שלב 5 – מימוש שכבת DAL ע"י קבצי XML

הקדמה

עד עכשיו השתמשנו בפרויקט **DalList** כדי להוסיף/למחוק/לעדכן/לקבל ישויות מבסיס הנתונים. כאשר בסיס הנתונים היה רשימות מסוג List<T> שמאותחלות מחדש בכל ריצה. מימשנו את הממשק **ICrud** עבור כל ישות בנפרד בעזרת גישה לרשימות רגילות List<T>. את הרשימות הראשוניות הגדרנו במחלקה **DataSource** ואתחלנו אותם בתוך הפונקציה **Initialize**.

כעת, אנחנו הולכים ליצור פרויקט חדש בשם **DalXml** שיממש גם הוא את **ICrud** אבל בעזרת כתיבה וקריאה מקבצי **XML**, כך שהנתונים המעודכנים ישמרו לנו מריצה לריצה.

חזית הנתונים (פרויקט **DalFacade**) לא תשתנה אלא רק יתווסף פרויקט חדש בשם DalXml שיהיה במבנה זהה לפרויקט DalList. הפרויקט DalXml יגדיר עבור כל ישות, מחלקה שתממש את הממשק המתאים לישות, באופן שהנתונים נשמרים ונשלפים ב/מקבצי XML. לכל ישות יהיה קובץ XML נפרד שבו ישמר אוסף נתונים מאותה ישות.

הפרויקט DalList והפרויקט DalXml לא ירוצו במקביל אלא כל פעם ירוץ רק אחד מהם.

1. יצירת פרויקט חדש בשם DalXml מסוג Class Library

הוסיפו ל **Solution** פרויקט חדש בשם **DalXml** מסוג **Class Library**.

לאחר שייפתח פרויקט חדש תחת ה solution, לחצו לחיצה כפולה על שם הפרויקט ב-Solution Explorer.

ייפתח לעריכה קובץ **לניהול המאפיינים של הפרויקט,** הוסיפו לקובץ את השורות הבאות (המודגשות בצהוב) בסוף האלמנט **PropertyGroup**:

*<Project Sdk="Microsoft.NET.Sdk">*

*<PropertyGroup>*

*<TargetFramework>net7.0</TargetFramework>*

*<ImplicitUsings>enable</ImplicitUsings>*

*<Nullable>enable</Nullable>*

**<BaseOutputPath>$(SolutionDir)\bin\</BaseOutputPath>**

**<OutputPath>$(SolutionDir)\bin\</OutputPath>**

**<AppendTargetFrameworkToOutputPath>false</AppendTargetFrameworkToOutputPath>**

**<AppendRuntimeIdentifierToOutputPath>false</AppendRuntimeIdentifierToOutputPath>**

**<UseCommonOutputDirectory>true</UseCommonOutputDirectory>**

**<RootNamespace>Dal</RootNamespace>**

*</PropertyGroup>*

*</Project>*

**נרצה שפרויקט DalXml יוכל לגשת בקוד לישויות ולמחלקות הציבוריות שיוגדרו בפרויקט DalFacade ולכן ניצור הפניה (reference) ממנו אליו.**

הכנסו **לקובץ המאפיינים של הפרויקט**, ובדקו שלאחר שהוספתם את הרפרנס בין פרויקט DalXml לבין פרויקט DalFacade, אז נוסף, הטקסט הבא:

*<ItemGroup>*

*<ProjectReference Include="..\DalFacade\DalFacade.csproj"/>*

*</ItemGroup>*

1. יצירת תיקיה בשם xml עבור קבצי ה xml של התוכנית.

הערה: יתכן שכבר הוספתם תיקיה זו בשלבים קודמים, אם כן- ניתן לדלג לסעיף הבא.

1. **פתחו את תיקיית ה solution (מקש ימני -> Open Folder in File Explorer)**
2. **צרו תיקיה חדשה בשם xml**
3. תחת תיקיה זו, צרו קובץ טקסט חדש בשם **dal-config.xml**. (שימו לב למחוק מסיומת שם הקובץ את ".txt")
4. לאחר מכן, בויז'ואל סטודיו, עמדו עם עכבר ימני על ה **solution** ואז: **Add → New Solution Folder**. קראו לספריה בשם **xml** (באותיות קטנות)
5. יצירת קובץ קונפיגורציה של נתונים כלליים בשם data-config.xml

**המטרה**: הוספה לספריה **xml**, קובץ קונפיגורציה שנוכל לשמור בו נתונים נוספים כללים, שאינם חלק מבסיס הנתונים עצמו. כלומר, נתונים כלליים שאינם חלק מרשימות הנתונים עצמן. למשל: נתון ששומר את המספר הרץ האחרון עבור ישות מסוימת. שישמר מריצה לריצה, ובריצה הבאה נוכל להתחיל מהמספר הרץ האחרון שניתן.

הנתונים שנשמור בקובץ זה, מתאימים באופן כללי לשדות שהגדרנו במחלקה Config הנמצאת בפרויקט DalList. נרחיב בהמשך.

1. **בסייר הקבצים (File Explorer),** גשו לספריה xml
2. תחת ספריה זו, צרו קובץ חדש בשם **data-config.xml**. (ע"י יצירת קובץ טקסט חדש ושינוי השם. שימו לב למחוק את הסיומת האוטומטית ".txt")
3. **בויז'ואל סטודיו,** עמדו עם עכבר ימני על הספריה **xml** ובחרו **Add → Existing Item**
4. בחרו להוסיף את הקובץ הקיים שזה עתה יצרתם: xml/**data-config.xml**
5. **פתחו את הקובץ דרך הויז'ואל סטודיו** והעתיקו את הטקסט הבא להיות תוכן הקובץ עצמו: **כרגע הקובץ ריק מנתונים**, **ומופיע בו רק אלמנט השורש**, <config>, בהמשך תוסיפו לו נתוני קונפיגורציה על פי הצורך):

<?xml version="1.0" encoding="utf-8"?>

<config>

</config>

1. בצעו שמירה (Save) של הקובץ
2. יצירת קבצי xml שיהוו את בסיס הנתונים

**המטרה**: כאמור, נרצה לשמור את בסיס נתונים שלנו בקבצים חיצוניים מסוג XML, על מנת שהוא ישמר מריצה לריצה.

כל רשימה בבסיס הנתונים מכילה את כל הישיויות מאותו סוג, וכל רשימה תשמר בקובץ XML נפרד.

בתחילה נייצר קבצים ריקים לגמרי.

**עבור כל אחת מישויות DO הוסיפו קובץ XML באופן הבא:**

1. כפי שהוסבר לעיל, הוסיפו לספריה xml **בסייר הקבצים** קובץ XML חדש. **שם הקובץ יהיה בפורמט הבא:** אותיות קטנות, שם הישות עם תוספת של s, וסיומת xml.  
   למשל: עבור ישות Student נייצר קובץ בשם students.xml
2. לאחר מכן, כפי שהוסבר לעיל, הוסיפו את הקובץ הזה לספריה xml גם בויז'ואל סטודיו
3. הקובץ הריק שנוצר יכיל בהמשך את רשימת כל האובייקטים מאותה ישות בפורמט XML - בשלב זה הרשימה ריקה ולכן נוסיף לשורת הכותרת בקובץ אלמנט ריק ששמו **המדויק** הוא **שם הישות** שמתחיל באות גדולה, עם תוספת משמאל של **ArrayOf**.  
   למשל עבור ישות **Student** הקובץ יראה בהתחלה כך:

<?xml version="1.0" encoding="utf-8"?>

<ArrayOfStudent>

</ArrayOfStudent>

1. יצירת מחלקת config עבור נתוני קונפיגורציה כלליים

**כזכור,** **בשלב 1, בפרויקט DalList** המממש את בסיס הנתונים עם רשימות List, הגדרנו מחלקה בשם **Config (מחלקה פנימית בתוך מחלקת DataSource),** עבור נתוני קונפיגורציה כלליים. **למשל**: תכונות שמיוחסות למספר רץ, שמחזירות את המספר הרץ הבא ומקדמות אותו באחד בכל קריאה. תכונות אלו אותחלו מחדש באותו ערך בין ריצה לריצה.

**כעת, בשלב 3,** **בפרויקט DalXml נגדיר גם כן מחלקה בשם Config, שתכיל תכונות שמקבילות לנתוני הקונפיגורציה הכלליים שנדרשו בשלב 1.**

מכיוון שהנתונים הללו נשמרים בקובץ ומתעדכנים בקובץ, אזי היתרון הוא שהם ישמרו מריצה לריצה. למשל: המספר הרץ האחרון מהריצה הקודמת ישאר ובריצה הבאה נמשיך להתקדם ממנו.

1. בפרויקט DalXml, הוסיפו מחלקה חדשה, סטטית בהרשאת internal, בשם Config.
2. הוסיפו למחלקה **שדה פרטי שישמור את שם קובץ** הקונפיגורציה של הנתונים הכללים. שם הקובץ (data-config) ללא סיומת xml.
3. בספריה xml, הוסיפו בתוך הקובץ data-config.xml אלמנט חדש שמקביל לכל תכונת קונפיגורציה שהוגדרה במחלקת Config, ומלאו אותם בערכים התחלתיים:

<?xml version="1.0" encoding="utf-8"?>

<config>

<NextCourseId>1008</NextCourseId>

<NextLinkId>41</NextLinkId>

</config>

1. במחלקת Config הוסיפו תכונה מקבילה לכל אלמנט בקובץ.  
   בפונקציה get של כל תכונה, החזירו את הערך המתאים מתוך הקובץ וקדמו אותו למספר הרץ הבא.
2. בהמשך נשתמש בתכונות הללו תוך כדי הקוד שמממש את מתודות ה CRUD בעזרת פניה לקבצי XML. למשל כך: int nextId = Config.NextCourseId;
3. מימוש מחלקות הגישה לנתונים בפרויקט DalXml
4. הגדרת מחלקה מממשת עבור כל ישות: עבור כל ישות DO, הוסיפו לפרויקט DalXml מחלקה חדשה בהרשאת internal, ששמה כשם הישות בתוספת Implementation.  
   שימו לב שהמחלקה נוצרה תחת namespace Dal;  
   הוסיפו הגדרת ירושה מהממשק המתאים לישות זו.
5. מימוש המתודות עבור כל ישות: כעת יש לממש את פעולות CRUD בכל אחת מהמחלקות ע"י גישה לקבצי xml שהגדרנו.

**להלן כללים למימוש תקין של מתודות המימוש בעזרת קבצי XML:**

1. **קבצי ה XML המהווים את בסיס הנתונים ימצאו כאמור תחת הספריה בשם xml ויהיו במבנה הבא:**

כל מחלקה (ישות DO) תהיה שמורה בקובץ XML נפרד כאשר

* + כל קובץ יכיל רשימה של אלמנטים בשם הישות, תחת אלמנט שורש אחד.
  + כל אלמנט בקובץ XML יקביל למופע של ישות ויכיל בתוכו אלמנט לכל מאפיין של הישות
  + מאפיינים ללא ערך (שערכם באותו זמן הוא null) - האלמנט שלהם לא יופיע באותה ישות

1. **תוכן קבצי ה XML יתמלא ויתעדכן דרך הקוד, ע"י מימוש מתודות ה CRUD. במימוש המתודות הללו תעשו שימוש במחלקה XElement ובמחלקה XmlSerializer כאשר:**
   * עבור כל הישויות תוכלו לממש בעזרת המחלקה **XmlSerializer**  (צורה פשוטה יותר למימוש)
   * חוץ מישות אחת (לפחות), שעליכם לממש באמצעות  **Linq to xml** (המחלקה **XElement**)
2. **מהלך עבודה תקין בזמן פעולה (עדכון/מחיקה/שליפה/הוספה לרשימה) - ע"פ שיטה 1 - המחלקה XmlSerializer:**

* **בתחילת הפעולה**, בעזרת המחלקה XmlSerializer טוענים מקובץ XML את רשימת האובייקטים לרשימה לוגית מסוג List.   
  פעולה זו נקראת **Deserialize**
* **במהלך הפעולה**, מנהלים את הרשימה דרך פעולות על אותה הרשימה הלוגית מסוג List: עדכון/מחיקה/שליפה/הוספה לרשימה
* **בסוף הפעולה**, שומרים את ה List בחזרה לקובץ XML דרך ה XmlSerializer.  
  פעולה זו נקראת **Serialize**
* חשוב שההרשאה של המחלקות/רשומות שמסוריאלות תהיה public (כבר דאגנו לכך בשכבת ה Dal)

1. **מהלך עבודה תקין בזמן פעולה (עדכון/מחיקה/שליפה/הוספה לרשימה) - ע"פ שיטה 2 - המחלקה XElement:**

* **בתחילת הפעולה**, טוענים מקובץ XML את רשימת האובייקטים לתוך אובייקט מסוג המחלקה XElement.
* **במהלך הפעולה**, מנהלים את הרשימה דרך פעולות על אותו אובייקט מסוג המחלקה XElement: עדכון/מחיקה/שליפה/הוספה לרשימה
* **בסוף הפעולה**, שומרים את האוסף מסוג XElement בחזרה לקובץ XML.

1. **ודאו שלכל ישות** בתוך ה {} של ה-record מוגדר **גם בנאי ללא פרמטרים** (בנוסף לבנאי עם הפרמטרים). מכיוון שזוהי דרישה של מחלקת **XmlSerializer.**
2. יצירת מחלקת DalXml שתממש את Idal

**המטרה**: הוספה לפרויקט **DalXml** מחלקה בשם **DalXml** שתירש ותממש את הממשק **IDal** ע"י אתחול תתי הממשקים במחלקות הגישה שמימשנו זה עתה בעבודה עם XML.

1. בפרויקט DalXML הוסיפו מחלקה עם הרשאה **public**  בשם **DalXml** שתממש את הממשק **IDal**.
2. **עבור כל אחד מתתי הממשקים** ממשו בתוך המחלקה **DalXml** את התכונה (property) המתאימה שהוגדרה עבורו בממשק **IDal**, כך שיחזירו אובייקטים מהטיפוס שמממש את הממשק המתאים לכל ישות נתונים (יש לאתחל את התכונה בהגדרתה).

**שימו לב שזה ממש דומה לקוד של המחלקה DalList, מכיוון ששתיהן ממשות את אותו ממשק. אחת בעזרת List ואחת בעזרת קבצי XML)**

1. **בדיקת שכבת הנתונים החדשה ע"י פרויקט DalTest**
2. **יש להוסיף reference (מאולץ) מפרויקט DalTest לפרויקט DalXml.**
3. **בקובץ dal-config.xml, יש לשנות את הערך באלמנט <dal> להיות xml.**
4. **בקוד של התוכנית הראשית, יש להפוך את אתחול הנתונים (initialization) לאופציונאלי (ע"י בחירה של המשתמש) כדי שבהרצות הבאות לא יתווספו שוב הנתונים השמורים בקבצי ה xml.**