# Esercitazione di Fine Settimana – Week 6

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Larisa Bianca |
|  |  | Cognome | Antonica |
|  |  | Data | 25/06/2021 |

Leggete attentamente ogni domanda e argomentare quanto più possibile fornendo anche degli esempi.

1. Descrivere le modalità di definizione del modello dati in Entity Framework?

Entity Framework è un insieme di tecnologie in ADO.NET che supportano lo sviluppo di applicazioni software orientate ai dati.

Il Entity Framework dà vita ai modelli consentendo di eseguire query su entità e relazioni nel modello di dominio (denominato modello concettuale nell'Entity Framework), basandosi sulla Entity Framework per tradurre tali operazioni in comandi specifici dell'origine dati.

Entity Framework ha diversi approcci:

Database First, quando il modello viene impostato da un Data Base esistente. In questo caso se il DB viene modificato posso quasi sempre aggiornare il modello.

Con Model First, il modello del database viene creato dal designer di Visual Studio e la sua implementazione fisica è basata sul modello generato. Il questo caso il riutilizzo de codice e la separazione tra contesto ed entità non si favorisce. Perché il modello definisce il database, le eventuali modifiche verranno perse.

E in fine con Code First, il modello viene creato dal nostro codice e l’implementazione fisica è basata sul nostro codice.

L’approccio preferito e Code First perché ha il focus sul domain design e perché C# potrebbe risultare più familiare delle query di SQL. Il database può essere messo facilmente sotto source control, evitando il script di SQL e la mola di codice auto generato da EDMX.

1. Scrivere con un esempio pratico come definire una chiave primaria ed una chiave esterna utilizzando le data annotation e fluent api.

Data Annotations:

LE Data annotazioni sono attributi che servono a specificare il comportamento per fare l’override della convenzioni di code-first.

PK:

Override della convenzione sulla PrimaryKey, viene applicato alle proprietà di una classe.

[Key]

[MaxLength(20), MinLength(10)]

public String URL { get; set; }

FK:

Rappresenta l’override della convenzione sulla chiave esterna e viene applicato solo alle proprietà di una classe.

public String CFPerson { get; set; }//FK

[ForeignKey(nameof(CFPerson))]

public Person Person { get; set; }

Fluent Api:

Sono una alternativa complete alla Data Annotations.

Viengono definiti dentro l’override di OnModelCreating.

PK:

modelBuider.Entity<Partecipante>().HasKey(k => k.ID);

FK:

modelBuider.Entity<Gita>().HasOne(i => i.Itinerario)

.WithMany(g => g.Gite)

.HasForeignKey(i => i.ItinerarioID);

1. Descrivere l’utilizzo delle Migration e i vantaggi che ne derivano

Le Migrations servono in caso di modifiche al modello di riflettere sul database.

E’ una nuova tipologia di inizializzazione del database dal EF 4.3.

Utile in caso di database già esistente e non si perdono le eventuali stored procedures, trigger ecc.

Le migrazioni sono di due tipi: automatiche (poco invasive) e manuali o code-based, che richiedono un intervento specifico sul database.

Con le migrazioni automatiche bisogna avviare un comando dalla Package Manager Console per abilitare le migrazioni. In questo caso se il comando ha successo, allora verrà creato il file /Migrations/Cofigurations.cs che rappresenta la nuova strategia di inizializzazione.

Le migrazioni code-based sono utili quando siamo in produzione e lo stato del database è già definito e vogliamo più controllo sulle modifiche automatiche.

Per la migrazione code-based servono due comandi dalla console che sono Add-Migrations con la Migration name che crea una nuova classe con tutte le modifiche rispetto allo stato procedente della database e Update-Database che aggiorna il database con il modello. Esiste anche la possibilità di fare il rolleback di una modifica con Update-Database-TergetMigration:”Migration name”.

Con la code-based viene creato un nuovo file per ogni migrazione.

**Esercitazione Pratica**

Sulla base del modello definito in allegato realizzare i requisiti richiesti.

L'applicazione deve consentire di gestire i clienti dell'assicurazione. In particolare deve essere possibile:

- Inserire nuovi clienti

- Inserire una polizza (rcauto, furto, vita) per un cliente

- Stampare i dati delle polizze presenti a db (comprese le info sul cliente che l'ha stipulata\*).

\*In fase di stampa dei dati del cliente deve essere visualizzabile anche la spesa totale mensile che il cliente sostiene per tutte le sue polizze.

Requisiti tecnici:

Utilizzare le Migration di EF per la modifica del DB

Utilizzare le Fluent-Api per la gestione dei requisiti del database

Realizzare un menù per l’interazione con l’utente