Entity Framework Core

Laboratorio di interfaccia uomo-macchina 2023-2024

Introduzione

Entity Framework è un object-relational mapper (ORM):

- Consente agli sviluppatori .NET di lavorare sui database utilizzando oggetti .NET
- Elimina la necessità di scrivere la maggior parte del codice di accesso ai dati che tipicamente deve essere scritto

Supporta, nativamente o tramite driver di terze parti, molti motori di database tra i quali: Sql Server, Sql Lite, Cosmos, PostgreSQL, MySQL, Oracle, Db2, Informix

Il modello

Con Entity Framework Core, l'accesso ai dati viene eseguito tramite un modello.

Un modello è costituito da classi di entità e da un contesto dell'oggetto che rappresenta una sessione con il database che consente di eseguire query e salvare i dati.

Esempio definizione modello

```
public class BloggingContext :DbContext {
  public DbSet<Blog> Blogs { get; set; }
  public DbSet<Post> Posts { get; set; }
}
public class Blog {
  public int BlogId { get; set; }
  public string Url { get; set; }
  public int Rating { get; set; }
  public List<Post> Posts { get; set; }
}
```

```
public class Post {
  public int PostId { get; set; }
  public string Title { get; set; }
  public string Content { get; set; }
  public int BlogId { get; set; }
  public Blog Blog { get; set; }
}
```

Configurazione modello

- Il modello che abbiamo visto contiene al suo interno tutte le informazioni necessarie per operare su un database che rispetta le convenzioni di nome standard di EF.
- E' però possibile personalizzare il modello arricchendolo di configurazioni personalizzati
- Le configurazioni possono essere descritte con due modalità:
  - Decorando con **attributi**
  - Utilizzando l'interfaccia **Fluent** nel metodo **OnModelCreating**

LinQ (Language-Integrated Query)

- Le interrogazioni di Entity Framework si esprimono in linguaggio LinQ
- LinQ è il nome di un set di tecnologie basate sull'integrazione delle funzionalità di query direttamente nel linguaggio C#
- Le espressioni di query possono essere usate per eseguire una query e trasformare dati da qualsiasi origine dati abilitata per LinQ, come ad esempio qualsiasi IEnumerable o IQuerable (Entity framework).
- Le variabili presenti in un'espressione di query sono tutte fortemente tipizzate, anche se in molti casi non è necessario specificare il tipo in modo esplicito perché il compilatore è in grado di dedurlo

LinQ

Sintassi

E’ possibile descrivere le query in due differenti formati

```
int[] numbers = { 5, 10, 8, 3, 6, 12};
```

- Sintassi a **Query**

```
IEnumerable<int> numQuery1 =
    from num in numbers
    where num % 2 == 0
    orderby num
    select num;
```

- Sintassi a **metodo (espressioni lambda)**

```
IEnumerable<int> numQuery2 = numbers
    .Where(num => num % 2 == 0)
   .OrderBy(n => n);
```

LinQ

Operatori

Clause	Description
from	Specifies a data source and a range variable (similar to an iteration variable).
where	Filters source elements based on one or more Boolean expressions separated by logical AND and OR operators (&& or ||).
select	Specifies the type and shape that the elements in the returned sequence will have when the query is executed.
group	Groups query results according to a specified key value.
into	Provides an identifier that can serve as a reference to the results of a join, group or select clause.
orderby	Sorts query results in ascending or descending order based on the default comparer for the element type.

Clause	Description
join	Joins two data sources based on an equality comparison between two specified matching criteria.
let	Introduces a range variable to store subexpression results in a query expression.
in	Contextual keyword in a join clause.
on	Contextual keyword in a join clause.
equals	Contextual keyword in a join clause.
by	Contextual keyword in a group clause.
ascending	Contextual keyword in an orderby clause.
descending	Contextual keyword in an orderby clause.

Quando vengono eseguite le query

Quando si chiamano operatori LINQ, è sufficiente creare una rappresentazione in memoria della query. La query viene inviata al database solo al momento dell'utilizzo dei risultati.

Le operazioni più comuni che causano l'invio della query al database sono:

- Iterazione dei risultati in un ciclo for
- Uso di un operatore come ToList, ToArray, Single, Count, o gli overload asincroni equivalenti

Operatori di query complessi

LinQ (Language Integrated Query) contiene molti operatori complessi, che combinano più origini dati o eseguono un'elaborazione complessa. Non tutti gli operatori LINQ dispongono di traduzioni appropriate sul lato server.

- Join
- GroupJoin
- SelectMany
- GroupBy
- LeftJoin

Paginazione

Impaginazione offset

```
var nextPage = context.Posts
.OrderBy(b => b.PostId)
.Skip(position)
.Take(10)
.ToList();
```

Impaginazione keyset

```
var nextPage = context.Posts
.OrderBy(b => b.PostId)
.Where(b => b.PostId > lastId)
.Take(10)
.ToList();
```

Salvare i dati

Partendo dai DbSet del modello, è possibile non proiettare (select) un nuovo tipo di dato ma materializzare interamente una Entity.

Una volta materializzata è possibile effettuare una serie di modifiche e renderle persistenti tramite il comando SaveChanges();

```
using (var context = new BloggingContext())
{
   var blog = context.Blogs.Single(b => b.Url == "http://example.com");
   blog.Url = "http://example.com/blog";
   context.SaveChanges();
}
```

Questo è possibile perché EF tiene per noi i Tracking changes delle entity modificate e provvede a generare le istruzioni di modifica necessarie in SQL.

Il Salvataggio tramite SaveChanges() corrisponde ad una transazione del DB.

Domande ?

Riferimenti

- https://learn.microsoft.com/it-it/ef/core/
- https://learn.microsoft.com/it-it/ef/core/providers/?tabs=dotnet-core-cli
- https://learn.microsoft.com/en-gb/dotnet/csharp/language-reference/keywords/query-keywords
- https://learn.microsoft.com/it-it/ef/core/querying/complex-query-operators
- https://learn.microsoft.com/it-it/ef/core/querying/pagination#keyset-pagination