**SEMINAR LINQ - .NET SE310.P12**

1. **Thành viên:**

* Trần Nhật Tân – 22521312
* Tô Vĩnh Tiến – 22521474

1. **Nội dung**
2. **LinQ là gì ?**

* LinQ là viết tắt của Language Integrated Query, là một cơ chế truy vấn được tích hợp trong ngôn ngữ lập trình C# và VB.NET của Microsoft. LinQ cung cấp một cách thức tập trung và linh hoạt để truy vấn dữ liệu từ nhiều nguồn khác nhau, bao gồm các tập tin, cơ sở dữ liệu, tài liệu XML và bộ nhớ.
* LinQ cho phép lập trình viên sử dụng một chuỗi truy vấn đơn giản để truy vấn và sắp xếp dữ liệu một cách linh hoạt, hiệu quả và dễ dàng hơn. Bằng cách sử dụng LinQ, các lập trình viên không cần phải biết rõ các ngôn ngữ truy vấn cụ thể như SQL, mà chỉ cần sử dụng các phương thức và thuộc tính LinQ để truy vấn dữ liệu.
* LinQ cung cấp các phương thức truy vấn thông dụng như Select, Where, OrderBy, Join, GroupBy và Count, cho phép lập trình viên thực hiện các truy vấn phức tạp một cách dễ dàng. Bên cạnh đó, LinQ cũng hỗ trợ các kiểu dữ liệu tự định nghĩa và các toán tử mở rộng, cho phép lập trình viên tạo ra các truy vấn truy vấn tùy chỉnh của riêng mình.
* Với các tính năng và ưu điểm vượt trội của nó, LinQ đã trở thành một công cụ quan trọng và phổ biến trong việc phát triển ứng dụng .NET.

1. **Kiến trúc và các loại LinQ (LinQ to Objects, LinQ to SQL, LinQ to Entities, LinQ to XML, LinQ to DataSet)**
   1. **LinQ to Objects**

* LinQ to Objects trả về các biến kiểu IEnumerable<T> và không cần LinQ provider (API) như LinQ to SQL hay LinQ to XML. LinQ to Objects thay thế các đoạn mã lặp bằng các câu truy vấn ngắn gọn, dễ đọc, hiệu quả. Có thể dễ dàng chuyển đổi giữa các nguồn dữ liệu mà không cần chỉnh sửa hay chỉnh sửa rất ít.
  1. **LinQ to SQL**
* Là một thành phần dùng để quản lý các dữ liệu quan hệ như đối tượng.
* Trong LinQ to SQL, mô hình dữ liệu của cơ sở dữ liệu quan hệ được ánh xạ đến mô hình đối tượng thể hiện trong ngôn ngữ lập trình của nhà phát triển. Khi ứng dụng thực thi, LinQ to SQL chuyển thành các truy vấn SQL trong mô hình đối tượng và đượcgửi đến cơ sở dữ liệu để thực thi. Khi cơ sở dữ liệu trả về kết quả, LinQ to SQL sẽ chuyển các kết quả này thành đối tượng để chúng ta có thể làm việc.
* Các luật thực thi truy vấn với LinQ to SQL giống như thực thi các truy vấn LinQ chuẩn. Các thành phần dùng trong thực thi truy vấn với LinQ to SQL gồm:
  + LinQ to SQL API: yêu cầu thực thi truy vấn trên ứng dụng và gửi đến LinQ to SQL Provider.
  + LinQ to SQL Provider: chuyển truy vấn đến T-SQL và gửi truy vấn mới đến ADO Provider để thực thi.
  + ADO Provider: sau khi thực thi truy vấn, kết quả trong hình thức của DataReader được gửi đến LINQ to SQL Provider để chuyển sang hình thức đối tượng người dùng.
  1. **LinQ to Entities**
* Entity Framework (EF – còn được gọi là OR/M Framework) làm việc ở mức trừu tượng cao hơn LinQ to SQL. Khi làm việc với EF một nhà phát triển phải tạo ra một lược đồ khái niệm (conceptual schema), một lược đồ lưu trữ (store schema), và một ánh xạ (mapping) giữa hai lược đồ này. Trong LinQ to SQL mỗi lớp sẽ ánh xạ đến chính xác một bảng, trong EF mỗi lớp có thề ánh xạ đến nhiều bảng.
* LinQ to Entities cho phép các nhà phát triển truy vấn đến mô hình dữ liệu khái niệm và các truy vấn là các truy vấn LinQ chuẩn.
  1. **LinQ to Dataset**
* DataSet là một hình thức bộ nhớ trong của dữ liệu quan hệ và từ lâu đã là một phần chủ yếu của việc truy cập dữ liệu trong thế giới .NET. LinQ to Dataset cho phép sử dụng các toán tử truy vấn chuẩn để truy vấn đến dữ liệu trong DataTable hay DataSet.
* Trước khi bắt đầu truy vấn một DataSet dùng LinQ to Dataset, Dataset cần được nạp dữ liệu bằng cách dùng lớp DataAdapter hay dùng LinQ to SQL. LINQ to Dataset làm việc với Dataset có kiểu (typed Dataset) và không kiểu (untyped Dataset). Với Dataset không kiểu, LinQ to Dataset cung cấp các phương thức mở rộng như Field<T> cho phép gán kiểu vào truy vấn.
  1. **LinQ to XML**
* LinQ to XML có thể dễ dàng truy cập đến các tính năng của LinQ như các toán tử LINQ, giao diện lập trình, v.v. Và do tích hợp trong .NET Framework nên LinQ to XML có thể sử dụng các tính năng mạnh mẽ của nó như debugging, compile time checking, strong typing, v.v.
* Với LinQ to XML, việc nạp tài liệu XML vào bộ nhớ, việc đọc hay ghi tài liệu XML đến ổ đĩa hay bộ nhớ đều được thực hiện rất dễ dàng. Đồng thời với LinQ to XML, các nhà phát triển sẽ tiết kiệm nhiều công sức trong việc học và sử dụng các ngôn ngữ truy vấn XML phức tạp.
* Sức mạnh của LinQ to XML chứa trong namespace System.Xml.LinQ với 19 lớp cơ bản sau:
  + XAttribute - Đại diện cho một thuộc tính của phần tử XML.
  + XCData - Đại diện cho dữ liệu kiểu CDATA (Character Data).
  + XComment - Đại diện cho một comment (ghi chú) trong XML.
  + XContainer - Lớp cơ sở cho các phần tử có thể chứa các nút con (sub-nodes).
  + XDeclaration - Đại diện cho khai báo XML, ví dụ <?xml version="1.0" encoding="utf-8"?>.
  + XDocument - Đại diện cho toàn bộ tài liệu XML.
  + XDocumentType - Đại diện cho DOCTYPE của tài liệu XML.
  + XElement - Đại diện cho một phần tử XML.
  + XName - Đại diện cho tên của một phần tử hoặc thuộc tính XML.
  + XNamespace - Đại diện cho một namespace XML.
  + XNode - Lớp cơ sở cho tất cả các nút trong cây XML.
  + XNodeDocumentOrderComparer - Hỗ trợ so sánh thứ tự tài liệu của các nút XML.
  + XNodeEqualityComparer - Hỗ trợ so sánh các nút XML để kiểm tra sự tương đương.
  + XObject - Lớp cơ sở cho các đối tượng XNode và XAttribute.
  + XObjectChange - Đại diện cho các loại thay đổi xảy ra đối với XObject (thêm, xóa, thay đổi giá trị...).
  + XObjectChangeEventArgs - Cung cấp dữ liệu cho sự kiện thay đổi trên XObject.
  + XObjectEventHandler - Đại diện cho phương thức xử lý sự kiện khi có thay đổi trong XObject.
  + XProcessingInstruction - Đại diện cho một hướng dẫn xử lý XML, ví dụ <?xml-stylesheet ...?>.
  + XText - Đại diện cho nội dung văn bản trong XML.

1. **Cú pháp LinQ (Query Syntax, Method Syntax) và so sánh sự khác biệt giữa 2 cú pháp**
   1. Query Syntax
      * Query Syntax: (cú pháp truy vấn) trong LinQ có cú pháp giống với SQL và dễ hiểu đối với những người đã biết qua cũng như quen thuộc với ngôn ngữ truy vấn dữ liệu SQL(Structured Query Language).
      * Query Syntax dùng các từ khoá quen thuộc như from, where, select, group, order by, join,… tương tự như SQL
      * Ví dụ về 1 Query Syntax trong LinQ:

A screenshot of a computer

Description automatically generated

* ở đây from đại diện cho nguồn dữ liệu mà ta muốn truy vấn, tương tự như từ khoá SELECT trong SQL
* where dùng để lọc dựa trên điều kiện, mà điều kiện ở đây chính là học sinh có tuổi từ 18 trở lên
* orderby dùng để sắp xếp các kết quả trả về và select dùng để chọn các giá trị cuối cùng.
  + 1. **Ưu điểm**
       - Cú pháp dễ hiểu, trực quan với người đã quen với SQL.
       - Phù hợp với các truy vấn đơn giản, hoặc truy vấn lọc/sắp xếp dữ liệu.
    2. **Hạn chế**
       - Khả năng mở rộng hạn chế và ít linh hoạt hơn so với Method Syntax khi làm việc với các thao tác phức tạp.
       - Khi sử dụng với các phương thức mở rộng như Sum, Max, hoặc các toán tử tổng hợp, cần kết hợp thêm cú pháp Method Syntax.

1. **Method Syntax**
   * + - Method Syntax sử dụng các phương thức mở rộng (extension methods) trong .NET như **Where**, **Select**, **OrderBy**, **GroupBy**, **Join**, v.v., trên các collection hoặc đối tượng có kiểu dữ liệu hỗ trợ LINQ.
       - Ví dụ về 1 Method Syntax

A screen shot of a computer code

Description automatically generated

1. **Ưu điểm**

* Linh hoạt và dễ dàng mở rộng khi kết hợp với các phương thức mở rộng khác, cho phép xây dựng các truy vấn phức tạp hơn.
* Method Syntax cho phép xử lý các thao tác mà Query Syntax không hỗ trợ trực tiếp, ví dụ như Sum, Count, Average.
* Dễ dàng tích hợp và sử dụng với các biểu thức lambda, giúp mã ngắn gọn hơn khi cần thực hiện nhiều thao tác.

1. **Hạn chế**

* Đối với những người mới, cú pháp Method Syntax có thể hơi khó hiểu do cách viết sử dụng nhiều phương thức mở rộng và lambda expressions.
* Trong các truy vấn dài và phức tạp, Method Syntax có thể khó đọc và khó bảo trì hơn so với Query Syntax.

1. **So sánh Query Syntax và Method Syntax**

|  |  |  |
| --- | --- | --- |
| **Đặc điểm** | **Query Syntax** | **Method Syntax** |
| Cú pháp | Giống với SQL | Sử dụng các phương thức mở rộng và lambda expressions |
| Dễ hiểu với người dùng SQL | Rất dễ hiểu | Có thể khó hiểu hơn cho người mới |
| Khả năng mở rộng | Hạn chế trong các truy vấn phức tạp | Linh hoạt hơn, dễ mở rộng với nhiều toán tử khác nhau |
| Khả năng kết hợp với các toán tử | Hạn chế trong một số toán tử tổng hợp | Dễ dàng kết hợp với các toán tử như Sum, Count, Max |
| Độ dài code | Có thể ngắn gọn hơn trong truy vấn đơn giản | Có thể dài hơn do sử dụng nhiều phương thức |

1. **Khi nào nên sử dụng Query, khi nào nên sử dụng Method?**

* **Query**
* Thích hợp cho các truy vấn đơn giản, dễ đọc, chủ yếu là lọc và sắp xếp dữ liệu.
* Phù hợp nếu bạn quen với cú pháp SQL và muốn viết mã dễ hiểu hơn.
* **Method**
* Phù hợp cho các truy vấn phức tạp, cần nhiều thao tác lọc, tổng hợp, hoặc các thao tác nâng cao.
* Được ưa chuộng hơn khi cần tận dụng các phương thức mở rộng và biểu thức lambda để viết mã ngắn gọn, linh hoạt hơn.

1. **Các toán tử của LinQ (Where, OrderBy, ...)**

* Toán tử LinQ là các phương thức hỗ trợ việc truy vấn và thao tác trên dữ liệu trong LinQ. Có nhiều toán tử LinQ khác nhau phục vụ các mục đích như lọc, sắp xếp, nhóm, và tổng hợp dữ liệu.
  1. **Toán tử lọc**
     + **Where**: Lọc các phần tử dựa trên một điều kiện nhất định
     + **Ví dụ**: students.Where(student => student.Age >= 18);
  2. **Toán tử sắp xếp**
* **OrderBy**: Sắp xếp các phần tử theo thứ tự tăng dần
* **Ví dụ**: students.OrderBy(student => student.Name);
  1. **Toán tử chọn**
* **Select**: Chọn và chuyển đổi dữ liệu từ tập hợp ban đầu
* **Ví dụ**: students.Select(student => student.Name)
  1. **Toán tử nhóm**
* GroupBy: Nhóm các phần tử theo một khóa xác định
* Ví dụ: students.GroupBy(student => student.Class)

1. **Ưu/ nhược điểm của LinQ, lợi ích và thách thức khi sử dung LinQ**
   1. **Ưu điểm**
      * **Mã ngắn gọn và dễ hiểu hơn**: LinQ giúp giảm bớt lượng mã viết ra so với các truy vấn truyền thống như SQL hoặc vòng lặp phức tạp, làm cho mã dễ đọc và dễ bảo trì hơn.
      * **Tính an toàn của mã (Type-Safe)**: LinQ kiểm tra tính hợp lệ của truy vấn ngay tại thời điểm biên dịch, giúp phát hiện lỗi sớm.
      * **Khả năng tái sử dụng mã cao**: Vì LinQ cho phép viết các truy vấn ngắn gọn và dễ đọc, mã truy vấn có thể được tái sử dụng và dễ dàng bảo trì.
   2. **Nhược điểm**
      * **Hiệu suất thấp hơn so với SQL thuần túy**: LinQ có thể tạo ra các truy vấn phức tạp và không tối ưu, dẫn đến hiệu suất chậm hơn so với SQL viết tay.
      * **Khó gỡ lỗi**: Khi xảy ra lỗi trong truy vấn LinQ, việc gỡ lỗi có thể phức tạp và khó khăn hơn so với việc sử dụng truy vấn SQL truyền thống.
      * **Học và sử dụng**: LinQ yêu cầu lập trình viên nắm vững cú pháp và các toán tử của nó, điều này có thể là trở ngại cho người mới hoặc chưa quen với kiểu truy vấn mới.
2. **Ví dụ minh hoạ**