בס"ד

Language-Integrated Query (LINQ)

מה זה שאילתות

אחת המשימות היותר נפוצות שתוכניתנים עוסקים בהם בחיי היום יום הוא אחזור נתונים על פי פרמטרים מסוימים, מתוך סטים של נתונים. נקבל משימות לעבור על סטים של נתונים, לסנן מהם מידע, או לאחזר מידע כזה או אחר שעונה על הגדרות מסוימות, ולאחזר את הנתונים בצורה מאוד מסוימת. לדוגמה, אם יש לנו מאגר נתונים ובו רשימת סטודנטים, עלולה להיות לנו משימה לשלוף ולהציג את השם הפרטי ושם המשפחה, והציון של כל סטודנט בקורס מסויים, ובתנאי שסיים לשלם למכללה את חובותיו.

המעברים האלו על סטים של נתונים נקראים "איטרציות".

הקוד שנכתוב שיבצע איטרציות על המנתונים וישלוף לנו אותם, נקרא שאילתה.

מה זה LINQ ולמה

בצורה בה ביצענו שאילתות בעבר, כתבנו string שמייצג את השאילתה בשפת שאילתות כלשהי, ולא הייתה לנו בדיקה בזמן פיתוח design time או בזמן קומפילציה על שגיאות אפשריות בקוד של השאילתה, היינו מקבלים exceptions בזמן ריצה, מה שכמתכנתים תמיד נרצה למנוע.

בנוסף, בעבר היינו צריכים להתמחות במספר שפות של שאילתות (כגון: SQL, XML documents ושירותים מסוגים שונים), וללמוד איזה שפה שעובדת מול אותו סוג מאגר נתונים..

LINQ זה שם של מספר טכנולוגיות שמיועדות לתת אפשרות למתכנתי #C לכתוב שאילתות בשפת #C עצמה. אם בעבר היינו צריכים להשתמש בשאילתות בשפות אחרות ולהפעילם מתוך #C, או לבצע קוד מורכב שיבצע איטרציות על סטים של נתונים, לולאות כאלו ואחרות, כעת ניתן להשתמש ב #C בתחביר שחוזר על עצמו מול כל מיני סוגי מאגרי נתונים, ולתשאל סטים של נתונים ולאחזר נתונים מהם (כלומר, לבצע שאילתות).

הכנסת שפת שאילתות לתוך #C מאפשרת במקרים רבים לבדוק תקינות של שאילתה כבר בזמן פיתוח. וכיום עקרונית נוכל להתמקד ולשכלל את הידע שלנו בשפה אחת שעובדת מול כל מיני סוגי סטים ומאגרים של מידע, ולא צריך להתמחות בשפות שאילתות שונות.

העיקרון ב LINQ הוא שאנחנו מבצעים את השאילתות שלנו מול מחלקות ושדות שאנחנו מכירים את ה type שלהם, וכך יש לנו type safety עם עזרה של הקומפיילר, ונשתמש באחת הטכנולוגיות ממשפחת הטכנולוגיות של LINQ כדי לתרגם את שאילתת ה LINQ שלנו ולהתאימה לעבודה מול מאגר הנתונים מולו רוצים לעבוד.

לדוגמה: יש לנו LINQ to SQL שיודע לתרגם את שאילתות ה LINQ שלנו לשפת SQL ולשלוף נתונים על פי השאילתה שלנו מתוך מסד נתונים SQL-י (קראו: אס קיו אלי). ויש לנו את LINQ to Objects ו LINQ to XML שמבצעים כשמם אחזור נתונים מרשימת אובייקטים או מתוך קובץ XML בהתאמה, ויש עוד תרגומים לכל מיני סוגי מאגרים (של חברות צד שלישי), מה שמאפשר לך לעבוד ב #C מול כל מיני שירותים בלי ללמוד את השפה הספציפית שלהם.

כבר למדנו על ה Interface שנקרא IEnumerable או בגרסתו הגנרית IEnumerable<T> שמייצג סטים של נתונים שאפשר לעשות עליהם איטרציות. כלומר, שאפשר לעבור עליהם אחד אחרי השני או בבסדר אחר ולאחזר מהם נתונים.

בס"ד

חשוב להדגיש ומעניין מאוד לדעת הוא שכאשר מממשים את הממשק IEnumerable מקבלים סט של מתודות LINQ שמאפשרות לתשאל את המאגר שמייצג האובייקט שמימש את ה IEnumerable.

כנ"ל כשמממשים את IQueryable<T> and IQueryable רק שזה מתקמפל בצורה אחרת למשהו שנקרא EpressionTrees.

כשאנחנו משתמשים ב LINQ לביצוע שאילתות על סט נתונים שמממש IEumerable אנחנו בעצם עושים שימוש ב LINQ to Objects.

אחרי שהתאמנת והתרגלת לכתוב שאילתות LINQ זה די שקוף מבחינתך, מול איזה סוג מאגר נתונים אתה עובד, מכיוון שהתחביר יהיה זהה בין אם אתה עושה איטרציות מול מסד נתונים או מול List של אובייקטים.

עם זאת, חשוב לדעת שכדאי מאוד להבין את צורת ההמרה של שאילתת LINQ לטובת המאגר מולו אתם עובדים, כי לפעמים זה כן ישפיע על הצורה בה נכתוב את השאילתה.

לדוגמה, כשעובדים מול מסד נתונים SQL אם נכתוב קוד שלא מתורגם לשאילתת SQL, המערכת עלולה לשלוף טבלאות שלימות לזיכרון ולעבוד מול הזיכרון, מה שיהיה מאוד לא יעיל, ובמקרים רבות יפיל לכם את האפליקציה. כשנכתוב על LINQ to SQL ניתן דוגמאות למקרים כאלו.

כשכותבים שאילתות LINQ יש לנו שני סוגי תחבירים לעבוד איתם. שניהם מתקמפלים מאחורי הקלעים לאותו קוד, ולכן אין עדיפות מבחינת יעילות לאחד מהם. תכניתן יבחר תחביר שנוח לו ו/או מקובל בארגון בו הוא עובד (תחביר שאילתות נחשב לקריא יותר).

תחביר ראשון, ישתמש בביטויי למבדה בכדי לאחזר נתונים.
התחביר השני ישתמש בביטויי שאילתות.

לפעמים תוכניתנים נוטים לבלבל בין הגדרת השאילתה להרצה שלה., לכן, רגע לפני שמתחילים עם דוגמה, יעזור לנו אם נשנן לעצמנו 3 נושאים/שלבים נשים לב אליהם ונחדד אותם:

1. יש את מאגר הנתונים שעליו אנחנו עובדים. הגדרת ה data source, מאגר הנתונים.
2. הגדרת השאילתה.
3. הרצת השאילתה.

ונתחיל עם דוגמאות, ונתייחס כרגע לעבודה עם LIST שכידוע מממש את IEnumerable, ולכן, ניתן לאחזר ממנו נתונים באמצעות LINQ to Objects.

נראה דוגמה לשימוש בביטוי שאילתות, ונזכור גם, כי יש לו סט של Extension Methhods שמייצגים שאילתות LINQ להם מעבירים ביטויי למבדה.

נראה את הקוד הבא:

```
// Specify the data source.
int[] scores = new int[] { 97, 92, 81, 60 };

// Define the query expression.
IEnumerable<int> scoreQuery =
                from score in scores
                where score > 80
```

```
                select score;

// Execute the query.
foreach (int i in scoreQuery)
{
      Console.Write(i + " ");
}
```

כמו שאנחנו רואים, בתחביר הזה שאילת הלינק מחולקת לשלושה שלבים:
1. מהו המאגר מולו עובדים
2. סינון (כלומר, אלו רשומות לאחזר)
3. מה לאחזר מתוך כל רשומה שעונה על הקריטריון (שהוגדר בסעיף הקודם).

בדוגמה הקודמת, scoreQuery הוא שם המשתנה שמצביע על השאילתה שלנו. רק באיטרציה על השאילתה, בשורת ה Foreach אנחנו מפעילים את השאילתה, ומדפיסים את התוצאה אחת אחרי השניה בלולאה.

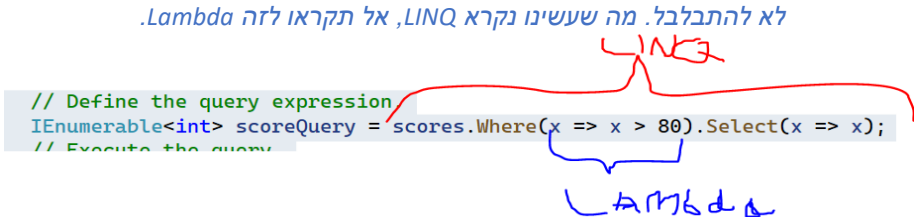
כעת נבצע את אותה שאילתה תוך שימוש ב LINQ בתחביר אחר, דרך פונקציה שחושפת delegate מתאים, ומאפשרת להעביר לה ביטוי למבדה שמייצג כזכור פונקציה לפילטור.

```
// Specify the data source.
int[] scores = new int[] { 97, 92, 81, 60 };

// Define the query expression.
IEnumerable<int> scoreQuery = scores.Where(x => x > 80).Select(x => x);
// Execute the query.
foreach (int i in scoreQuery)
{
    Console.Write(i + " ");
}
```

בדיוק כמו בדוגמה הקודמת. יישמנו לינק בדרך אחרת.
בדוגמה האחרונה יכולנו לוותר על ה Select. לא יכולנו לוותר על ה select בדוגמה עם תחביר שאילתה.

לא להתבלבל. מה שעשינו נקרא LINQ, אל תקראו לזה Lambda.



עשינו שימוש ב LINQ תוך העברת Lambda Expression.

אז איך בונים שאילתה בתחביר שאילתה?

לשאילתות יש מקטעים שונים, הם נקראים clauses. נציין את המרכזיים שבהם. אין חובה להשתמש בכל המקטעים בכל שאילתה, ואפשר להשמיט חלק מהם, לחלק מהמקטעים אפשר להחליף את הסדר ביניהם וחלקם אי אפשר. ה from יבוא ראשון, ה select יבוא אחרון.

1	הגדרת מאגר הנתונים	נגדיר את מאגר הנתונים מולו עובדים (ושם משתנה שישמש כפרמטר, הוא מייצג את השם של כל איבר במאגר במהלך האיטרציה)	`from score in scores`
2	Filtering - סינון	ביטוי ב #C שמייצג את הלוגיקה לסינון	`where score > 80`
3	Ordering - מיון	אין למיין את הנתונים, על פי אלו שדות והאם בסדר עולה או יורד	`orderby cust.Name ascending`
4	Grouping - קיבוץ	ניתן לקבץ את הנתונים לקבוצות. למשל: נרצה לקבץ את הסטודנטים שעונים על השאילתה שלנו, לפי עיר מגוריהם. כלומר, להציג יחד את כל הסטודנטים הירושלמיים, ואחריהם את התל אביביים	`group cust by cust.City`
5	Joining (חיבור ואיחוד)	יפורט בגרסה הבאה של מסמך זה (אולי כשנלמד מסדי נתונים)	
6	Selecting - בחירה של מה לאחזר מכל רשומה		

Execution of a LINQ Query

נזכור שיש שלב של הגדרת השאילתה, ואז ורק אז עושים בה שימוש. זה אמנם יכול להיכתב בשורה אחת, אך נזכור שאלו הם השלבים.

ביצוע השאילתה מתרחשת מול מאגר הנתונים, אחרי שהגדרנו שאילתה ועשינו על השאילתה איטרציה באמצעות foreach למשל, או לחילופין, החזרנו את התוצאה לתוך מערך, רשימה, מילון () וכדומה באמצעות פקודות מובנות, למשל: ()ToList וכמו בדוגמה הבאה.

```
// Define the query expression.
IEnumerable<int> scoreQuery1 = scores.Where(x => x > 80).Select(x => x);
// Execute the query.
foreach (int i in scoreQuery1)
{
    Console.Write(i + " ");
}

// Executing the query again.
var list = scoreQuery1.ToList();
```

בס"ד

Selecting

אנחנו יכולים לבחור מה השאילתה מחזירה עבור כל איבר שחוזר מהאיטרציה. אפשר לאחזר את האובייקט שעליו עשינו מעבר, אפשר להחזיר ערך שמורכב מחישוב שנעשה על האובייקט, אפשר להחזיר אובייקט שנאתחל ונשים לו ערכים מתוך האיבר עליו עוברים. אפשר להחזיר KeyValuePair עבור קיבוצים (אפשר להמיר את השאילת שמקבצת ל dictionary), ואפשר גם להחזיר מחלקה אנונימית. דיברנו בשיעור על דוגמאות שונות.

Anonymous Class

הנושא של מחלקה אנונימית לא קשור ל LINQ אבל פעמים רבות כשעובדים עם LINQ מחזירים מופעים של מחלקה אנונימית, אז נסביר במספר שורות במה מדובר.

למחלקה אנונימית אין שם, יש לה רק מאפיינים שהם read only ולא ניתן להוסיף לה שדות, מתודות ואירועים או כל דבר אחר. מגדירים את זה על ידי שימוש במילה new והשמת ערכים בתחביר שמוכר לנו כ Object Initializer.

לדוגמה

```
var myInstance = new { FirstName = "Moshe", LastName = "Nissan" };
Console.WriteLine(myInstance.FirstName);
```

הגדרנו מחלקה אנונימית, וכרגע יש לנו מחלקה עם המאפיינים שם פרטי ומשפחה. ניתן להשתמש במופע ולהדפיס או לעשות מה שרוצים עם הערכים שלו.

הרבה פעמים נמצא את עצמנו מחזירים משאילתת לינק, מופעים של מחלקה אנונימית.

```
IEnumerable<int> listOfNames =
                from student in students
                where student.Grade > 80
                select new { Name = student.FirstName + " " + student.LastName,
Grade = student.Grade };
```

בדוגמה זו אינני רוצה להחזיר את כל המחלקה סטודנט עבור כל סטודנט מצטיין, ומספיק לי להחזיר את שמו וציונו, והשתמשתי לצורך כך במחלקה אנונימית.

כמו בכל נושא בתכנות, יש עוד כל כך הרבה מה ללמוד על LINQ (ולא שייך במסגרת קורס ללמוד הכל, וגם תוכניתנים מנוסים לא בהכרח יודעים או זוכרים את הכל), ומי שנשאר לו זמן ירחיב את ידיעותיו עם לימוד עצמי, כרגע המשימה הכי חשובה שלכם, היא להמשיך ולהתאמן על הידע שכן צברתם בינתיים.

אצרף כמה לינקים, לתכנים בנושא LINQ:

https://www.w3resource.com/csharp-exercises/linq/index.php

/https://linqsamples.com

בס"ד

תרגילי כיתה ובית

נבצע את התרגילים גם בתחביר שאילתה וגם בתחביר של delegates והעברת פונקציה (להשתדל להעביר פונקציות למבדה).

תרגילים על מאגר של מספרים שלמים:

1. שלוף ממערך או רשימת מספרים שלמים את כל המספרים השליליים.
2. שלוף את כל המספרים האי זוגיים, מיין אותם בסדר יורד
3. עבור כל מספר שגדול מ 5 ולא מתחלק ב 3 בלי שארית, החזר את כפולת המספר ב 3.
   למשל: למספר 4 או 5 לא נתייחס כי הם אינם גדולים מ 5. ל 6 לא נתייחס כי הוא מתחלק ב 3 בלי שארית, ועבור המספר 7 נחזיר 21.

נבנה רשימת ערים string.

4. נכתוב פונקציה שמקבלת string, ומחזירה באמצעות שאילתת לינק את כל הערים ברשימה בהם מופיע המחרוזת שקיבלנו.
5. נכתוב פונקציה שמקבלת string ומחזירה את כל הערים שאינם מתחילות במחרוזת שקיבלנו.
6. נכתוב שאילתה המחזירה את העיר הראשונה שיש בה x בשם שלה.
7. נכתוב שאילתה שתמיין את רשימת הערים לפי שם ותחזיר את שלושת הראשונים.

נבנה רשימת ערים, עיר תהיה בתרגילים הבאים מופע של מחלקת City לה יש Id, Name, NumberOfPopulation

8. נחזיר את הערים להם יותר מ 25,000 תושבים.
9. נחזיר את שמות הערים לערים להם מעל 25,000 תושבים.
10. נבנה שאילתה שמחזירה מחלקה אנונימית עם שם העיר והאם זו עיר או יישוב (עיר ייחשב אם יש מעל 25000 תושבים).