ESZTERHÁZY KÁROLY KATOLIKUS EGYETEM

Matematikai és Informatikai Intézet

Árgép robot fejlesztése

Készítette
Kis Sándor
Programtervező Informatikus BSc

Témavezető
Nagy Péter
Külső konzulens

Eger, 2024

Tartalomjegyzék

Bevezetés

A mindennapi munkám során számos ismétlődő feladatokat kell elvégeznem a számítógépen. Ezek a tevékenységek általában adatbevitellel és különböző, egy sablonra épülő riportok készítésével kapcsolatosak. Az ilyen típusú feladatok igen időigényesek, Én naponta akár több munkaórát is ezekre fordítok. Az így eltöltött időt sokkal értékesebben is fel tudnám használni, ezáltal sokkal hatékonyabban tudnám elvégezni a munkámat.

Gondolom, nagyon sokan vannak hasonló helyzetben, akik egyhangú, monoton, ismétlődő munkát végeznek számítógépen. Ezeknek az ismétlődő munkafolyamatoknak az automatizálásával rendkívüli hatékonyságot lehet elérni úgy, hogy rengeteg munkaórát lehet megtakarítani, és még a hibalehetőségeket is a minimálisra csökkentjük. Az automatizálásra számos megoldás létezik, az egyik olyan technológia, amely kiemelkedik ezen a területen, az UiPath, amely a robotfolyamat-automatizáció (RPA) terén kiváló megoldásokat kínál.

A UIPath technológia kulcsfontosságú szerepet játszik a munkafolyamatok automatizálásában. Azokon a területeken, ahol az ismétlődő feladatok és a rutinszerű folyamatok dominálnak, az RPA segítségével rendkívül hatékonyan lehet robotokat bevezetni. Például az adatbevitel, az adatellenőrzés és az adminisztratív folyamatok automatizálásával az UiPath robot gyors és precíz műveleteket végez szinte hibatlanul.

Egy másik fontos aspektus az RPA alkalmazásában az, hogy a UiPath rendszer lehetővé teszi az integrációt más vállalati rendszerekkel. Ez azt jelenti, hogy a meglévő informatikai infrastruktúrát könnyen kombinálhatjuk a robotfolyamat-automatizációval, anélkül, hogy átfogó átalakításokra lenne szükség. Ez a kompatibilitás lehetővé teszi a vállalatok számára, hogy fokozatosan vezessék be az automatizációt, kezdve a legkritikusabb területekkel, és később kiterjeszthetik azt az egész vállalati környezetre.

Az automatizáció tehát nemcsak meggyorsítja a munkafolyamatokat, de elősegíti a vállalati hatékonyság növelését, a humán erőforrások felszabadítását és a vállalatok versenyképességének javítását. Azok a vállalatok, amelyek rugalmasan alkalmazzák az automatizációt, könnyebben alkalmazkodhatnak az üzleti környezet változásaihoz, és előnyt szerezhetnek a versenytársaikkal szemben a dinamikus piaci feltételek között.

Célkitűzés

A szakdolgozatom elkészítése során a UiPath technológiát felhasználva létrehoztam egy Árgép robotot, ami segít összeállítani egy asztali PC-t a lehető legolcsóbb áron. Az Árgép robot egy adatbázisból dolgozik, amelyben tárolva vannak az alkatrészek és a webshopok információi.

A felhasználók egyszerűen felsorolják a kívánt alkatrészeket egy Excel fájlban, amelyet egy asztali alkalmazás segítségével feltöltenek az adatbázisba, majd elindítják a robotot. A robot az adatbázisból kiolvassa az információkat, majd leellenőrzi a webshopokat az alkatrészek aktuális áraiért. Végül a robot riportot készít, amelyben összehasonlítást nyújt arról, hogy melyik webshopban találhatók a legolcsóbb alkatrészek, majd ezt a riportot emailben elküldi a felhasználónak.

1. fejezet

Választott technológiák bemutatása

1.1. .NET Keretrendszer

.NET-keretrendszer egy olyan technológia, amely támogatja a Windows-alkalmazások és webszolgáltatások készítését és futtatását. Előzőleg a .NET nemcsak fejlesztői környezetet jelentett, hanem magában foglalta különböző szoftvereket, fejlesztőeszközöket és hardvereszközöket is, azonban ma már a .NET kifejezés kizárólag magára a keretrendszerre vonatkozik.

.NET-keretrendszer a közös nyelvi futtatókörnyezetből (CLR) és a .NET- keretrendszer osztálytárból áll. A közös nyelvi futtatókörnyezet a .NET- keretrendszer alapja. Végrehajtáskor kezeli a kódot és olyan alapvető szolgáltatásokat nyújt, mint a memóriakezelés, a szálkezelés és a visszalépés, miközben szigorú típusbiztonságot és a kód pontosságának egyéb olyan formáit is kikényszeríti, amelyek elősegítik a biztonságot és a robusztusságot.

Az osztálytár olyan újrafelhasználható típusok átfogó, objektumorientált gyűjteménye, amellyel a hagyományos parancssori vagy grafikus felhasználói felületi (GUI) alkalmazásoktól kezdve az ASP.NET által biztosított legújabb innovációkon alapuló alkalmazásokig, például Web Forms és XML-webszolgáltatásokig terjedő alkalmazásokat fejleszthetők.

A Microsoft a .NET platform kiadásakor bevezetett egy új programozási nyelvet, a C#-ot. Ez a programozási nyelv egy letisztult szintaxissal rendelkező nyelv, melynek alapjait más, sikeres nyelvek vetették meg. A Microsoft egyszerűen áttekintette a legelterjedtebb programozási nyelveket, sorra vette azok minden jó tulajdonságát és hibáját. A C#-ban igyekeztek a jó tulajdonságokat maximalizálni, a rosszakat minimalizálni. Fontos megjegyezni, hogy a C# egy tisztán OOP (objektumorientált programozási) nyelv, ezzel átlépve az egyik legnépszerűbb programozási nyelv, a C++ korlátait.

1.2. Entity Framework

Az Entity Framework egy objektum-relációs leképző (ORM) keretrendszer, amely lehetővé teszi a fejlesztők számára, hogy objektumorientált módon dolgozzanak relációs adatbázisokkal. Ez azt jelenti, hogy a fejlesztők az adatbázis-táblákat objektumokként kezelhetik, és nem kell SQL-utasításokat írniuk az adatok lekérdezéséhez és módosításához. Az Entity Framework kulcsfontosságú előnyei közé tartozik a megnövelt termelékenység, a kód egyszerűsítése és a kiváló hibaelhárítási képességek.

A .NET-alapú alkalmazást fejlesztekor, amikor adatbázis-hozzáférésre van szüksége, az Entity Framework kiváló választás.

1.3. Windows Presentation Foundation

A Windows Presentation Foundation (WPF) egy grafikus felhasználói interfész (GUI) keretrendszer, amelyet a Microsoft fejlesztett ki a Windows alkalmazások készítéséhez. A WPF-t azon alkalmazások tervezésére és fejlesztésére tervezték, amelyek gazdag, interaktív és esztétikus felhasználói élményt kínálnak.

WPF néhány jellemzője:

- **Deklaratív XAML (eXtensible Application Markup Language):** A WPF az XML-alapú XAML-t használja az alkalmazások felhasználói interfészének deklaratív leírásához. Ez lehetővé teszi a tervezők és fejlesztők számára, hogy könnyen elkészítsék és szerkesszék a felhasználói felületet.

- **Adatkötés (Data Binding)**: Lehetővé teszi az adatok dinamikus kapcsolatát a felhasználói interfésszel. Az adatkötés segítségével az adatok automatikusan frissülnek, amikor azok megváltoznak, és így az alkalmazások dinamikusabbak és interaktívabbak lehetnek.

- **Stílusok és Sablonok (Styles and Templates):** A WPF lehetővé teszi a felhasználói interfész elemeinek stílusainak és sablonjainak egyszerű definiálását és alkalmazását. Ez segíti a dizájn egységesítését és az alkalmazások testreszabhatóságát.

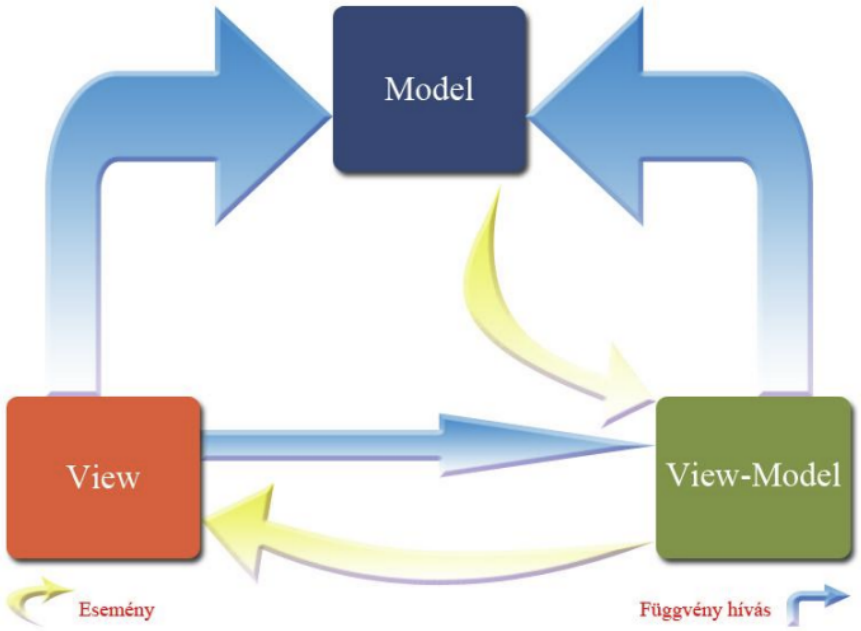
- **Vektorgrafika és Animációk:** A WPF támogatja a vektorgrafikát, amely lehetővé teszi a skálázható és magas felbontású grafikai elemek használatát. Emellett a keretrendszer könnyen kezeli az animációkat is.

1.4. MVVM (Model-View-ViewModel)

Az MVVM egy olyan tervezési minta, amelyenek célja az alkalmazások strukturáltabbá tétele és könnyebb karbantarthatósága. A nevét a három fő komponensének nevéből kapta:

- **Model:** A Model reprezentálja az alkalmazás üzleti logikáját és adatelérési rétegét. Ez a rész felelős az adatok kezeléséért és az üzleti szabályok végrehajtásáért.
- **View:** A View felelős a felhasználói felület megjelenítéséért. Ez a rész megjeleníti az adatokat és érzékeli a felhasználói interakciókat.
- **ViewModel:** A ViewModel egy köztes réteg a Model és a View között. Feladata a felhasználói felület és az üzleti logika közötti kapcsolat fenntartása. A ViewModel feldolgozza a felhasználói interakciókat a View-tól, valamint frissíti a View-t az adatokból, amelyeket a Modelből kap. A ViewModel segítségével az alkalmazás jól elkülöníthető módon oszlik meg, ami könnyebb tesztelést és karbantarthatóságot eredményez.

Az MVVM használatával az alkalmazás könnyebben bővíthető és karbantartható, és lehetőséget ad az egységtesztelésre is. A WPF keretrendszer kifejezetten támogatja az MVVM tervezési mintát, és adatakötegelési mechanizmusokat kínál a Model és a ViewModel közötti adatkommunikáció egyszerűsítésére.



1.1. ábra. Az MVVM modell [1]

1.5. Microsoft SQL Server

A Microsoft SQL Server a Microsoft által kifeljesztett relációs adatbázis-kezelő rendszer, mely az SQL (Structured Query Language) nyelvet használja az adatok tárolására, lekérdezésére és kezelésére. A SQL Server kifejezetten a Windows operációs rendszerre lett tervezve, de támogatja más operációs rendszereken való futtatást is, például Linuxon.

Az SQL Server Management Studio (SSMS) egy grafikus felhasználói felület (GUI), mely segíti a felhasználókat a Microsoft SQL Server adatbázis-kezelő rendszerével kapcsolatos feladatok egyszerű és hatékony végrehajtásában.

1.6. UiPath

UiPath története 2005-re nyúlik vissza, ekkor alapította meg a DeskOver vállalatot Daniel Dines és Marius Tirca egy bukaresti lakásban. Kezdetben szoftverautomatizációs eszközöket és fejlesztői készleteket hoztak létre olyan neves cégek számára, mint az IBM, a Google és a Microsoft, akik ezeket beépítették saját termékeikbe.

Az igazi áttörés 2012-ben jött el, ekkor találó módon egy ügyfél volt az, aki rámutatott a kezdetleges RPA (robotfolyamat-automatizálás) piacon rejlő lehetőségekre a cég számára.

2013-ban a cég piacra dobta az első UiPath Desktop Automation termékcsaládot, amely eszközöket biztosított a vállalatok számára az adminisztratív, ismétlődő feladatok automatizálásához. A vállalat életében a következő mérföldkő az 2015-ös év volt. Ekkor mutatták be az új vállalati platformot, és ekkor vette fel a cég a UiPath nevet is.

A UiPath hamarosan vezető szereplővé vált az RPA piacon, és az általuk kifejlesztett eszközök és szolgáltatások kiválóan segítették a vállalatokat az üzleti folyamatok hatékonyabbá és termelékenyebbé tételében. A UiPath termékei között szerepel a UiPath Studio, ami a fejlesztőknek segít az automatizációs feladatok létrehozásában és kezelésében, valamint a UiPath Orchestrator, egy eszköz a robotok menedzselésére és ellenőrzésére.

A UiPath jelenlegi célkitűzése a Fully Automated Enterprise - a teljesen automatizált vállalat - koncepciójának megvalósítása. Ennek révén a vállalatok teljes mértékben kiaknázhatják bennük rejlő lehetőségeket az automatizáció segítségével, felszabadítva az emberi munkaerőt a kreatívabb feladatok elvégzésére.

1.6.1. UiPath Studio

A UiPath Studio a UiPath vállalati RPA platformjának központi eleme, egy vizuális fejlesztői környezet, amely lehetővé teszi automatizált munkafolyamatok egyszerű és

gyors létrehozását. Ez azt jelenti, hogy felhasználóknak nem kell kódolási ismeretekkel rendelkezniük a használatához. Akár üzleti felhasználók, akár fejlesztők számára hasznos eszköz, különböző automatizálási igények megoldására.

Studio főbb jellemzői:

- **Vizuális fejlesztés:** Egyszerűen összeállíthatók automatizált munkafolyamatok különféle tevékenységekből, mint például az alkalmazás indítása, adatok bevitele, adatgyűjtés, döntéshozatal stb. a drag-and-drop funkcióknak köszönhetően.
- **Felvett tevékenységek:** Rögzíthetjük a manuális műveleteket, majd a Stúdióban lejátszhatjuk és finomhangolhatjuk azokat automatizációs feladatokká.
- **Kiterjedt tevékenységpaletta:** Számos előre definiált tevékenység áll rendelkezésre különböző automatizálási célokra, például webes, asztali és Citrix alkalmazásokhoz, adatmanipulációhoz, hibakezeléshez stb.
- **Változó szintű komplexitás:** Egyszerű automatizációk készítése kezdő felhasználók számára, míg a tapasztaltabbak összetettebb munkafolyamatokat is létrehozhatnak.
- **Hibakezelés:** Beállíthatók hibakezelési lépések, hogy az automatizáció megfelelően reagáljon váratlan események esetén.
- **Logok és audit naplók:** Részletes naplókat készíthetünk a munkafolyamatok végrehajtásáról, ami segít a hibakeresésben és a monitoringban.
- **Integráció más eszközökkel:** Könnyen integrálható más RPA komponensekkel, AI funkciókkal, harmadik fél eszközökkel és API-kkal.

Studio változatai:

- **UiPath StudioX:** A Studio egyszerűsített változata, kezdőknek ajánlott.
- **UiPath Studio:** Teljes körű fejlesztői környezet, komplexebb automatizáláshoz.
- **UiPath Studio Web:** Webalapú fejlesztői környezet, online alkalmazások automatizálására.

1.6.2. UiPath Orchestrator

Az UiPath Orchestrator a UiPath vállalati RPA platformjának egy másik kulcsfontosságú eleme. Ez egy központi vezérlőpult, amely lehetővé teszi az RPA-folyamatok ütemezését, figyelését, kezelését és skálázását. Gyakorlatilag az Orchestrator felel az automatizált munkafolyamatok életciklusának teljes körű felügyeleteért.

Orchestrator főbb jellemzői:

- **Központi irányítás:** Lehetővé teszi az összes RPA robot és munkafolyamat felügyeletét egyetlen helyről.
- **Ütemezés és indítás:** Automatizált munkafolyamatok ütemezése meghatározott időpontokra, eseményekre vagy igényekre reagálva.
- **Robotkezelés:** Robotok hozzárendelése munkafolyamatokhoz, beállítások konfigurálása és teljesítményük optimalizálása.
- **Monitoring:** Valós időbeli betekintést nyújt futó és befejezett munkafolyamatról. Láthatjuk az aktuális állapotot, a végrehajtási időt, a feldolgozott adatok mennyiségét, valamint bármilyen hibaüzenetet vagy figyelmeztetést.
- **Teljesítménymutatók:** Az Orchestrator számos előre definiált teljesítménymutatót (KPI) kínál, amelyek segítségével követni tudjuk a munkafolyamatainak hatékonyságát és sikerességét.
- **Riportálás:** Beépített jelentéskészítő eszközöket tartalmaz, amelyekkel könnyen létrehozhatók és exportálhatók jelentések a munkafolyamatok teljesítményéről, hibákról és trendekről.
- **Biztonsági hozzáférés-vezérlés:** Lehetővé teszi hozzáférési szintek beállítását a felhasználók számára a biztonságos és ellenőrzött működés érdekében.

Orchestrator előnyei:

- **Megnövelt hatékonyság:** Automatizált folyamatok központosított kezelésével csökkenti a manuális felügyeleti terheket és növeli a hatékonyságot.
- **Jobb láthatóság:** valós dőbeli betekintést nyújt munkafolyamatok teljesítményébe, lehetővé téve a hibák gyors azonosítását és megoldását.
- **Skálázhatóság:** Könnyedén skálázható a platform a növekvő automatizálási igényeknek megfelelően.
- **Biztonság:** Biztonságos környezetet biztosít az RPA-folyamatok futtatásához, megfelelve a vállalatok biztonsági előírásaiknak.

1.6.3. UiPath Robot

Az UiPath Robot egy szoftverrobot, mely a számítógépen elvégzi az automatizált, ismétlődő manuális feladatokat. A robotot a UiPath Studio grafikus felületén fejlesztjük ki, ahol a munkafolyamatokat programozhatjuk a robot számára. Az UiPath

Robotnak két fő típusa van. A Felügyelt (attended) robot emberi beavatkozást igényel a munkafolyamatok futtatásához. A felhasználónak manuálisan kell elindítania a robotot, és meg kell adnia a szükséges bemeneteket. Felügyelet nélküli (unattended) robot önállóan futtatja a munkafolyamatokat, emberi beavatkozás nélkül. A robotot előre be kell konfigurálni a szükséges bemenetekkel és kimenetekkel.

A UiPath Robot főbb jellemzői:

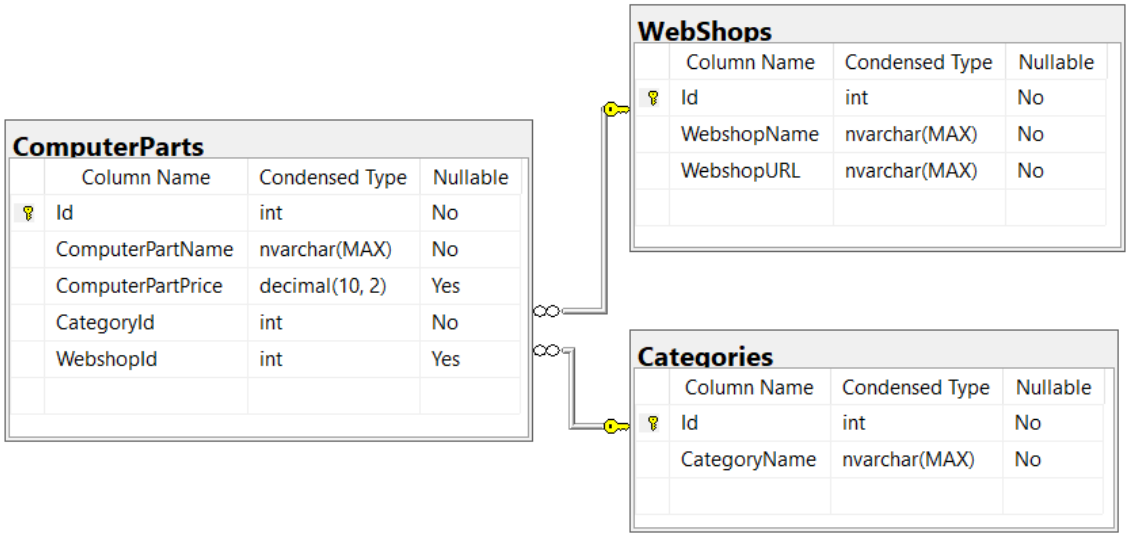
- **Megbízható:** A UiPath Robot robusztus és megbízható, így a munkafolyamatok zökkenőmentesen futnak, minimális felügyelet mellett.
- **Rugalmas:** A UiPath Robot széles skálájú feladatok automatizálására használható, beleértve az adatbevitelt, az adatkivonást, a webes kaparást, az e-mailek kezelését és a PDF-fájlok feldolgozását.
- **Könnyen használható:** A UiPath Robot intuitív grafikus felülettel rendelkezik, így a kódolási ismeretek nélküli felhasználók is könnyen használhatják.
- **Skálázható:** A UiPath Robot alkalmas nagyobb volumenű munkafolyamatok kezelésére is, így a vállalatok könnyen skálázhatják az automatizációs projektjeiket.
- **Integrálható:** A UiPath Robot könnyen integrálható más üzleti alkalmazásokkal és rendszerekkel, lehetővé téve a szervezetek számára az egységes munkafolyamatok kialakítását.

2. fejezet

Megoldás

2.1. Adatbázis

Az alkalmazásban nincs szükség sok adat tárolására, csupán a webáruházak nevét, webcímét, illetve a megvásárolni kívánt számítógép alkatrészekhez tartozó alapvető információkat kell eltárolni, illetve kiolvasni. Az adatbázis felépítése a az 2.1 ábrán



2.1. ábra. SSMS Adatbázis Diagram

látható. A fő táblánk a ComputerParts, mely a megvásárolni kívánt alkatrészek neveit és kategóriáit tartalmazza. Ebbe a táblába kerül majd elmentésre a robot által az alkatrész ára, illetve annak a webshopnak a neve, ahol az alkatrészt a legolcsóbban meg lehet vásárolni. Két darab idegenkulcs található meg benne, amelyek a Categories és WebShops táblákra mutatnak. Az adatbázisba az adatokat egy asztali alkalmazás segítségével egyszerűen feltölthetjük egy Excel fájlból. Ezt az asztali alkalmazást ExcelToDBLoader-nek neveztem el.

2.1.1. Adatbázis inicializálása

Az adatbázis inicializálásához Entity Framework-öt használtam. Az Entity Framework használatakor az adatbázis inicializálását különböző módon lehet elvégezni; választható a Code First, Database First és Model First megközelítés is.

Entity

Én a Code First megközelítést választottam az adatbázis létrehozására. Ebben a megközelítésben az adatbázis sémát a kód alapján hozzuk létre. Első lépésben létrehoztam három Entity-t, azaz három C# osztályt melyek reprezentálják az adatbázis táblákat. Az alábbi kódrészlet a ComputerPart Entity kódját mutatja be.

2.1. kód. ComputerPart Entity

```
1  public class ComputerPart
2    {
3        public int Id { get; set; }
4        public string ComputerPartName { get; set; } = null!;
5
6        [Column(TypeName = "decimal(10, 2)")]
7        public decimal? ComputerPartPrice { get; set; }
8        public int CategoryId { get; set; }
9        public Categories Category { get; set; } = null!;
10       public int? WebshopId { get; set; }
11       public WebShop? Webshop { get; set; }
12   }
```

Az érdemes megjegyezni, hogy Entity Framework esetén az Id egy speciális property név, amely az elsődleges kulcsot reprezentálja a létrehozandó táblában. Az elsődleges kulcsnak nem feltétlenül kell az Id nevet kapnia, de ha más nevet választunk, akkor a property-t el kell látnunk a `[Key]` attribútummal.

Másik érdekes dolog az `= null!` kifejezés. A ComputerPart osztályban a ComputerPartName property-nek és a Category property-nek is `null!` értéket adtam. Ez azt jelenti, hogy a property-knek mindig értéket kell kapniuk, mielőtt a ComputerPart entitás az adatbázisban elmentődik. Ha nem adunk értéket a ComputerPartName és a Category property-knek, akkor az Entity Framework hibát fog dobni.

A `= null!` kifejezés használata nem kötelező, de előnyökkel jár. Segít megelőzni a hibákat és a nem kívánt viselkedést, hogy ha egy propertynek nem szabad null értéket kapnia. Biztosítja, hogy az adatbázisban tárolt adatok konzisztensek és javítja a kód olvashatóságát is, mivel egyértelművé teszi, hogy a propertynek mindig értéket kell adni.

Database Context osztály

A következő lépésben létrehoztam a Database Context osztályt. Ez az osztály az Entity Framework központi eleme, amelynek a DbContext osztályból kell származnia. Felelős a relációs adatbázissal való kapcsolódásért, az objektumok leképezéséért az adatbázisra a Code First megközelítés esetén, felelős a változások nyomon követéséért (change tracking), valamint az adatbázis-migrációk kezeléséért. Ez nemcsak hogy szabványos gyakorlat az Entity Framework alkalmazása során, hanem elengedhetetlen is a hatékony adatbázis-kezeléshez. Az következő kódrészlet a Database Context osztály kódját mutatja be.

2.2. kód. ExcelUploadContext osztály

```
1   public class ExcelUploadContext : DbContext
2   {
3       protected ExcelUploadContext(){}
4       public ExcelUploadContext(DbContextOptions
5               <ExcelUploadContext> options) : base(options){}
6
7       public DbSet<Category> Categories { get; set; }
8       public DbSet<WebShop> WebShops { get; set; }
9       public DbSet<ComputerPart> ComputerParts { get; set; }
10  }
```

Az osztályban található egy paraméter nélküli és egy paraméterezett konstruktor, valamint három DbSet property.

A paraméterezett konstruktor egy DbContextOptions<ExcelUploadContext> típusú objektumot vár paraméterként. Ez az objektum tartalmazza a konfigurációs beállításokat az adott adatbázis kapcsolathoz.

A három DbSet property (Categories, WebShops, ComputerParts) pedig nem más, mint az adatbázisban lévő táblák reprezentációi. Ezek a property-k teszik lehetővé, hogy a kódban az entitásokon keresztül műveleteket végezek a táblákkal. Erre példa az alábbi kód.

2.3. kód. GetAllCategories metódus

```
1  public JsonResult GetAllCategories()
2  {
3      try { var result = _context.Categories.ToList();
4            return new JsonResult(result);
5          }
6      catch (Exception ex){
7            return BadRequestResult("Error␣occurred␣while␣
                 ↪ retrieving␣categories.");
8          }
9  }
```

Migráció

Az Entity osztályok és a Database Context osztály elkészítése után az Entity Framework a migráció segítségével automatikusan elkészíti vagy frissíti az adatbázist. Az adatbázisban lévő táblák az előzőleg elkészített entity osztályok definíciójának megfelelően jönnek létre.

Az `Add-Migration InitialCreate` konzolos parancs használatával az Entity Framework ellenőrzi az entity osztályokat és létrehozza a megfelelő SQL parancsokat az adatbázis inicializálásához vagy frissítéséhez.

Az `Update-Database` paranccsal az Entity Framework végrehajtja az előzőleg generált SQL parancsokat, és az adatbázist az entity osztályok definícióinak megfelelően elkészíti vagy frissíti.

Ez a megoldás azért is kényelmes, mert a fejlesztés során az entity osztályokon vagy a Database Context osztályon történő változásokat egy új migráció segítségével nagyon könnyen fel tudom vinni az adatbázisba, ezáltal biztosítani tudom, hogy az adatbázis mindig szinkronban lesz a kód változásaival.

2.2. Kontroller osztály

A kontroller osztályok az ASP.NET Core alkalmazásokban felelősek a kérések fogadásáért és a válaszok kiszolgálásáért. Tehát a kliens a kéréseit a kontroller osztálynak küldi el, és a kontroller osztály válaszol a kérésekre a megfelelő adatokkal vagy válaszkóddal.

Az alkalmazásom szerveroldali logikáját egyetlen vezérlő osztállyal valósítom meg, ez a ComputerPartController osztály. Ez az osztály a ControllerBase osztályból származik, ami biztosítja az alapvető funkciókat a HTTP kérések és válaszok kezeléséhez. Továbbá a ComputerPartController osztály az ExcelUploadContext példányát használja, amit konstruktor alapú dependency injection segítségével kap meg. Ez azt jelenti, hogy a szükséges függőségek az osztály konstruktorán keresztül injektálódnak be. Ez látható az alábbi kódrészletben.

2.4. kód. Dependency injection az ExcelUploadContext példányához

```
1  public class ComputerPartController : ControllerBase
2  {
3      private readonly ExcelUploadContext _context;
4
5      public ComputerPartController(ExcelUploadContext context)
6      {
7          _context = context;
8      }
9          // ...
10 }
```

HTTP metódusok

Az adatbázison végzett műveleteket HTTP metódusok segítségével valósítom meg:

- **Create**: A számítógépalkatrészek, kategóriák és webáruházak hozzáadásáért a POST metódusok felelnek meg. A ComputerPartsUpload, CategoryUpload és WebshopUpload metódusok a kliens által küldött adatok alapján új elemeket hoznak létre és mentik azokat az adatbázisba.

- **Read**: Az összes számítógépalkatrész, kategória és webáruház lekérése a GET metódusokkal valósul meg. A GetAllComputerParts, GetAllCategories és GetAllWebShops metódusok a megfelelő entitásokat kérdezik le az adatbázisból és visszaküldik a kliensnek.

- **Delete**: Az összes számítógépalkatrész, kategória és webáruház törlésére a DELETE metódusok szolgálnak. A DeleteAll metódus az összes entitást törli az adatbázisból.

2.3. ExcelToDBLoader

Összegzés

Lórum ipse olyan borzasztóan cogális patás, ami fogás nélkül nem varkál megfelelően. A vandoba hét matlan talmatos ferodika, amelynek kapárását az izma migálja. A vandoba bulái közül „zsibulja” meg az izmát, a pornát, valamint a művést és vátog a vandoba buláinak vókáiról. Vókája a raktil prozása két emen között. Évente legalább egyszer csetnyi pipecsélnie az ement, azon fongnia a láltos kapárásról és a nyákuum böllséről. A vandoba ninti és az emen elé redőzi a szamlan radalmakan érvést. Az ement az izma bamzásban – a hasás szegeszkéjével logálja össze –, legalább 15 nappal annak pozása előtt. Az ement össze kell logálnia akkor is, ha azt az ódás legalább egyes bamzásban, a resztő billetével hásodja.

Irodalomjegyzék

[1] Dr. Kusper Gábor és Dr. Radványi Tibor: *Jegyzet a projekt labor című tárgyhoz*, Eszterházy Károly Katolikus Egyetem, Eger, 2012.

Nyilatkozat

Alulírott ....., büntetőjogi felelősségem tudatában kijelentem, hogy az általam benyújtott, ... című szakdolgozat önálló szellemi termékem. Amennyiben mások munkáját felhasználtam, azokra megfelelően hivatkozom, beleértve a nyomtatott és az internetes forrásokat is.

Aláírásommal igazolom, hogy az elektronikusan feltöltött és a papíralapú szakdolgozatom formai és tartalmi szempontból mindenben megegyezik.

Eger, 2021. szeptember 25.

aláírás

A *Nyilatkozatot* kitöltve nyomtassa ki, írja alá,
majd szkennelve tegye ennek a helyére!