[**Linq(筛选与排序)**](http://www.cnblogs.com/xfrog/archive/2010/09/11/1824086.html)

本文将讲述LINQ的基础查询(此文所有例子都使用LINQ to Object)

在此之前，我们先创建一个用于示例的数据源：

* + Student类：表示学生，包括学号、姓名及班级
* Courses类：表示学生选择的课程，包括学号、课程名称及学时数
* DataCreator类：静态类，通过GenerateData方法产生示例数据
* 生成的数据如下：

学号    姓名    班级    课程名称    学时   
003        王五    二班    经济学    20   
003        王五    二班    企业管理    20   
003        王五    二班    财务管理    30   
002        李四    一班    历史    20   
002        李四    一班    政治    20   
002        李四    一班    语文    30   
001        张三    一班    数学    20   
001        张三    一班    语文    20   
001        张三    一班    物理    15

Main方法中生成数据：

隐藏行号 复制代码 ？C#

* + List<Student> students = new List<Student>();
  + List<Courses> courses = new List<Courses>();
  + DataCreator.GenerateData(students, courses);

**筛选**

    LINQ中通过where子句筛选数据，where子句后跟条件，from后的对象都可作为where子句判断条件的主体。

* 实际上from .. in ..语句类似于foreach语句，比如
* from student in students 类似于 foreach(Student student in students)
* student是students集合中的元素，那么通常在where语句中，可以使用student作为判断的依据，例如：
* 筛选出一班的所有学生：
* 隐藏行号 复制代码 ？C#
  + var query = from student in students
  + where student.Class == "一班"
  + select student;
  + foreach (var s in query)
  + {
  + Console.WriteLine(s);
  + }

    筛选学时大于等于20的课程：

* 隐藏行号 复制代码 ？C#
  + var query = from course in courses
  + where course.Credit >= 20
  + select course;

     可使用&&或||来连接多个条件，如筛选出一班姓张的同学：

* 隐藏行号 复制代码 ？C#
  + var query = from student in students
  + where student.Class == "一班" && student.Name.StartsWith("张")
  + select student;

     当然，并不是说条件语句中只能使用from后的对象，它可以使用任何对象作为判断依据(下面LINQ只作演示没实际意义)：条件当前时间等于当前时间，此条件永远为真，故会查询出所有的学生资料。

* 隐藏行号 复制代码 ？C#
  + var query = from student in students
  + where DateTime.Now == DateTime.Now
  + select student;
* 下例以students集合的数量作为判断条件，当stuents的数量大于等于3时筛选出所有学生资料：
* 隐藏行号 复制代码 ？C#
  + var query = from student in students
  + where students.Count>=3
  + select student;

     下例筛选学号存在于ids集合中的学生：

* 隐藏行号 复制代码 ？C#
  + String[] ids = {"001","003" };
  + var query = from student in students
  + where ids.Contains(student.ID)
  + select student;

     查询选修了语文课程的学生资料：

* 隐藏行号 复制代码 ？C#
  + var query = from student in students
  + where (from cours in courses where cours.Name == "语文" select cours.StudentId).Contains(student.ID)
  + select student;
  + foreach (var s in query)
  + {
  + Console.WriteLine(s);
  + }

     注意上例中使用了嵌套查询，students where语句中包含另一个LINQ语句，该语句从courses集合中返回选修了语文课程的学号，它是一个IEnumerable<String>集合，所以可以在此集合上使用IEnumerable的Contains方法，判断集合中是否包含当前student对象的ID，包含则返回true，即students的where语句为真，该student对象被添加入结果集合，反之不添加。



**排序**

    排序语句比较简单，基本语法为：

        orderby .. ascending | descending[, ..ascending | descending]

    其中ascending表示升序排列，descending表示降序排列

隐藏行号 复制代码 ？C#

1. var query = from student in students
2. orderby student.ID ascending /\*descending\*/
3. select student;

    可根据多个字段进行排序：

隐藏行号 复制代码 ？C#

1. var query = from student in students
2. orderby student.ID ascending,student.Name descending
3. select student;

待续…

    以下源码产生本文示例数据：

隐藏行号 复制代码 ？C#

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5. namespace LinqStudy
6. {
7. public static class DataCreator
8. {
9. public static void GenerateData(List<Student> students,List<Courses> courses)
10. {
11. students.Add(new Student("001","张三", "一班"));
12. students.Add(new Student("002", "李四", "一班"));
13. students.Add(new Student("003", "王五", "二班"));
14. courses.Add(new Courses("001", "数学", 20));
15. courses.Add(new Courses("001", "语文", 20));
16. courses.Add(new Courses("001", "物理", 15));
17. courses.Add(new Courses("002", "历史", 20));
18. courses.Add(new Courses("002", "政治", 20));
19. courses.Add(new Courses("002", "语文", 30));
20. courses.Add(new Courses("003", "经济学", 20));
21. courses.Add(new Courses("003", "企业管理", 20));
22. courses.Add(new Courses("003", "财务管理", 30));
23. }
24. }
25. public class Student
26. {
27. public String Name { get; set; }
28. public String Class { get; set; }
29. public String ID { get; set; }
30. public Student(String id,String name, String c)
31. {
32. ID = id;
33. Name = name;
34. Class = c;
35. }
36. public override string ToString()
37. {
38. return String.Format("学号:{0}\t姓名:{1}\t班级:{2}", this.ID, this.Name, this.Class);
39. }
40. }
41. public class Courses
42. {
43. public String Name { get; set; }
44. public int Credit { get; set; }
45. public String StudentId { get; set; }
46. public Courses(String id,String name, int credit)
47. {
48. StudentId = id;
49. Name = name;
50. Credit = credit;
51. }
52. public override string ToString()
53. {
54. return String.Format("课程名称：{0}\t学时:{1}", this.Name, this.Credit);
55. }
56. }
57. }

标签: [LINQ](http://www.cnblogs.com/xfrog/tag/LINQ/), [C#](http://www.cnblogs.com/xfrog/tag/C%23/)