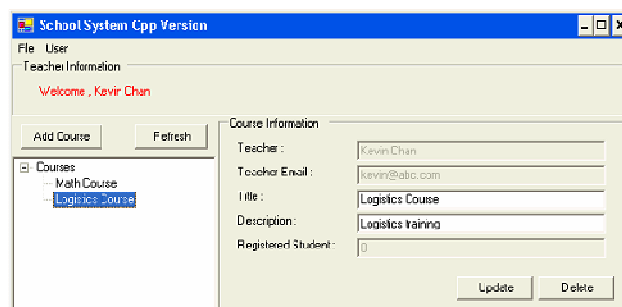
1. Teacher can create the course by clicking the **Add Course** button, fill in Course name and Description.



2. Student can register the course by clicking the **Register** button.



3. The teacher can view how many students registered his course, and he can delete the course in the system.



4. Click Delete of a Course then it will trigger the **deleteButton_Click()** method.

```
private: System::Void deleteButton_Click(System::Object *sender,
System::EventArgs *e)
{
    if (MessageBox::Show("Delete", "Delete", MessageBoxButtons::OKCancel)
    == DialogResult::OK) {
```

Source File : Standalone School System cpp\SchoolSystemform.h

5. Call **deleteAndDissociate()** method to delete the course object and remove the relationships of student and teacher with it.

```

        PersistentTransaction *t =
SchoolSystemPersistentManager::Instance()->GetSession()-
>BeginTransaction();

        try{
            selectedNode->get_Course()->DeleteAndDissociate();
            selectedNode->Remove();
            t->Commit();
        }catch(Exception *ex){
            Console::WriteLine(ex->InnerException->Message);
            t->RollBack();
        }
    }
}

```

Source File : Standalone School System cpp\SchoolSystemForm.h

Running the Application

To execute the C++ application, select **Debug > Start (F5)** on the menu bar Visual Studio .NET 2003.

