# 此示例演示如何对一个 Student 对象列表执行简单查询。

每个 Student 对象都包含一些有关该学生的基本信息，以及一个表示该学生的四次考试得分的列表。

此应用程序充当此部分中其他很多示例的框架，这些示例都使用相同的 students 数据源。

下面的查询返回那些在其第一次考试中得分为 90 分或更高的学生。

public class StudentClass

{

#region data

protected enum GradeLevel { FirstYear = 1, SecondYear, ThirdYear, FourthYear };

protected class Student

{

public string FirstName { get; set; }

public string LastName { get; set; }

public int ID { get; set; }

public GradeLevel Year;

public List<int> ExamScores;

}

protected static List<Student> students = new List<Student>

{

new Student {FirstName = "Terry", LastName = "Adams", ID = 120,

Year = GradeLevel.SecondYear,

ExamScores = new List<int>{ 99, 82, 81, 79}},

new Student {FirstName = "Fadi", LastName = "Fakhouri", ID = 116,

Year = GradeLevel.ThirdYear,

ExamScores = new List<int>{ 99, 86, 90, 94}},

new Student {FirstName = "Hanying", LastName = "Feng", ID = 117,

Year = GradeLevel.FirstYear,

ExamScores = new List<int>{ 93, 92, 80, 87}},

new Student {FirstName = "Cesar", LastName = "Garcia", ID = 114,

Year = GradeLevel.FourthYear,

ExamScores = new List<int>{ 97, 89, 85, 82}},

new Student {FirstName = "Debra", LastName = "Garcia", ID = 115,

Year = GradeLevel.ThirdYear,

ExamScores = new List<int>{ 35, 72, 91, 70}},

new Student {FirstName = "Hugo", LastName = "Garcia", ID = 118,

Year = GradeLevel.SecondYear,

ExamScores = new List<int>{ 92, 90, 83, 78}},

new Student {FirstName = "Sven", LastName = "Mortensen", ID = 113,

Year = GradeLevel.FirstYear,

ExamScores = new List<int>{ 88, 94, 65, 91}},

new Student {FirstName = "Claire", LastName = "O'Donnell", ID = 112,

Year = GradeLevel.FourthYear,

ExamScores = new List<int>{ 75, 84, 91, 39}},

new Student {FirstName = "Svetlana", LastName = "Omelchenko", ID = 111,

Year = GradeLevel.SecondYear,

ExamScores = new List<int>{ 97, 92, 81, 60}},

new Student {FirstName = "Lance", LastName = "Tucker", ID = 119,

Year = GradeLevel.ThirdYear,

ExamScores = new List<int>{ 68, 79, 88, 92}},

new Student {FirstName = "Michael", LastName = "Tucker", ID = 122,

Year = GradeLevel.FirstYear,

ExamScores = new List<int>{ 94, 92, 91, 91}},

new Student {FirstName = "Eugene", LastName = "Zabokritski", ID = 121,

Year = GradeLevel.FourthYear,

ExamScores = new List<int>{ 96, 85, 91, 60}}

};

#endregion

//Helper method, used in GroupByRange.

protected static int GetPercentile(Student s)

{

double avg = s.ExamScores.Average();

return avg > 0 ? (int)avg / 10 : 0;

}

public void QueryHighScores(int exam, int score)

{

var highScores = from student in students

where student.ExamScores[exam] > score

select new {Name = student.FirstName, Score = student.ExamScores[exam]};

foreach (var item in highScores)

{

Console.WriteLine("{0,-15}{1}", item.Name, item.Score);

}

}

}

public class Program

{

public static void Main()

{

StudentClass sc = new StudentClass();

sc.QueryHighScores(1, 90);

// Keep the console window open in debug mode.

Console.WriteLine("Press any key to exit");

Console.ReadKey();

}

}

我们特意将此查询编写得非常简单，以使您能够进行试验。例如，您可以在 **where** 子句中尝试使用更多的谓词，也可以使用 **orderby** 子句对结果进行排序。

**编译代码**

* 创建面向 .NET Framework 3.5 版的 Visual Studio 项目。默认情况下，该项目具有一个对 System.Core.dll 的引用以及一条针对 System.Linq 命名空间的 **using** 指令。
* 将代码复制到项目中。
* 按 F5 编译并运行程序。
* 按任意键退出控制台窗口。