ADO .NET

אביעד דאדון

ActiveX Data Object (.Net)

ADO.Net היא טכנולוגיה אשר באמצעותה אנו מתחברים למסדי נתונים. הטכנולוגיה עצמה היתה קיימת עוד לפני ה.NET

ADO.NET

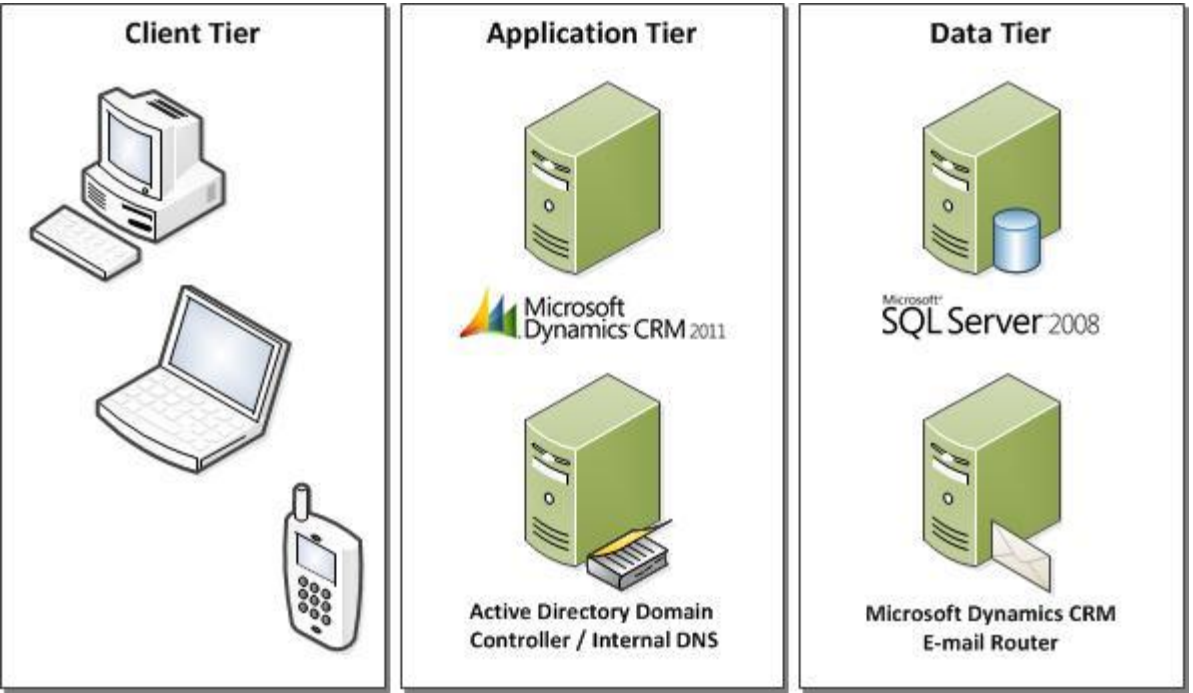
מודל 3 שכבות 3-Tier

PL – Presentation Layer	GUI – Winform, WPF, Browsers, …
BLL – Business Logic Layer	השכבה המחברת בין הPL ל DAL. שכבה זו אחראית על קבלת המידע, ביצוע חישובים ומניפולציות והעברת המידע לPL. (קוד בC#)
DAL – Data Access Layer	שכבת המידע – בסיס הנתונים

3-Tier

הרעיון הוא לחלק את הפיתוח ל3 שכבות מופרדות.

באפליקציה "לוקאלית" החלוקה עצמה תראה קצת מוזרה אך במציאות ההפרדה קיימת בצורה המומחשת מצויין בעולם האינטרנט:



3-Tier

התקשורת בין השכבות חייבת להיות בסדר הבא:

PL -> BLL -> DAL

PL <- BLL <- DAL

3-Tier

אז למה לפתח בשכבות?

1. אבטחת מידע.
2. סדר לוגי לתחזוקה – יותר קל לזהות היכן התקלות.

3-Tier הוא הבסיס אשר ממנו פיתחו את מודל n-Tier אשר יכולים להיות יותר מ3 שכבות אך רעיון 3 השכבות הוא הבסיס.

ADO .NET

בפיתוח .NET קיימות 3 שיטות להתחברות לבסיסי נתונים:

1. Connected Layer - קלאסי
2. Disconnected Layer – מ2005
3. Entity Framework – מ2010 (יצאה לפני גרסא ראשונה ב2008 שמפתחים לא רצו להשתמש בה מחוסר ביצועים ותקלות).

Connected Layer

שיטה זו מבוססת על יצירת קובץ DLL אשר כולל בתוכו פונקציות כלליות לביצוע פעולות בבסיס הנתונים.

השיטה נקראת Connected מאחר ואנחנו כל הזמן מחוברים לבסיס הנתונים.

יתרונות:
שימושי לבסיסי נתונים עם הרבה "בקשות"
ידוע כהתחברות יעילה מבחינת ביצועים
קל לזהות מהיכן מגיעות שגיאות

חסרונות:
עבודת קוד רבה.
תחזוקה.
צריך לבצע שינויים בקובץ הDLL כל פעם.
חובה להכיר מבנה DLL

Disconnected Layer

בשיטה זו ניתן ליצור התחברות לבסיס נתונים ברמת הקוד או בצורה ויזואלית באמצעות אובייקטים כאשר המידע נלקח ממסד הנתונים, נשמר בתוך אובייקט והעבודה היא מול האובייקט ולא המסד נתונים עצמו.

- שיטה זו מאפשרת לנו לייצר אובייקט DataSet.
- אובייקט DataSet מכיל מספר אובייקטים מטיפוס DataTable אשר מכילות את הנתונים של הטבלאות.
- ההתחברות לשאילתות בסיסי נתונים מתבצעת בעזרת אובייקט SqlDataAdapter.

Disconnected Layer

יתרונות:

פיתוח מהיר מבוסס על ויזואליזציה

ממוקד צד לקוח ולא צד שרת

שימושי באפ' WEB

חסרונות:

שיטה פרימיטיבית של שימוש באובייקטים של בסיסי נתונים

שימושי לצד לקוח (לא רצוי לפתח צד שרת)

Entity Framework

שיטה זו ידוע כ Pure ORM

Object-Relational-Mapping המאפשרת לנו להמיר באופן אוטומטי טבלאות מבסיסי נתונים לאובייקטים (Entity) ב#C

Entity Framework

יתרונות:

פיתוח מבוסס על אובייקטים #C

עדיף מאשר שימוש ב DataSet

מקל על כתיבת הלוגיקה. במקום להתעסק בלוגיקת ההתחברות.

חסרונות:

עובד על Lazy loading

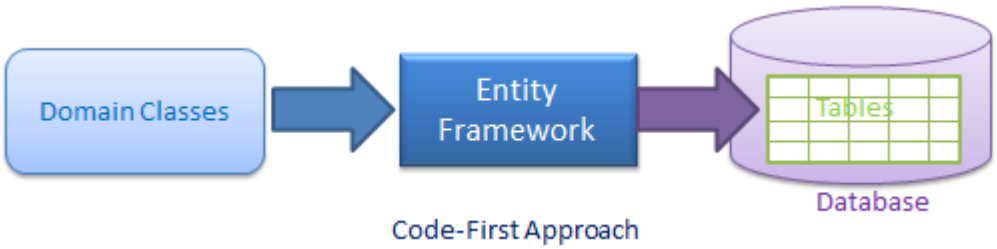
שימושי ונפוץ לאפליקציות צד לקוח בעיקר אפליקציות WEB – ז"א לא יחזיק טוב הרבה בקשות.

שיטות התממשקות ל-DB

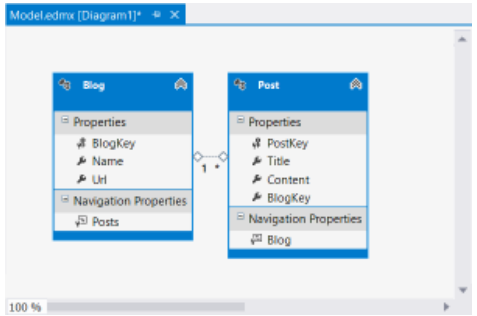
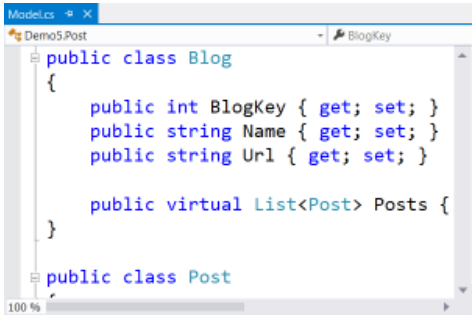
Database First - ראשית יש בסיס נתונים ואז כותבים קוד.

Model First – ניתן למדל את הEntities ואז ליצור את בסיס הנתונים.

Code First – כתיבת קוד בC# ומהקוד ניצור בסיס נתונים. POCO – Plain Old CLR Objects



שיטות התממשקות ל-DB

	Designer Centric	Code Centric
		
New Database	**Model First** Create model in EF Designer Generate database from model Classes auto-generated from model	**Code First** Define classes and mapping in code Database created from code Migrations apply model changes to database
Existing Database	**Database First** Reverse engineer model in EF Designer Classes auto-generated from model	**Code First** Define classes and mapping in code EF Power Tools provide reverse engineer