A projekt az Európai Unió támogatásával és az Európai Szociális Alap társfinanszírozásával valósul meg (támogatási szerződés száma TÁMOP 4.2.1/B-09/1/KMR-2010-0003)

Eötvös Loránd Tudományegyetem
Informatikai Kar
Térképtudományi és Geoinformatikai Tanszék

Egy digitális térképtár megvalósítási lehetőségei a Mapserver eszköz használatával

Simó Benedek
térképész szakos hallgató

Témavezető:

Elek István
Egyetemi docens

Budapest, 2011

I. Bevezetés

Diplomamunkám célja, hogy egy olyan adatbázis alapú térkép-nyilvántartó rendszert hozzak létre, mely minden tekintetben kielégíti a felhasználói csoportok által támasztott igényeket, és ennek megvalósítását lehetőleg ingyenesen használható szoftverek segítségével oldjam meg.

A szoftverpiac folyamatosan bővülő kínálatában már egyre inkább jellemző az egyes programok ingyenes hozzáférési lehetősége. Ez egy rendkívül fontos jelenség, hiszen így fennáll annak a lehetősége, hogy mindenféle jelentősebb anyagi ráfordítás nélkül tudjunk egyszerű vagy bonyolultabb problémákat megoldani a térképek és adatbázisok témakörén belül. Néhány évvel ezelőtt még szinte elképzelhetetlen volt az, hogy egy komolyabb adatbázis-kezelő rendszer vagy szoftver mindenki számára elérhető legyen, de napjainkra már maga a Microsoft cég is ezt az utat választotta, és ennek következtében az SQL Server különböző verziói és a hozzájuk kapcsolódó szoftveres termékek egész csomagja vált elérhetővé mindazok számára, akik adatbázis-kezelő rendszereket szeretnének használni.

Meg kell említenünk még a geoinformatikával kapcsolatos különböző olyan szervezeteket és társulásokat, akik szándékosan azt a célt tűzték ki maguknak, hogy az általuk fejlesztett szoftverek mind nyílt forráskódúak legyenek, és hogy ebből kifolyólag egy egész felhasználói közösség legyen az, aki ezeket a szoftvereket fejleszti. Ilyen közösség például az Open Source Geography nevű társulás, melyhez számos termék mellett a nagy sikerre szert tett MapServer is kapcsolódik. Ez az eszköz talán az egyik legfontosabb eleme lehet a napjainkban gyorsan fejlődő webkartográfiának, és habár a fejlesztésben jellemzően informatikusok vesznek részt, kell, hogy a térképészek is legalább alapszinten kezelni tudják ezt az eszközt, hogy későbbi munkájuk során majd hasznosítani tudják.

II. Háttér és alapanyagok

A kartográfiában évszázadokra visszamenőleg a papírt alkalmazták a térbeli adatok megjelenítéséhez. Ez napjainkra átalakulni látszik, hiszen a kézzelfogható térképek mellett egyre inkább jelentkezik az igény, hogy a térképek digitális formában, illetve webes felületen is elérhetőek legyenek, és hogy ezeket szabadon böngészhesse a térképekre kíváncsi felhasználó. A meglévő papírtérképek egy szkennelési eljárás során könnyen átalakíthatóak digitális állományokká, és ez megteremti a lehetőséget a számítógépen, illetve a világhálón való megjelenítéshez. A korábbi idők térképeit térképtárak őrzik, de a hagyományos tárolásból és a papír öregedésből fakadó káros hatások miatt az ezekben tárolt anyagok sokszor jelentős értékcsökkenésen mennek keresztül és gyakran olvashatatlanná válnak. Ezzel szemben egy esetleges digitalizálást követően ezek a térképek egy adott pillanathoz rögzített állapotban maradnak fenn az utókornak .

Mivel egy egész térképtár digitalizálását követően ez rendszerint egy nagyszámú állományt jelent, és minden egyes térképhez rengeteg kapcsolódó adat is társul, érdemes ezekhez az adatokat adatbázisba szervezni. Ez jelentősen megkönnyíti az egyes térképekhez kapcsolódó adatok kezelését, illetve kritérium-feltételek meghatározásával lényegesen leegyszerűsíti egy tetszőleges tulajdonsággal rendelkező térkép megtalálását.

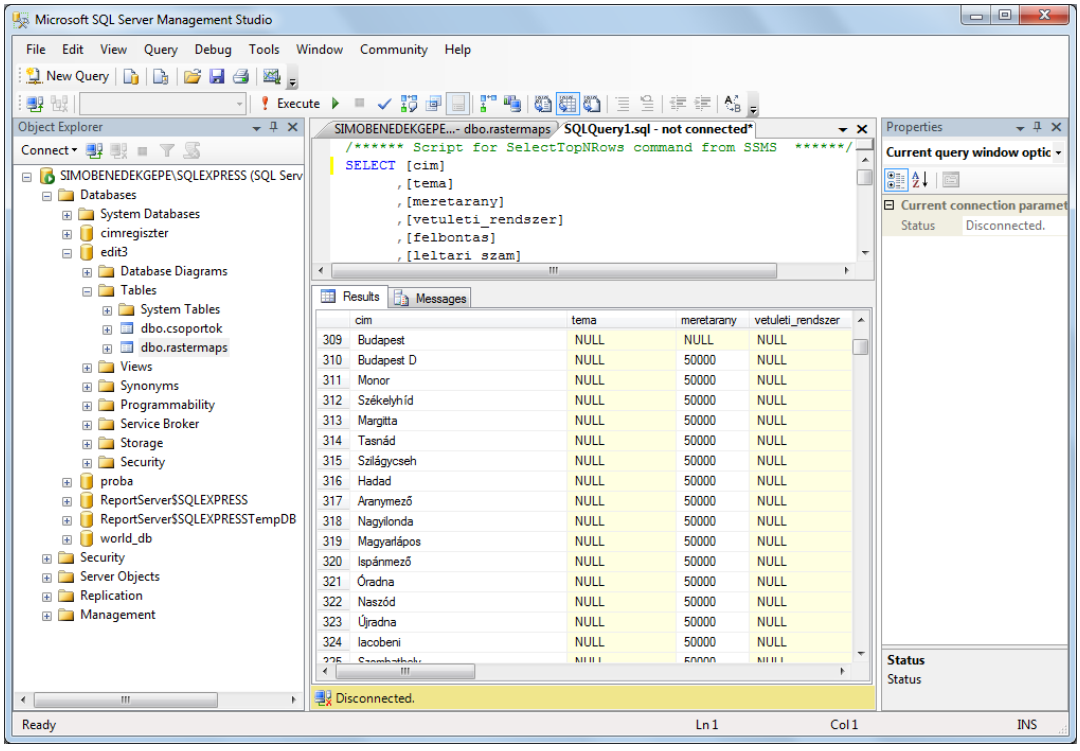
A fent leírtak értelmében tehát két alapvető eleme van egy digitális térképtárnak. Tetszőleges minőségben digitalizált, raszteres képi állományok, illetve egy adatbázis, melyben az egyes rekordok egy-egy térképet képviselnek, míg az adatbázis oszlopaiban az egyes elemekhez kapcsolódó attribútumok jelennek meg.

III. A digitális térképtárhoz kapcsolódó adatbázis

III.1. Adatbázis-kezelő kiválasztása

A térképeinkről tehát egy adatbázist szeretnénk készíteni, melyben az egyes rekordok adott elemhez tartoznak. Például tárolni szeretnénk a térképünk címét, méretarányát, vetületi rendszerét stb., hogy aztán könnyen hozzáférhessünk az adott tulajdonságú elemekhez. Ez egy elég egyszerű táblaszerkezetet jelent, mely nagyban hasonlít egy egyszerű táblázathoz, de abban azért mégis különbözik, hogy egy adott szerverre feltöltve az adatbázisunk adatai elérhetővé válnak a világháló irányából is.

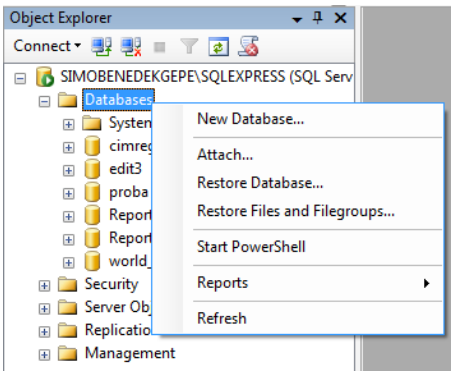
Az adatkezeléshez válasszunk tehát egy adatbázis-kezelőt. Mivel később majd a Visual Studio 2010 szoftvert, valamint a .NET keretrendszert használjuk a webes felület kialakításához, ezért célszerű egy olyan adatbázis-kezelőt választanunk, mely szintén a Microsoft cég termékei közé tartozik, és így biztosítva lehetünk a két szoftver közötti átjárhatóságról. A Microsoft cég a szerveren található adatbázis kezeléséhez a Microsoft SQL Server Management Studio eszközt biztosítja, melyben egy grafikus felületen kezelhetjük a tábláinkat, és a bennük található adatokat.



1. ábra: A Management Studio kezelőfelülete

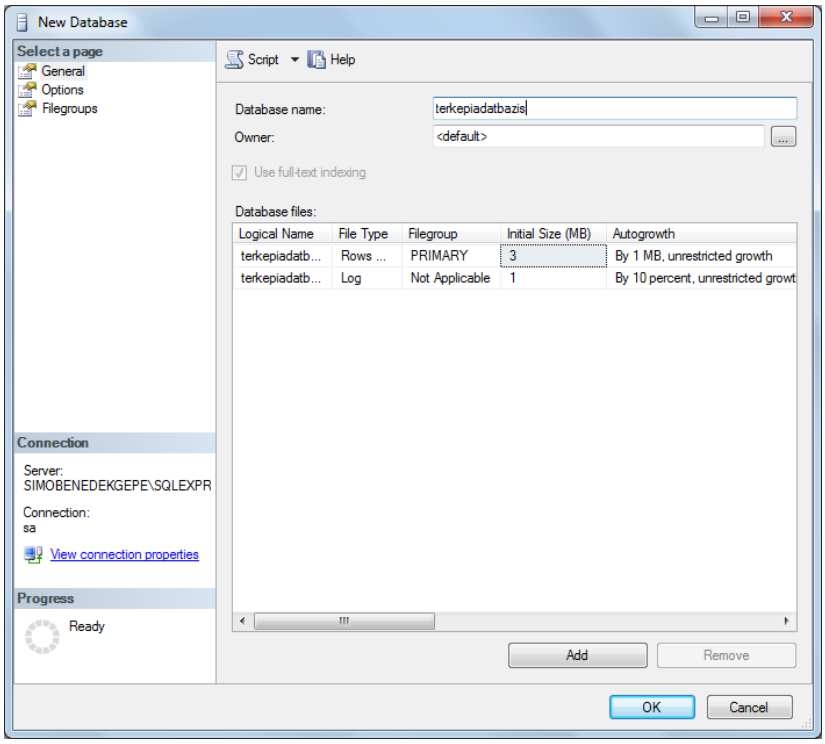
III.2. Új adatbázis létrehozása

Új adatbázis létrehozásához a baloldali panelon kattintsunk jobb egérrel a ***Databases*** elemre, majd válasszuk a ***New database…*** elemet.



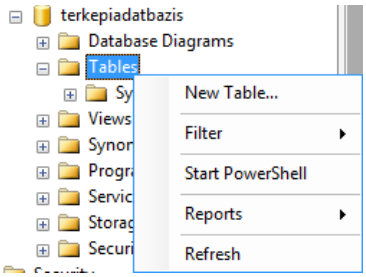
2. ábra: új adatbázis létrehozása

A felugró ablakban adjuk meg az adatbázisunk nevét:



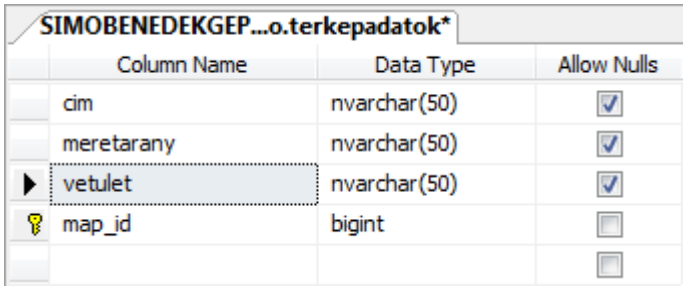
3. ábra: Az adatbázis nevének meghatározása

Ha létrehoztuk az adatbázisunkat, akkor adjunk hozzá egy táblát, melyben majd az adatokat fogjuk tárolni. Ehhez az ***Object Explorerben*** adatbázisunkat kibontva a ***Tables*** gyűjteményre kattintsunk, majd az ***New Table…*** lehetőségre.



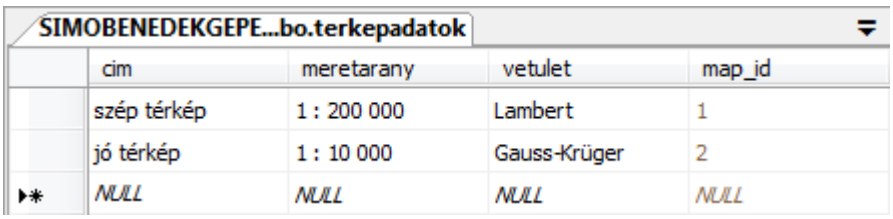
4. ábra: Új tábla létrehozása

Ha létrejött a táblánk, akkor most már csak az egyes adatmezőket és típusaikat kell meghatározzuk a megnyíló grafikus felületen. A legalsó, üres sorra kattintva mindig újabb mezőnevet tudunk felvenni, a ***Data type*** mezőben pedig legördülő menüből választhatjuk ki az adott mező adattípusát. Fontos megjegyezni, hogy a táblánk létrehozásakor mindenképpen kell egy kapcsolómezőt létrehoznunk, mely később majd kapcsolatot teremt a térképeket raszteresen tároló fájlok és az adatbázis között. Ha ebben a mezőben például a 147-es azonosító található az adott rekordhoz kapcsolódóan, akkor a fájlnév is célszerű, hogy 147.*** legyen. Ez tehát a táblánk ***id*** mezője lesz, mely egyben a táblánk elsődleges kulcsa is lesz. Ha ezzel elkészültünk, elmentjük a táblánkat, és nevet adunk neki.



5. ábra: Az adatmezők létrehozása

Ha létrejött a kívánt táblaszerkezetünk, akkor már csak megfelelő adatokkal kell feltöltenünk az adattáblánkat, melyet a táblánkra jobb egérrel kattintva ***Edit Top 200 Row…*** paranccsal tehetünk meg.



6. ábra: Az adatok feltöltése

III.3. Meglévő adatbázis használata

A valóságban sokszor fordul elő, hogy a térképeket leíró adatbázis már rendelkezésre áll. Ekkor többféle lehetőség állhat elő. Ha az adatbázis az általunk preferált adatbázis-kezelőn található, akkor ez a legszerencsésebb eset, és már csak „használnunk” kell az adatbázisunkat.

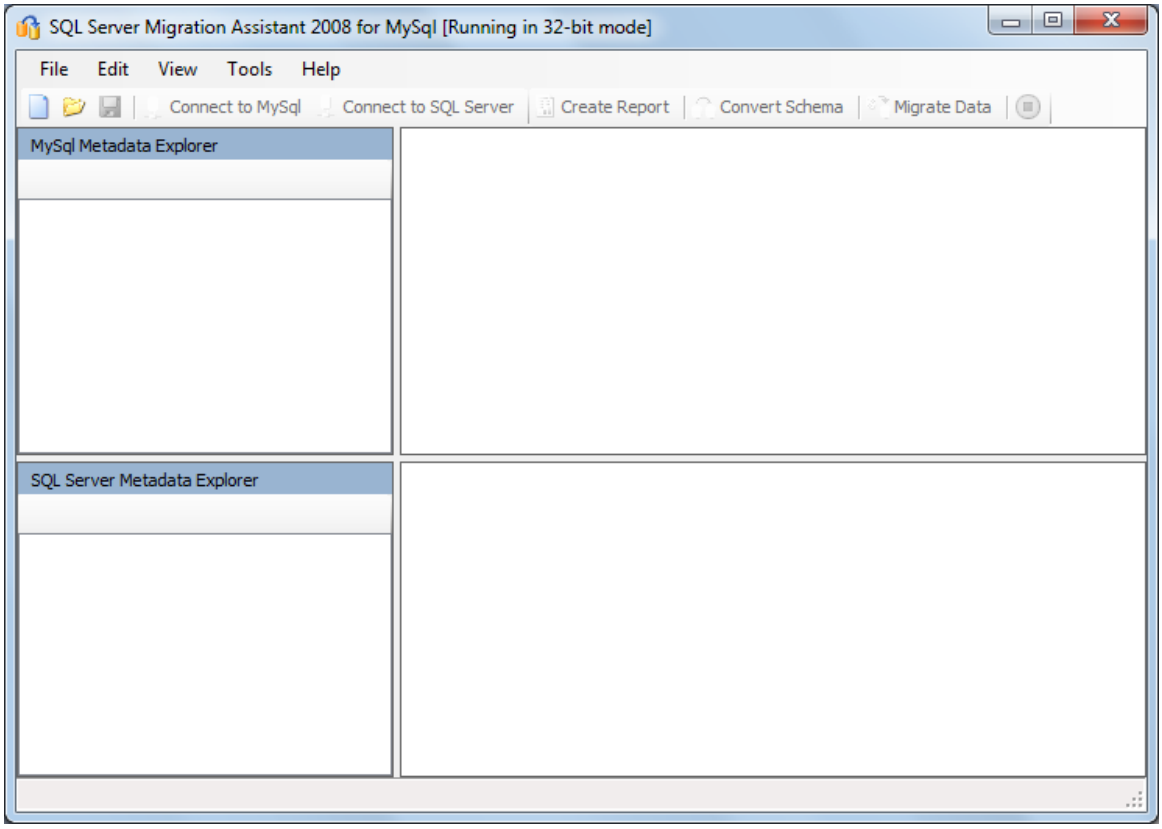
Gyakran megesik azonban, hogy az adatbázisunk nem a nekünk megfelelő adatbázis-kiszolgálón található. Ez azt jelentheti, hogy az adatbázisunk ugyan megvan, de mondjuk egy másik adatbázis-kezelő formátumában és nem a Microsoft SQL Server adatszerkezetében. Ilyen másik adatkezelő lehet a MySQL, a PostgreSQL, az Oracle stb. Ezeknek az adatbázis-kezelőknek sokszor elérhető az egymás közötti konverziót elősegítő eszköze, melyek viszonylag egyszerűen lehetővé teszik az adatbázisunk konverzióját, de természetesen az is elképzelhető, hogy az átalakítás csak igen keserves munka árán valósítható meg. A következőkben egy ilyen konverziós eszköz alapvető funkcionalitásának bemutatására kerül sor.

III.4. Egy adatbázis konverziós eszköz használatának a bemutatása

Az adatbázisok konverziós lehetőségeinek a bemutatására egy olyan eszközt használunk, mely a MySQL adatkezelőn található tetszőleges adatbázist alakítja át, illetve helyezi át az MS SQL Serverre. Ez az eszköz a Microsoft SQL Server Migration Assistant 2008 for MySQL.

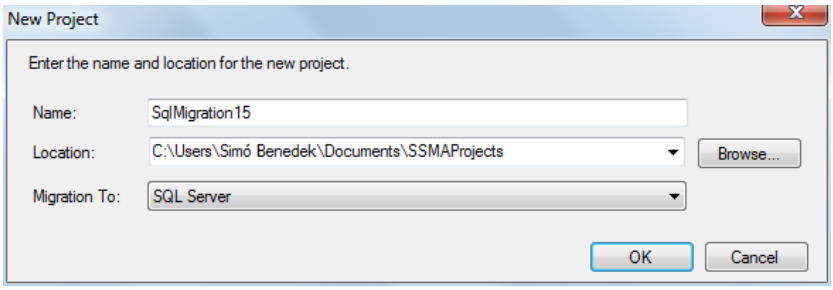
A program használata előtt meg kell vizsgálnunk, hogy milyen előfeltételei vannak a sikeres működésnek. A számítógépünkön fel kell legyen telepítve egy MySQL szerver, valamint egy MS SQL Server 2008, és mindkét szerverhez meg kell legyenek a megfelelő kezelői jogosultságaink.

Ha az előfeltételek sikeresen teljesültek, elindíthatjuk a programot, és megismerkedhetünk a viszonylag egyszerű felhasználói felülettel.



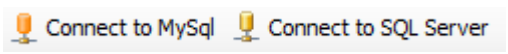
7. ábra: a konverziós eszköz felhasználói felülete

Elsőként egy új projektet kell létrehoznunk, melyet a ***File/New project…*** menüre kattintva tehetünk meg. Itt meg kell adnunk egy új nevet, illetve hogy milyen adatkezelőbe szeretnénk átalakítani a MySQL adatbázisunkat.



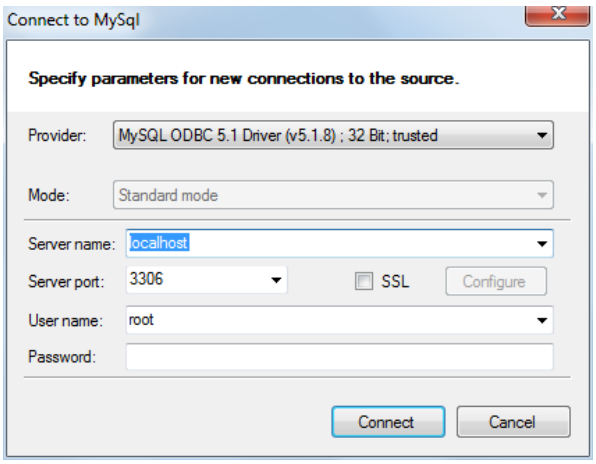
8. ábra: Új projekt létrehozása

Ha sikerült létrehoznunk a projektünket, akkor következő lépésként az egyes adatbázis-kezelőkhöz kell kapcsolódnunk. Ezt tegyük meg mindkét adatbázis-kezelő, a MySQL és az MS SQL esetében is.



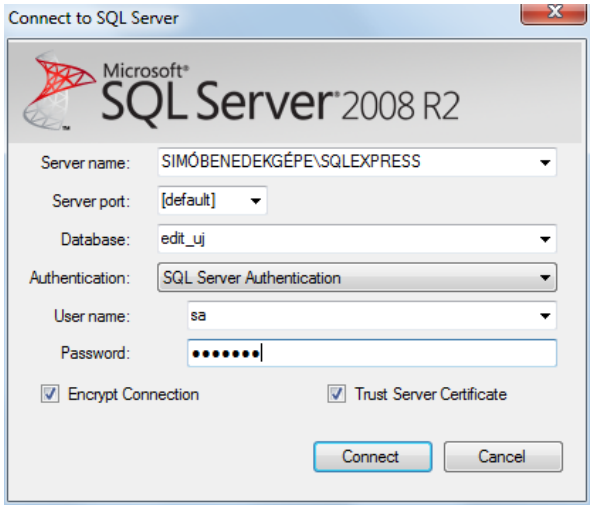
9. ábra: A kapcsolódáshoz szükséges ikonok

A MySQL-hez való csatlakozáshoz kattintsunk a fenti ábrán található ikonra, majd a felugró ablakban adjuk meg a szerverhez kapcsolódó felhasználónevet és jelszót.



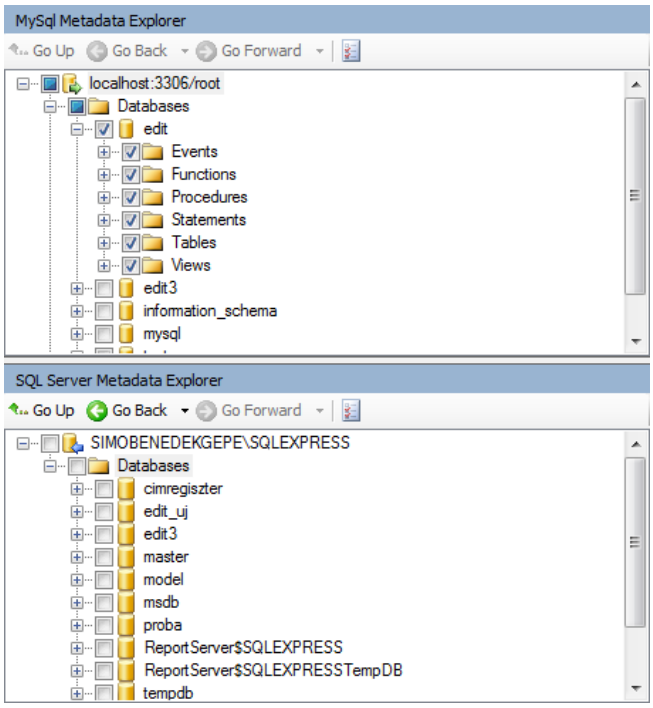
10. ábra: A MySQL szerverre való bejelentkezés

Hasonlóképpen csatlakozzunk az MS SQL-hez is.



11. ábra: Csatlakozás az MS SQL szerverhez

Ha létrejött a kapcsolat mindkét adatbázis-kiszolgálóval, akkor megtekinthetjük az adatbázisainkat mindkét platformon, és tetszőlegesen böngészhetjük a tábláinkat. Mindezt egy ún. ***Metadata explorer*** ablakban tehetjük meg, mely mindkét kiszolgáló esetében rendelkezésre áll.

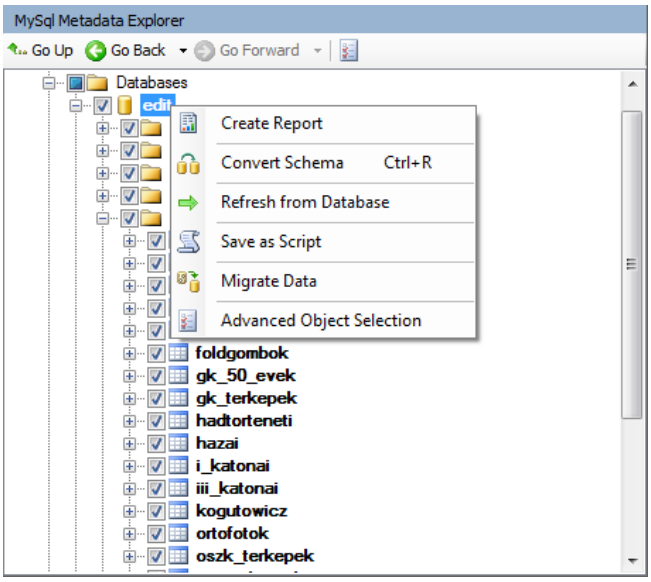


12. ábra: A Metadata Explorer ablakok

Az átalakítás során tehát a MySQL adatkiszolgálón található adatbázisok közül szeretnénk egyet áthelyezni az MS SQL szerverre. Miután kiválasztottuk a kívánt adatbázist, váltsunk át az SQL Server Management Studio-ra és hozzunk létre az átalakítandó/áthelyezendő adatbázis nevével megegyező üres adatbázist az MS SQL szerveren az előző fejezetben leírtak szerint (ezt mindenképpen meg kell tegyük a következő lépések végrehajtása előtt).

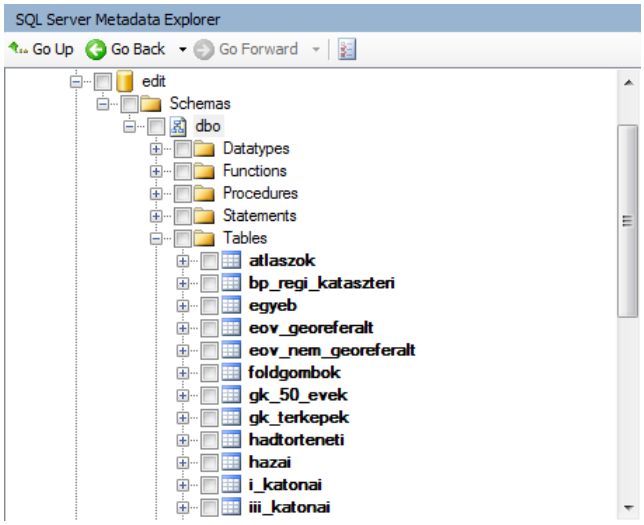
Az átalakítás, illetve áthelyezés során alapvetően két egymást követő lépést kell elvégeznünk. Az első lépés az ún. séma (idegen szóval schema) átmásolása, míg a második lépés maguknak az adatoknak az átmásolása.

Elsőként vizsgáljuk meg hogy miként helyezhező át a sémánk. (És hogy mi is a séma? A séma kifejezés ez esetben egy olyan fogalmat takar, mellyel az adatbázisunk szerkezetét, felépítését jellemezhetjük, vagyis ez főként arra vonatkozólag tartalmaz információt, hogy az adatbázisunk milyen és mennyi táblát tartalmaz, illetve hogy ezek milyen asszociációval randelkeznek egymás irányába.) Ennek az átmásolásához pipáljuk ki a MySQL Metadata Explorerben az adatbázisunk előtti jelölőnégyzetet, majd kattintsunk jobb egérrel az adatbázisunkra, és válasszuk a ***Convert Schema*** lehetőséget.



13. ábra: A táblaszerkezetünk átmásolása

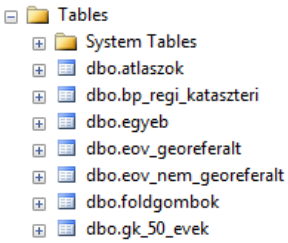
Eredményképpen azt láthatjuk, hogy az MS SQL szerveren is megjelenik a megfelelő táblaszerkezetünk, csak éppenséggel még semmilyen adat nincsen benne, sőt, hogyha átváltunk a Management Studio-ra, akkor láthatjuk, hogy még az üres táblák sem kerültek bele az adatbázisba.



14. ábra: A táblaszerkezet megjelenése a másik szerveren

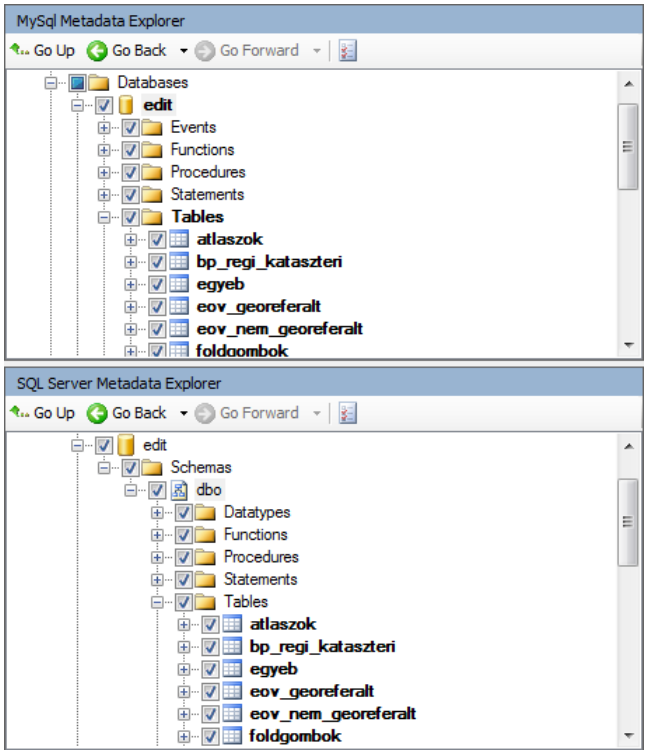
Ahhoz tehát, hogy a megfelelő táblaszerkezetünk „beleíródjon” az adatbázisba, összhangba kell hoznunk a változtatásokat az adatbázissal. Ezt megtehetjük az adatbázisunk nevére jobb kattintással előugró menü ***Synchronize with Database…*** parancsával. Ha ezután megvizsgáljuk az adatbázisunkat a

Management Studio-ban, akkor láthatjuk, hogy most már létrejöttek a megfelelő táblák.



15. ábra: A létrejött táblák

Az adatok valós átviteléhez még az szükséges, hogy mindkét ***Metadata Explorer*** ablakban ki legyen jelölve az egész adatbázis, vagyis az adatbázis neve előtti jelölőnégyzet mindkét esetben ki legyen pipálva az alábbi képen látható módon.

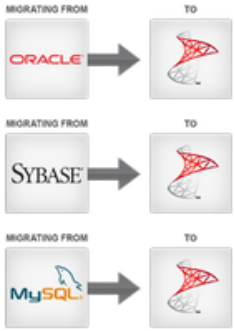


16. ábra: Mindkét adatszolgáltatón az adatbázis kijelölése

Ezek után már csak az adatok konkrét átvitele történik, melyet a ***Migrate Data*** paranccsal hajthatunk végre.

Eredményképpen létrejön az adatbázisunk az MS SQL szerveren, mely pontosan azt a szerkezetet és felépítést mutatja, mint az eredeti MySQL adatbázisunk.

Megjegyzendő, hogy a Microsoft SQL Server termékcsalád folyamatosan törekszik arra, hogy különböző más adatbázis-kezelő rendszerekhez rendelkezésre álljon egy a fentiekhez nagyon hasonló migráció-konverziós eszköz. Ennek oka részint ezeknek a konkurenseknek az ellehetetlenítése, részint a felhasználók minél nagyobb számban való vonzása a termék irányába.



17. ábra: Más adatbázis-kezelő rendszerekhez kapcsolódó konverziós eszközök

IV. Webes alkalmazás készítése ASP.NET-ben

Az előbbi fejezetekben bemutatásra került, hogy milyen lehetőségeink vannak a digitális térképtárunkhoz kapcsolódó adatbázissal kapcsolatban. Azonban ha ezeket az adatokat meg is szeretnénk jeleníteni a világhálón, akkor ehhez célszerű egy webes alkalmazást kell készítenünk. Erre a Visual Studio 2010-ben beépítve található ASP.NET szerveroldali alkalmazás-készítő eszközt fogjuk használni, mely használja a .NET néven ismertté vált keretrendszert.

IV.1. Bevezetés az ASP.NET alapvető használatához

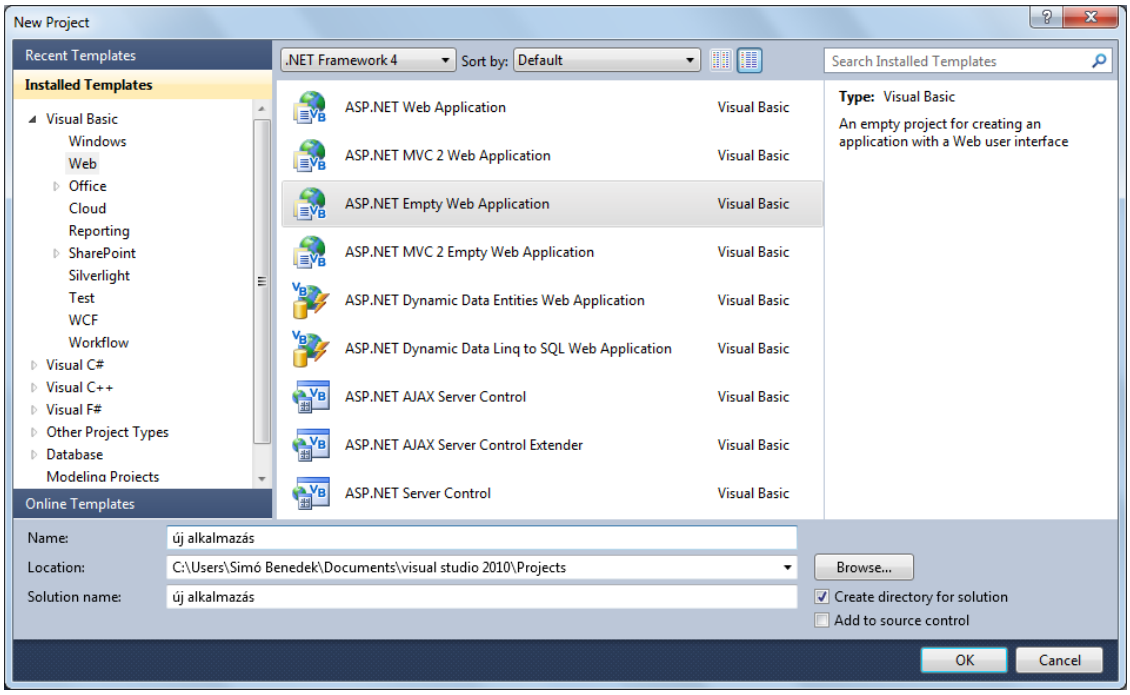
A .NET keretrendszer egy olyan szoftverfejlesztői platform, mely lehetővé teszi sokféle típusú alkalmazás, többek között desktop, kliensoldali, illetve szerveroldali alkalmazások fejlesztését. Ez a keretrendszer alapvetően az objektum-orientált programozási nyelveket támogatja, többek között beépített nyelvként jelenik meg benne a C#, a C++, illetve a Visual Basic. A keretrendszer lényege, hogy az egyes programozási nyelvek közös objektum-modellt és osztálykönyvtárat használnak. Az alkalmazás létrehozása során a fejlesztő kontrolokat ad hozzá a felhasználói felülethez, és ezeket szabadon testre szabja a háttérkódban (Code behind) az adott osztály eseménykezelőivel, tulajdonságaival és eljárásaival.

Az ASP.NET egy olyan a .NET keretrendszerbe beépülő eszköz, mely dinamikus szerveroldali alkalmazások készítését teszi lehetővé. (Erre az eszköz angol neve is utal: ASP = Active Server Pages.) Segítségével nemcsak egy felhasználói felületet tudunk megalkotni és hozzá arculatot készíteni, hanem megoldható a szerverünkön található adatok közlése is a felhasználó irányába.

IV.2. Az ASP.NET alapvető használata

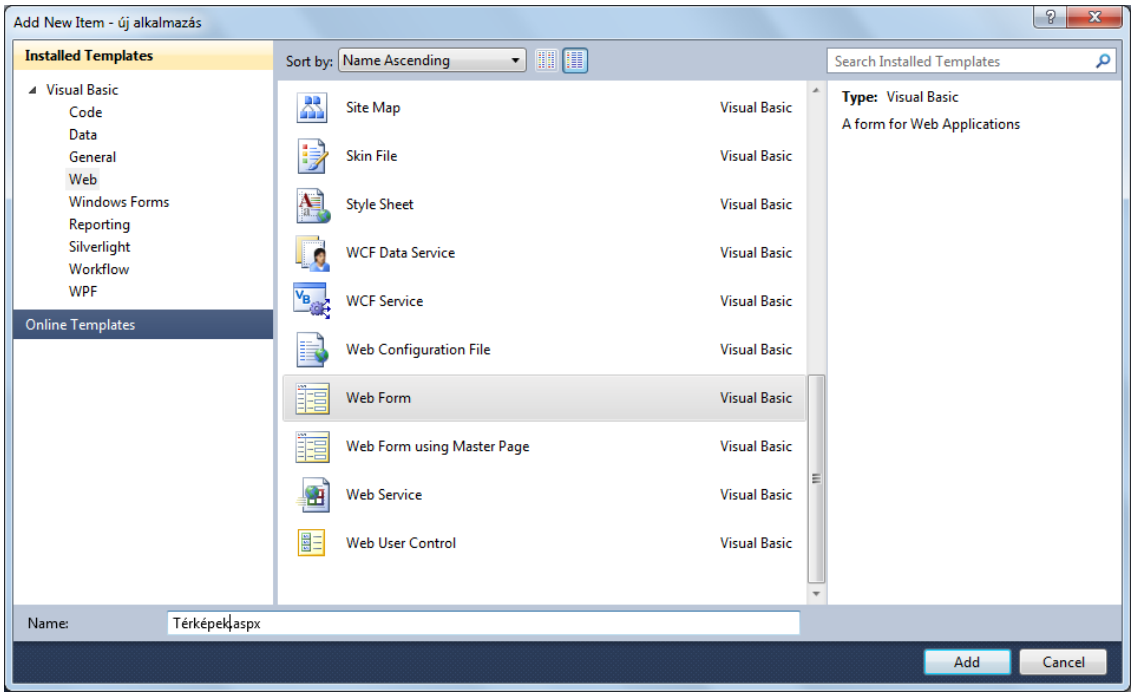
Első lépésként ismerkedjünk meg a felhasználói környezettel. Mivel az alkalmazásunkat ASP.NET-ben szeretnénk elkészíteni, indítsuk el a Visual Studio 2010-es verzióját, és hozzunk létre egy új projektet, mely egy ASP.NET üres

webalkalmazás lesz. Ehhez válasszuk a ***File*** menü ***New Project…*** lehetőségét, és a kész minták (***template***-ek) közül válasszuk a nekünk megfelelő típust.



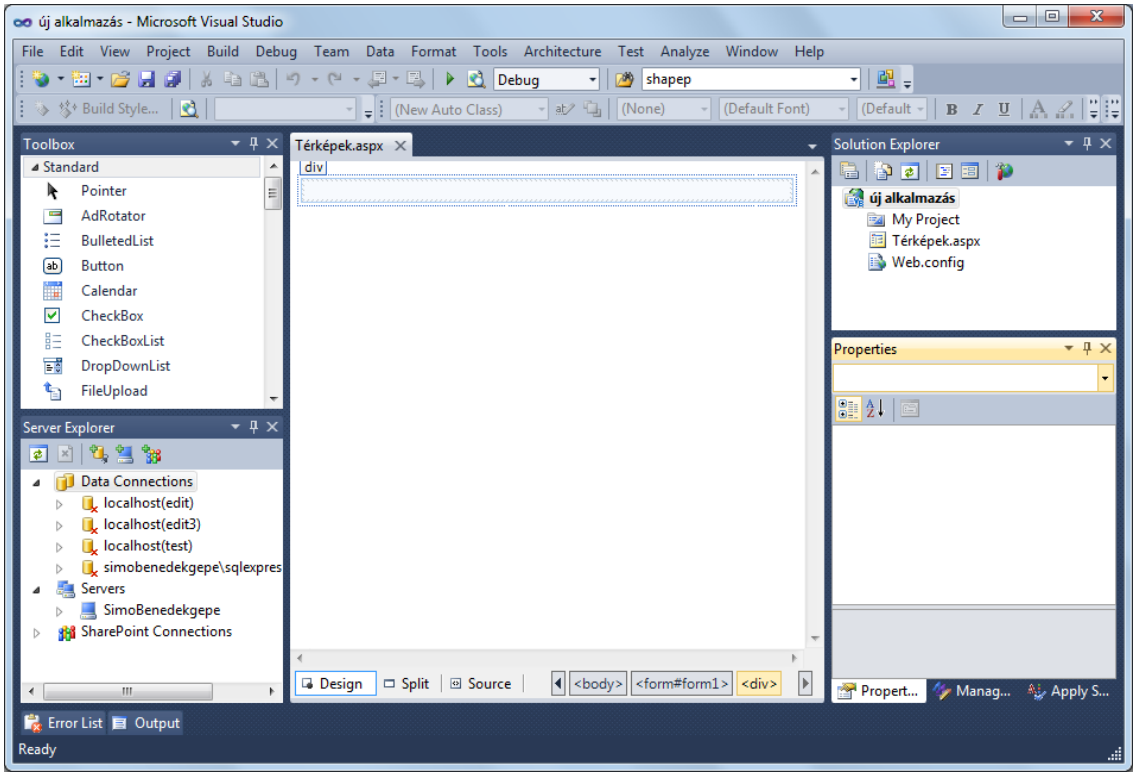
18. ábra: Új üres ASP.NET projekt létrehozása

Ha sikeresen létrehoztuk az új projektünket, akkor ismerkedjünk meg a felhasználói környezettel. Az egyik legfontosabb az általában a jobb oldalon elhelyezkedő ***Solution Explorer***, melyben a projektünkhöz tartozó összes állományt nyilvántarthatjuk, illetve kezelhetjük. Itt kattintsunk most a projektünk nevére, és a felugró menüből válasszuk az ***Add../New Item…*** lehetőséget. A felugró ablakban válasszuk a ***Web Form*** elemet, és adjunk neki egy tetszőleges nevet.



19. ábra: Új oldal hozzáadása a projektünkhöz

Ha az ***Add*** gombra kattintunk, megjelenik a most már sokkal teljesebb képet mutató fejlesztői környezetünk. Vizsgáljuk meg ezek közül a legfontosabbakat.



20. ábra: A fejlsztői környezet ASP.NET-ben

A fejlesztői környezet a szokásos menü és ikonsorok mellett három alapvető mezőből áll. Két oldalsó, ún. tabokat tartalmazó oldalsávból, valamint az ezek által közrefogott kódolási és dizájner felületből.

A tabokat tartalmazó oldalsávok tartalma és elrendezése szabadon testre szabható, az egyes tartalmi elemeket a ***View*** menüpont alatt tudjuk kiválasztani és a megjelenítésre kijelölni. Nézzük ezek közül a legfontosabbakat.

A ***Toolbox*** tabunk az elérhető kontrolok listáját tartalmazza; ezeket tetszőlegesen hozzáadhatjuk a projektünkhöz (drag-and-drop = megfog és elenged módszerrel).

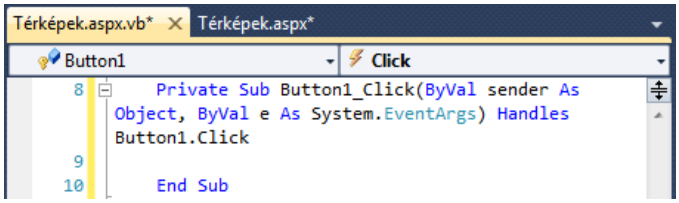
A következő a már korábban említett ***Solution Explorer***, melyben a projektünkhöz tartozó összes elemet kezelhetjük, illetve itt adhatunk a projektünkhöz újabb elemeket.

Fontos elem még a ***Properties*** ablak, melyben az adott kiválasztott elemhez tartozó attribútumokat tekinthetjük meg, illetve lehetőségünkben áll ezeken módosításokat is végrehajtani.

Az adatkezeléshez és az adatbázisokkal való kapcsolat nélkülözhetetlen eleme a ***Server Explorer*** tab, melyben az egyes adatszolgáltatók irányában rendelkezésre álló kapcsolatok állapotait tekinthetjük meg. Ez a későbbi adatkezelés során kulcsfontosságú szerepet fog majd betölteni.

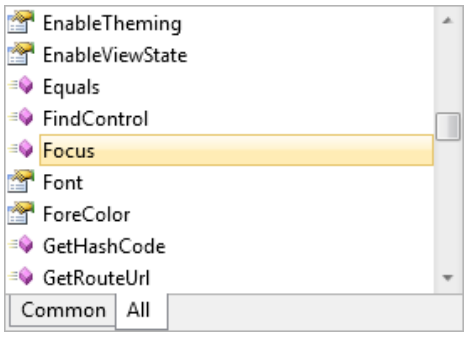
Legnagyobb jelentőséggel a középső oldalsáv bír, az összes kódolási és dizájn tevékenység ebben az ablakban történik. Legfontosabb azt tudnunk, hogy például egy alapvető Web Form elemhez háromféle nézet létezik ebben a középső ablakban. Az első, talán legegyszerűbb forma az oldalunk dizájn nézete. Ekkor gyakorlatilag a majdani működés során megjelenő felülethez egy nagyon hasonló nézetet láthatunk. A következő lehetőséget úgy érjük el, hogy az oldalunk alján a ***Source*** gombra klikkelünk, ekkor az oldalunk kódolt hátterét nézzük (ez nem egyezik meg a háttérkóddal!), mely gyakorlatilag az a kód lesz, melyet az alkalmazásunk küldeni fog majd a szervernek (ezt ***markup***-nak is nevezzük). A harmadik lehetséges nézet, amikor az alkalmazásunkat objektum-orientált formában szeretnénk kódolni. Ezt úgy érhetjük el, hogy a ***Solution Explorerben*** az .aspx oldalunk nevére kattintunk és a ***View Code*** lehetőséget választjuk. Ekkor megjelenik egy fejlesztői felület, mely az általunk preferált programozási nyelven teszi lehetővé az egyes kontrolok működésének a szabályozását.

Az alkalmazásunk, illetve az egyes elemek működésének a meghatározásához eseménykezelőket használunk. Minden objektum-osztálynak vannak megfelelő eseménykezelői, melyek az objektumon bekövetkezett eseményekre reagálnak. Egy ***Button*** osztályba tartozó elem alapvető eseménye például a ***Click*** (ráklikkelés) esemény. Ha ez az esemény megtörténik, akkor a hozzá társított eseménykezelő lép életbe.



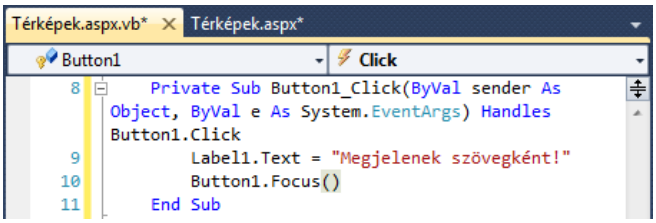
21. ábra: Az egyes objektumokhoz tartozó eseménykezelők

Az egyes objektumok testreszabásához és értékadásaihoz az egyes objektumok eljárásait és tulajdonságait használhatjuk. Az alábbi ábrán a projektünkhöz hozzáadott ***Button1*** azonosítóval rendelkező elem néhány ilyen eljárását és tulajdonságát láthatjuk.



22. ábra: Egy tetszőleges kontrol eljárásai és tulajdonságai

Nézzünk akkor egy alapvető példát. Dizájn felületben drag-and-drop eljárással adjunk a webalkalmazásunkhoz két egyszerű kontrolt, egy ***Buttont*** és egy ***Labelt***. A ***Button1 Click*** eseményének bekövetkeztekor a ***Labelünk*** text tulajdonságát tegyük egyenlővé egy általunk meghatározott szöveggel, majd a ***Button1*** elem eljárásai közül válasszuk a ***Focus***-t, mely kiemeli a gombunkat a felhasználói felületből.



23. ábra: Egy egyszerű eseménykezelés

A fent leírtak remélhetőleg már egy jó alapot teremtenek ahhoz, hogy a később közölt részek érthetőbbé váljanak.

IV.3. Az adatkezelés megvalósítása ASP.NET-ben

A fejlesztői környezet alapvető fontosságú elemeinek megismerése után vizsgáljuk meg, hogy milyen adatkezelési módszereket használhatunk a .NET keretrendszeren belül.

Az adatok kezelése .NET keretrendszerben egészen a a 3.5-ös verzió megjelenéséig szinte kizárólagosan az ADO.NET 2.0-ás adatkezelési módszeren alapult. Ennek lényege, hogy az adatkezelés első lépésként kapcsolatot építünk, majd egy parancs objektumot hozunk létre, mely a lekérdezéseinket közvetíteni fogja az adatbázis irányába. Ez a parancs objektum közvetít az adatbázisunk irányába, és visszaadja az általunk megfogalmazott lekérdezésnek megfelelő adathalmazt. A kapott eredményt tárolhatjuk egy DataSet objektumban, melyből aztán még az adatokat is kell olvasnunk.

Ez a módszer egy nagyon hatékony eszköz tud lenni az adatkezelésben, de a vele kapcsolatban felmerülő legnagyobb problémát az jelentheti, hogy kevés köze van az objektum-orientált programozáshoz, és pont ebből fakadóan nem érhető el hozzá intellisense, vagyis automatikus kód-kiegészítési opció. Ez csupán egy kényelmi szempontból hátrányos tulajdonság, de további felmerülő probléma lehet az, hogy mivel az adatkezelés minden lépcsőjét mi járjuk végig, jelentősen megnövekszik az emberi figyelmetlenségből adódó hibázás lehetősége.

IV.4. A LINQ to SQL elvi működési vázlata

Az előbbi fejezetben vázolt viszonylag bonyolultabb adatkezelési módszer, az ADO.NET mellé fejlesztett ki a Microsoft egy új technológiát a .NET 3.5-ös

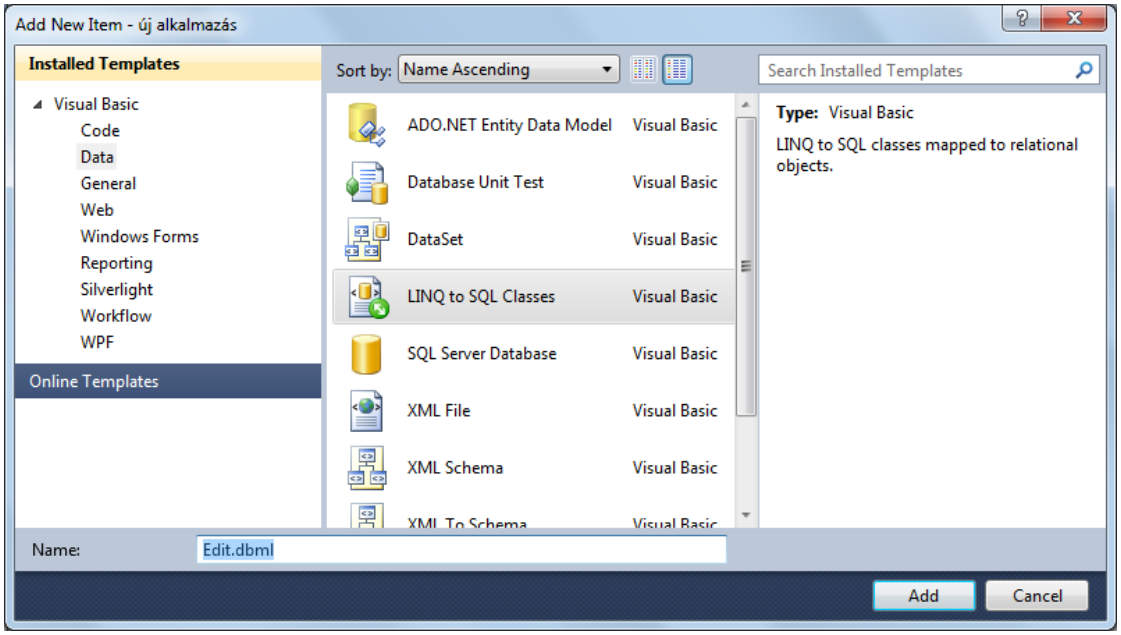
keretrendszerben, mely a LINQ to SQL nevet kapta (LINQ = Language Integrated Query = nyelvbe ágyazott lekérdezés).

A technológia lényege röviden, hogy a relációs adatbázisban található adatokat egy speciális objektum segítségével objektum-orientált formába hozza. (Ezt a speciális objektumot ***DataContext***-nek nevezzük.) Ez azt jelenti, hogy az adatbázisban található adott tábláknak megfelelően létrejönnek osztályok, míg az adott adatbázis tábla oszlopainak megfelelően létrejönnek az ezekhez az osztályokhoz kapcsolódó tulajdonságok. Például ha van egy térképek táblánk, melynek oszlopai az egyes térképek jellemzőit tartalmazzák, akkor a fejlesztői környezetben létre fog jönni egy olyan térkép osztályom, melynek a tulajdonságai között szerepelni fognak az egyes térképi jellemzők. (Megjegyezném, hogy ez a módszer csak MS SQL szolgáltatóval működik.)

IV.5. A LINQ to SQL gyakorlati működése

Lássuk akkor, hogy miként működik ez az eszköz a gyakorlatban, és hogy milyen lehetőségeket biztosít a fejlesztőnek.

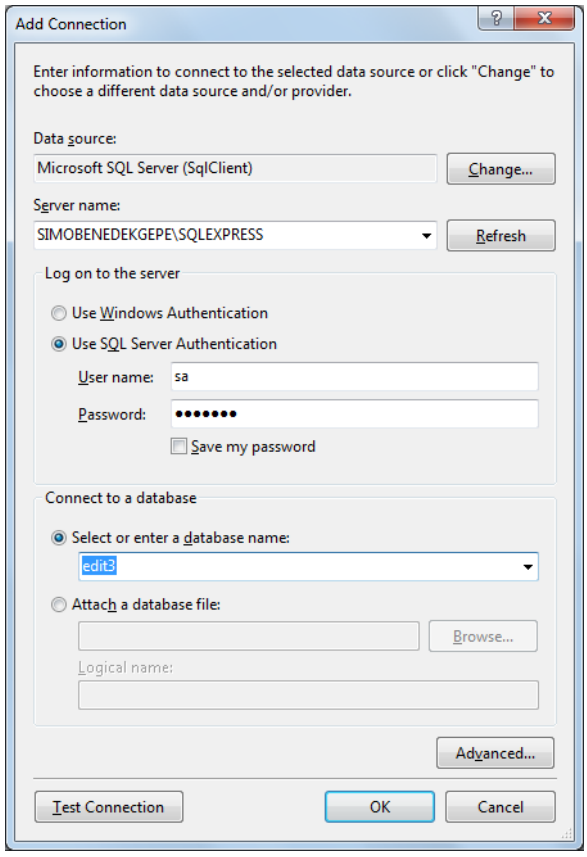
Első lépésként adjunk a projektünkhöz egy új elemet, melynek a típusa LINQ to SQL Classes, a nevét pedig tetszőlegesen adjuk meg.



24. ábra: A megfelelő .dbml fájl létrehozása

Ez egy .dbml kiterjesztésű fájlt fog hozzáadni a projektünkhöz. Ha létrejött a megfelelő fájlunk, akkor hozzá kell adnunk a megfelelő adatbázisunk kívánt tábláit.

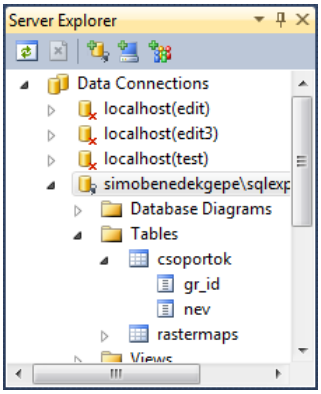
Ehhez kattintsunk A ***Server Explorer*** fülön belül a ***Data Connections*** elemre az egér jobb gombjával, majd a felugró menü lehetőségei közül válasszuk a ***New Connection*** lehetőséget. Itt adjuk meg az adatbázis-kiszolgáló típusát, majd az újabb felugró ablakban adjuk meg a kívánt szerver nevét, adjuk meg a bejelentkezéshez szükséges adatokat, valamint a kívánt adatbázisunk nevét.



25. ábra: Új kapcsolat létrehozása

Amennyiben biztosak szeretnénk lenni abban, hogy a kapcsolatunk létrejön és valóban működik, a ***Test Connection*** gombra kattintva ezt ellenőrizhetjük.

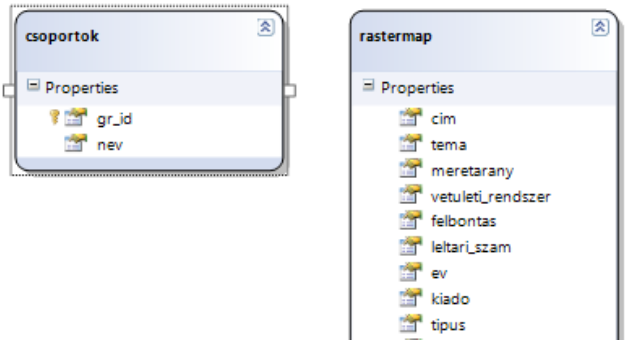
Ha sikeres volt a tesztelés, akkor OK-t nyomva a ***Server Explorer*** ablak ***Data Connections*** menüjében elérhető a kívánt adatbázis.



26. ábra: Az adatbázisunk megjelenése a Server Explorerben

Folytatva a .dbml fájlunk elkészítését, jelöljük ki az adatbázisunkból az alkalmazásunk fejlesztése közben használni kívánt táblákat, és ezeket húzzuk a .dbml fájlunk grafikus felületére.

Ezzel a grafikus felületen létrejönnek az egyes táblák képei, mint osztályok, valamint az egyes oszlopok adatai, mint tulajdonságok.



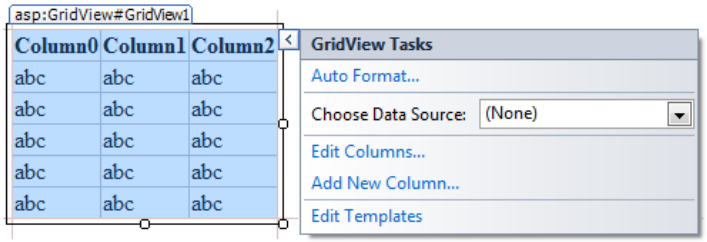
27. ábra: Az osztályok és tulajdonságok grafikus megjelenése

Ha elmentjük a .dbml fájlunkat, akkor a háttérben legenerálódik az adatbázisunk lemappelése, és létrejönnek az adott táblánknak megfelelő nevű gyűjtemények a ***DataContext*** objektumunkon keresztül. (Megjegyzendő, hogy ahhoz, hogy a ***DataContext*** objektumunkat használni is tudjuk, fontos ***Build***-et nyomnunk az alkalmazásunkra.)

Nézzük ezek után, hogy mi is a létrejött ***DataContext*** objektumunk szerepe, és hogy milyen formában hivatkozhatunk rá. Alapvetően kétféleképpen tudjuk használni. Az első módszer során mint adatforrásként tudunk hivatkozni rá, a következőkben ezt fogjuk majd áttekinteni. A másik módszer a háttérkódban történő használat, vagyis lekérdezések megfogalmazása, de ezt a módszert majd csak később tekintjük át.

IV.6. A GridView vezérlő LINQ adatforrással való haszálata

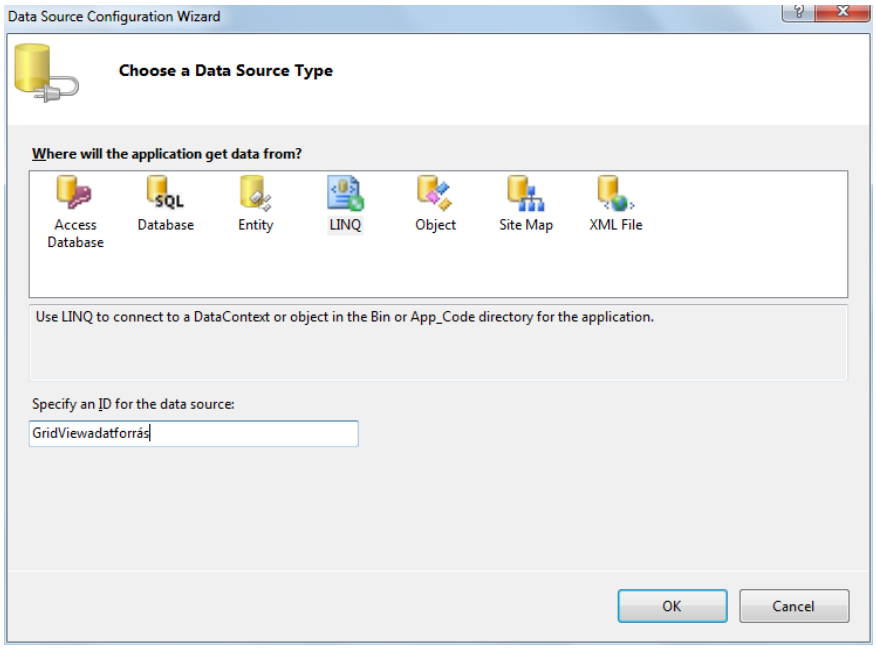
Az adatbázisunk megjelenítéséhez többféle vezérlő is rendelkezésre áll, ilyen például többek között a Listview vagy a Gridview elem. Ezeket az ún. ***smart-tag***-eken keresztül szabhatjuk testre; ezeken keresztül állíthatjuk be a kinézetet, az egyes oszlopok tulajdonságai, valamint a vezérlőnk adatforrását is.



28. ábra: A GridView vezérlő testreszabása

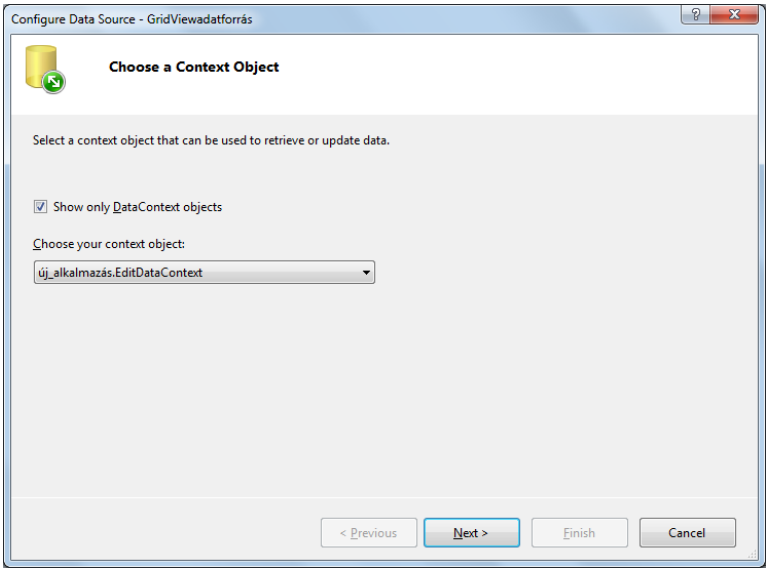
Válasszunk tehát új adatforrást, kattintsunk a ***Choose Data Source*** felirat melletti legördülő listára, és válasszuk a ***New Data Source…*** lehetőséget.

A felugró ablakban többféle adatforrás típus közül választhatunk. A felsorolt lehetséges adatforrások közül mi most a LINQ-t szeretnénk használni, tehát válasszuk ezt a lehetőséget, adjunk neki egy elnevezést, és alul válasszuk az OK lehetőséget.



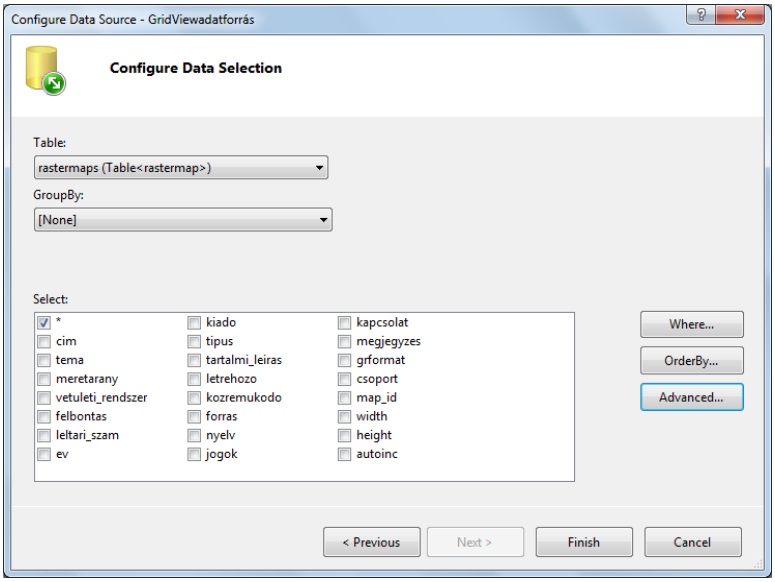
29. ábra: Az adatforrás kijelölése

Ezt követően az újabb felugró ablakban megjelenik a projektünkhöz hozzáadott valamennyi ***DataContext*** objektum, esetünkben az az egy, amit hozzáadtunk Edit néven.



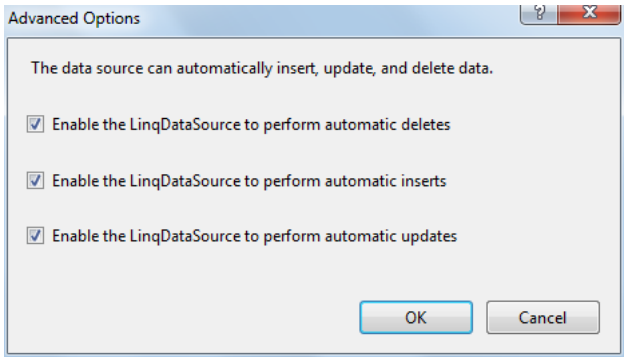
30. ábra: A Data Context objektum kiválasztása

Miután kiválasztottuk a ***DataContext*** objektumunkat, a következő lépésben konfigurálhatjuk az adatforrásunkat, ami több lehetőséget is jelenthet. Egyfelől kiválaszthatjuk, hogy melyik táblánk adatait jelenítse meg a vezérlőnk, illetve azt, hogy ezeket az adatokat alapértelmezés szerint milyen oszlop adatai szerint csoportosítsa. Ezen kívül meghatározhatunk ***Where*** és ***OrderBy*** feltételeket, valamint speciális lehetőségeket.



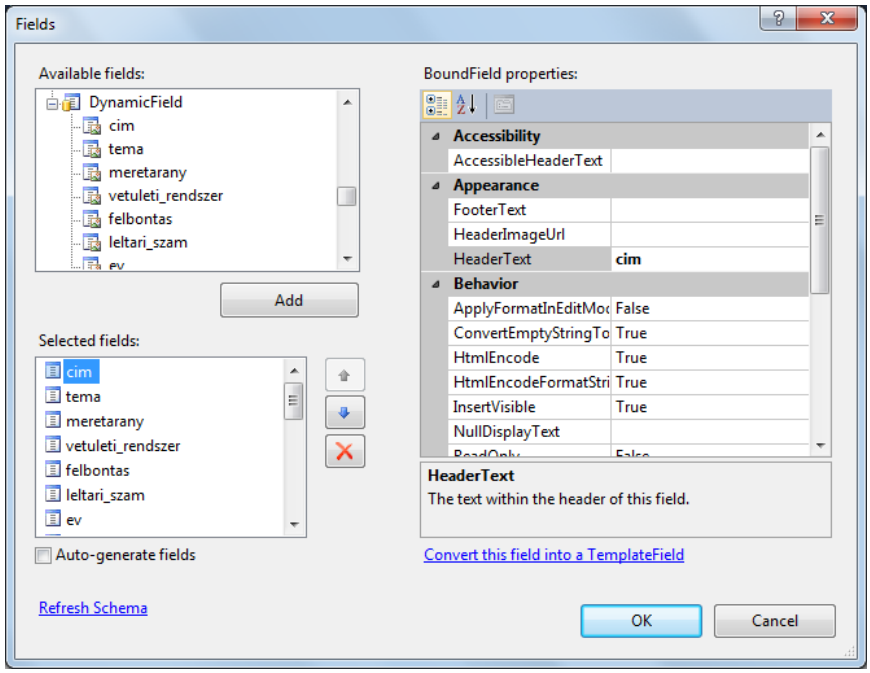
31. ábra: Az adatforrásunk konfigurálása

Erre az utóbbi ***Advanced*** lehetőséggel beállíthatjuk, hogy a ***GridView*** vezérlőnk alapértelmezésben támogassa-e a beszúrást, a felülírást, illetve a törlést.



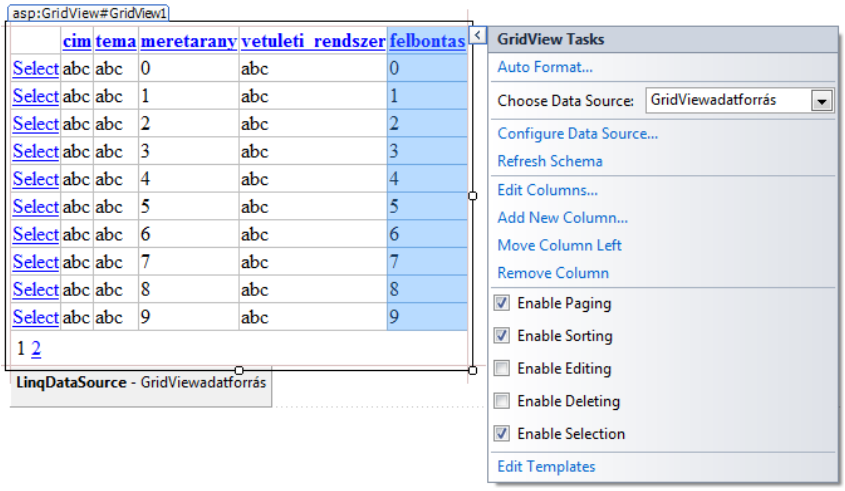
32. ábra: Speciális lehetőségek a konfiguráláshoz

Amennyiben minden kívánt beállítást elvégeztünk, a befejezés gombra kattintva dizájn nézetben átalakul a táblázatos megjelenítésünk; megjelennek az egyes oszlopok nevei a vezérlőnkben. Amennyiben van olyan adat vagy oszlop amelyet most nem szeretnénk megjeleníteni, akkor a vezérlőnk ***smart-tag***-jén belül válasszuk az ***Edit Columns*** lehetőséget, és itt szabadon testreszabhatjuk az egyes oszlopok szinte minden tulajdonságát, illetve azt, hogy egyáltalán megjelenjenek-e a táblázatunkban.



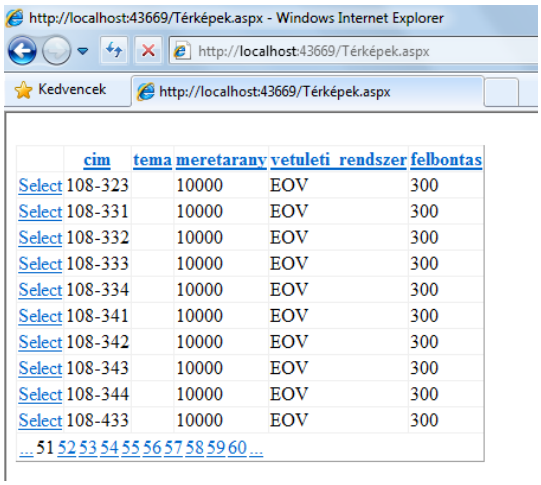
33. ábra: Az oszlopaink testreszabása

A megjeleníteni kívánt oszlopok kiválasztása után vizsgáljuk meg a vezérlőnk ***smart-tag***-jét, és vegyük észre, hogy új elemekkel gazdagodott; megjelentek beépülő lehetőségek jelölőnégyzetek formájában. Ezek közül kiválaszthatjuk azokat, amelyekre várhatóan szükség lesz.



34. ábra: A vezérlő testreszabása

Ha elkezdjük futtatni az alkalmazásunkat, akkor láthatjuk, hogy a felhasználói felületen megjelenik a meghatározott táblánk, de természetesen csak az általunk kiválasztott oszlopok adataival.



35. ábra: A felhasználói felület

Az adatok böngészését segíti az oszlop szerinti rendezés és lapozás, valamint lehetőség van egy adott rekord kiválasztására is. Ezt a funkcionalitást természetesen később bővíthetjük is; az imént való bemutatás csak azt a célt

szolgálta, hogy megmutassuk, hogy néhány egyszerű lépésből hogyan lehet egy működő felületet létrehozni.

IV.7. A GridView vezérlő szűrési lehetőségei

Gyakran előfordul, hogy egy adatbázis valamelyik mezője alapján szeretnénk egy vagy több rekordot megkeresni az adathalmazunkból. Lényegében azt szeretnénk elérni, hogy az oldalunk egy keresőfunkcióval bővüljön, tehát egy keresőmezőbe beírt érték alapján „megszűrjük", illetve kiválogassuk a megjelenítendő adatokat. Ennek a megvalósításához a megjelenítéshez használt vezérlő (esetünkben a ***GridView***) létező adatforrására kell egy olyan lekérdezést megfogalmaznunk, melyben megjelenik egy ***Where*** feltétel az adott mezőre vonatkozólag.

Erre a szűrésre a sokrétű lehetőséget biztosító ***Query Extender*** eszközt használhatjuk. Ennek egyetlen hátránya, hogy nem rendelkezik semmilyen grafikus kezelési lehetőséggel, tehát a ***markup***-ba kell némi kódmennyiséget elhelyeznünk, de ez csak néhány sor kódolást jelent. Nézzük akkor ennek az eszköznek a használatát.

Adjunk hozzá a felületünkhöz a ***Toolbox*** fül ***Data*** csoportjából egy ***Query Extender*** kontrolt, majd váltsunk át forráskód (***source***) nézetre, és keressük meg a hozzáadott elemhez kapcsolódó kódot. Ebben a kódban adhatjuk meg a szűrésünk paramétereit.

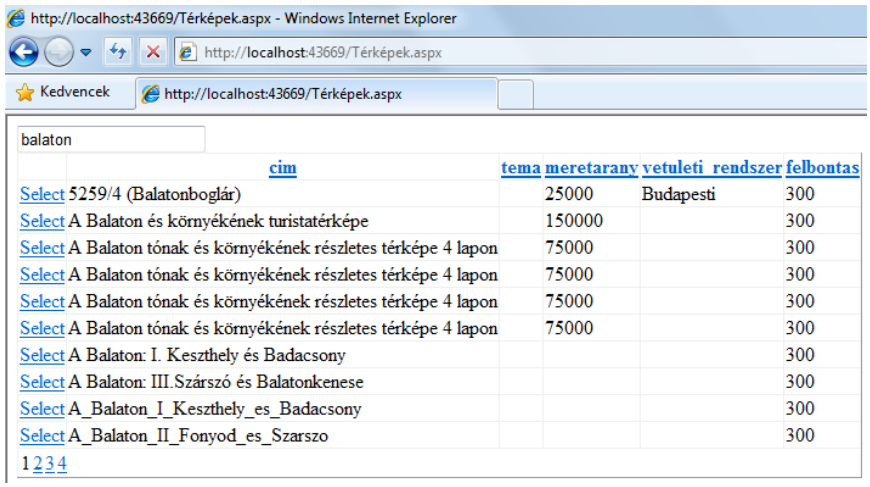
```
28        <asp:QueryExtender ID="QueryExtender1" runat="server" TargetControlID="GridViewadatforrás">
29            <asp:SearchExpression DataFields="cim" SearchType="Contains">
30                <asp:ControlParameter ControlID="TextBox1" />
31            </asp:SearchExpression>
32        </asp:QueryExtender>
```

36. ábra: a Query Extender eszköz konfigurálása

A ***TargetControlID*** értékhez annak az adatforrásnak az azonosítóját kell megadnunk, amelyikre a szűrést el szeretnénk végezni. A ***Search Expression*** tag-ben meg kell adnunk, hogy melyik mező alapján szeretnénk a keresésünket megvalósítani, illetve azt, hogy a keresésünk milyen típusú legyen. A ***StartsWith*** lehetőség az összes olyan rekordot visszaadja, melynek az adott mezőjében ezzel a megadott karaktersorozattal kezdődik az érték. A ***Contains*** típus az előbbihez nagyon hasonló, és eredményként visszaadja az összes olyan rekordot,

melynek az adott mezőben felvett értéke tartalmazza a megadott karaktersorozatot. A tag-en belül egy másik, ***ControlParameter*** tag-et is meg kell adnunk; ebben határozzuk meg, hogy mi lesz az a bizonyos kontrol, melynek az értéke a keresés alapja lesz. (Esetünkben ez egy egyszerű ***TextBox*** kontrol lesz.)

Ha most futtatjuk az alkalmazásunkat, akkor láthatjuk, hogy a keresőfeltétel megadása után a ***GridView*** vezérlőnkben már csak a szűkített adathalmazunk jelenik meg.



37. ábra: A keresőmotor működés közben

A módszer előnye, hogy a keresést egy olyan eljárás segítségével oldottuk meg, melyben nem kellett semmilyen T-SQL-ből ismert hagyományos lekérdezést megfogalmaznunk, és lehetővé vált egy olyan keresés, mely jóval túlmutat a hagyományos lekérdezési eljárások során használt szűrési lehetőségeken.

V. A Mapserver eszköz

Az előző fejezetben áttekintettük, hogy milyen módon tudjuk a térképek adatait tartalmazó adatbázisunkat „kivezetni” a felhasználói felületre. Következő lépésként azt szeretnénk elérni, hogy az adatbázisunk kiválasztott tetszőleges elemét képi formában tudjuk megjeleníteni a felhasználói felületen.

Mivel a megjelenítendő térképek esetünkben raszteres formában állnak rendelkezésre, ennek a megvalósításához akár használhatnánk egy egyszerű ***Image*** kontrolt is. Mivel azonban ezek az állományok térbeli adatokat tartalmaznak, ezért a megjelenítéshez célszerű egy olyan eszközt használnunk, mely tud mindenféle térképi jellegzetességet kezelni. Ilyen internetes publikálást tesz lehetővé többek között a Mapserver, melynek óriási előnye a versenytársaival szemben, hogy nyílt forráskódú, és ingyenes.

V.1. A Mapserver eszköz rövid áttekintése

A Mapserver eszközt a Minnesota Egyetem dolgozta ki, kezdetben azzal a céllal, hogy a NASA űrfelvételeinek egy publikációs felületet teremtsen. Később a felhasználás területe fokozatosan kiterjedt, és népszerűsége még ma is folyamatosan növekvő tendenciát mutat.

Működésének a lényege, hogy a működést biztosító alkalmazás (mapserv.exe) fut a webszerverünkön, és fogadja az általunk megfogalmazott lekérdezéseket, majd eredményként visszaküldi a lekérdezésnek megfelelő eredményt, ami egy mondjuk .png kiterjesztésű fájl formájában érkezik el hozzánk. Természetesen az egyes lekérdezések megfogalmazása általában nem a mi feladataink közé tartozik; ezt általában kliens oldali eszközökkel tudjuk megoldani, és így a felhasználónak csak a lekérdezés alapját jelentő fájl helyére kell hivatkoznia a böngésző keresőmezőjében.

V.2. A Mapserver meghatározó eleme, a mapfájl

A webes formában megjeleníteni kívánt raszteres vagy vektoros térképi állományunkról valamilyen információt kell nyerjen a Mapserver program, melyre alapozva aztán a megjelenítést meg tudja oldani. Erre a célra szolgál a .map kiterjesztésű fájl, melyről a fejlesztőnek kell gondoskodnia. Ez egy egyszerű szövegszerkesztőben is létrehozható és megnyitható állomány, melynek felépítése leginkább az XML kódolásra hasonlít, és viszonylag egyszerű szintaktikával rendelkezik.

Ebben az állományban határozzuk meg a megjeleníteni kívánt valamennyi térképi rétegünk minden leíró adatát. A mapfájlt szerkezete alapján osztályokra bonthatjuk, és minden osztálynak megvannak a leíró paraméterei.

A következőkben tekintsük át, hogy milyen képet mutat egy mapfájl, és hogy milyen nélkülözhetetlen elemeket kell tartalmaznia. Mivel esetünkben nem georeferált raszteres állományokkal dolgozunk, ezért főként az ilyen típusú állományok mapfájljának leíró adataival fogunk foglalkozni.

```
MAP
  NAME "1"
  STATUS ON
  SIZE 589 596
  EXTENT 0 0 4665 4724
  UNITS DD
  IMAGECOLOR 255 255 255

  OUTPUTFORMAT
    NAME 'AGG'
    DRIVER AGG/PNG
    IMAGEMODE RGB
  END

  WEB
        IMAGEPATH "C:\OSGeo4W\apache\htdocs\umn\tmp\"
        IMAGEURL "/umn/tmp/"
        TEMPLATE templ_mymap.html
  END

  LAYER
    NAME nevtelen1
    TYPE RASTER
    STATUS DEFAULT
        DATA "1.jpg"
  END

END
```

38. ábra: Egy egyszerű mapfájl felépítése

A mapfájlunk felépítése során objektumokat határozunk meg, melyekben előfordulhatnak leíró paraméterek vagy más objektumok. Minden objektum leírását az objektum nevével kezdjük és egy END taggal zárjuk le. Látható, hogy a mapfájlunk befoglaló keretét a legalapvetőbb objektum, egy MAP tag alkotja,

melyben a megjelenítés legalapvetőbb paraméterei találhatóak. Nézzük, hogy ezek mit jelentenek, illetve hogy milyen értéket célszerű nekik adni.

Az első a NAME paraméter, mely célszerű, hogy megegyezzen a fájlunk nevével. A második a STATUS paraméter, mely meghatározza, hogy a térképünk megjelenjen–e vagy sem. Ezt alapesetben ON vagy DEFAULT értékre állítjuk.

A következő két paraméter a Mapserver által küldött kép pixelekben mért nagysága, valamint a kép földrajzi kiterjedésére vonatkozik. Koordináta nélküli raszteres állományok esetében a kettő között igen szoros összefüggés van. Mivel ezekben az esetekben a kép földrajzi koordinátáit, vagyis az EXTENT paraméter értékeit az eredeti kép magassági és szélességi pixelértékei adják, ezért célszerű a befoglaló keret nagyságát is ehhez igazítani, és ezeket az előbbiek arányosan kicsinyített értékeként megadni. Ha például 2000 x 1600 pixeles az eredeti képünk akkor mondjuk célszerű mindekét értéket egy azonos értékkel elosztani. Így létrejöhet például 1000 x 800-as, vagy 500 x 400-as befoglaló keret, melynek értékét a SIZE paraméterben rögzíthetjük. Mindkét paraméter megadása során az értékek megadásának a sorrendje: szélesség majd magasság.

A következő érték a UNITS, melyben meghatározhatjuk, hogy fokokban, vagy kilométerekben szeretnénk-e megadni az fenti koordináta értékeket. Mivel ebben az esetben nem ismerjük a valós koordinátákat, ezért ennek esetünkben nincs jelentősége, de többek között felvehet DD=fokok, illetve kilometers és meters értékeket is.

A mapfájl IMAGECOLOR paramétere a befoglaló-keret háttérszínét határozhatja meg az additív színkeverésnek megfelelő red = vörös, green = zöld, illetve blue = kék színek 0-255-ig terjedő értékeinek meghatározásával.

A következő objektum, mely a mapfájlon, illetve a MAP tag-en belül megjelenik, az OUTPUTFORMAT nevet viseli. Ebben meghatározhatjuk, hogy milyen formátumú raszteres képet állítson elő a Mapserver eszköz, valamint hogy a generált raszter színes legyen-e vagy sem.

Az ezt követő objektum a WEB nevet hordozza. Itt adhatjuk meg, hogy az ideiglenesen létrehozott raszteres képeket hol tárolja a rendszer, illetve hogy ***browse*** üzemmódban milyen html template állományt használjon a Mapserver. Az objektumnak tehát három fő paramétere van. Az első kettő az ideiglenes fájlok tárolására vonatkozik; az IMAGEPATH ennek a mappának a helyi gépen (szerveren) történő abszolút megadása, míg az IMAGEURL ennek a mappának az

interneten keresztül elérhető, szerverhez képesti útvonalát adja meg. A TEMPLATE paraméter a browse üzemmódban használt html temlate fájl mapfájlhoz képest való elérését és nevét tartalmazza.

Az itt utolsóként megjelenő LAYER objektum a mapfájlunk talán egyik legfontosabb paramétere. Ebben adhatjuk meg a megjeleníteni kívánt rétegünk valamennyi jellemzőjét. A NAME paraméter értelemszerűen a rétegünk nevét határozza meg, míg a TYPE paraméter az adott réteg típusát adhatja meg, meg RASTER, illetve VECTOR értékeket vehet fel. A STATUS érték ismét ON vagy DEFAULT értéket vehet fel; ez határozza meg, hogy megjelenik-e az adott rétegünk. Ennek főként abban az esetben van jelentősége, ha esetleg több rétegünk is van, melyeket meg szeretnénk jeleníteni. Végül a DATA paraméter kell meghatároznunk. Ebben meg kell adnunk a megjeleníteni kívánt állomány fájlnevét és kiterjesztését, valamint a mapfájlhoz képest való elhelyezkedését a könyvtárfában. Utóbbinak akkor lehet jelentősége, ha a mapfájlunk és az állományunk nem egy mappában találhatóak, de tömeges megjelenítés esetén célszerű az összes mapfájlunkat és raszteres állományunkat egy helyen, egy mappában tárolnunk.

A fentiekben meghatároztuk, hogy milyen alapvető vonásokat hordozhat egy mapfájl. Ezek után vizsgáljuk meg, hogy miként hivatkozhatunk erre az állományra, és hogy milyen módon válhat lehetővé az állományunk megjelenítése.

V.3. Megjelenítés a mapfájl segítségével

Az előző fejezetben áttekintettük, hogy milyen szabályok szerint kell kialakítanunk a mapfájlunkat, de azzal nem foglalkoztunk még, hogy a gyakorlatban ezt hogyan használhatjuk. (A következőkben azzal a feltételezéssel élünk, hogy a szerverünkön, illetve a fejlesztéshez használt számítógépen telepítve van egy webszerver (mondjuk Apache szerver), valamint a Mapserver eszköz, melyet a webszerverünk CGI (Common Gateway Interface) programon keresztül tudunk elérni. A telepítéshez praktikus egy olyan telepítő-csomagot használnunk, mely a működéshez szükséges minden elemet tartalmazza. Ilyen lehet például az OSGeo4W telepítő-csomag, vagy esetleg a MS4W csomag, melyeket szabadon elérhetünk a Mapserver honlapján keresztül.

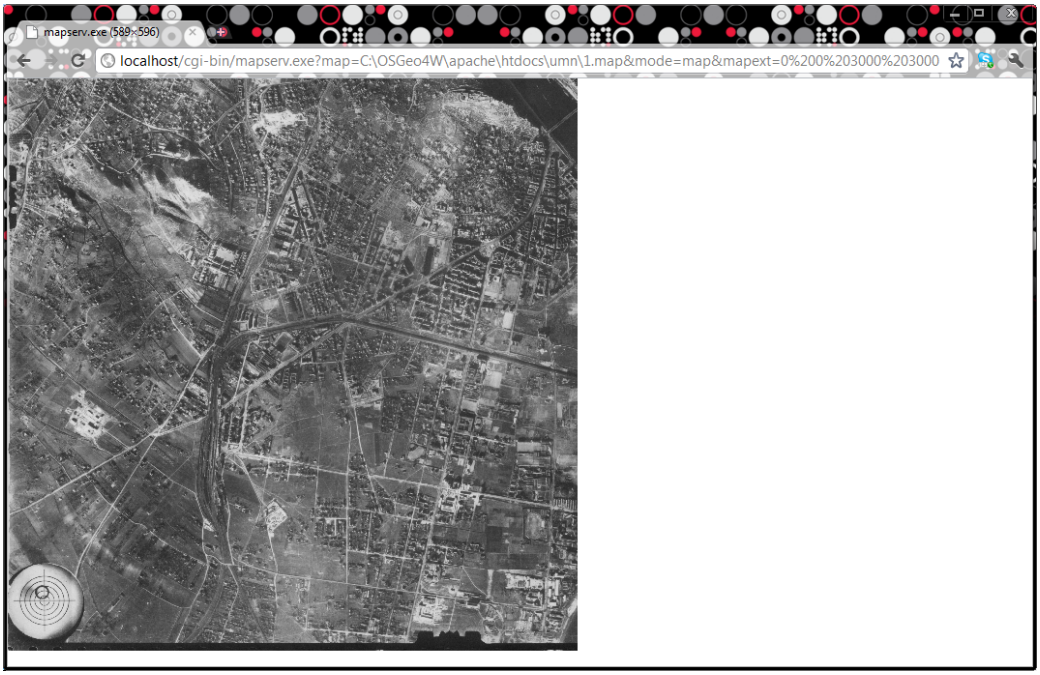
Ha sikerült megfelelően konfigurálni a rendszerünket, vizsgáljuk meg, hogy egy tetszőleges böngészőben hogyan érhetjük el az elkészült mapfájlunk által leírt állományt. (Az áttekintés során azt az esetet írjuk le, amikor helyi fejlesztői szerveren (localhost) dolgozunk.)

Első lépésben meg kell adnunk a mapserv.exe elérését a szerverünkön. Ez általában így néz ki: localhost/cgi-bin/mapserv.exe. Ezután egy kérdőjel következik, mely arra utal, hogy most jönnek azok a paraméterek, melyeket a webszerverünkön található cgi program majd lekérdezésként kezelni fog. Következő elemként a mapfájlunk abszolút elérési útját kell megadnunk a helyi merevlemezen. Ez a következőképpen nézhet ki: map=C:\OSGeo4W\apache\htdocs\umn\1.map. Végül az egyéb paramétereket adhatjuk meg egymástól & jelekkel elválasztva. Ilyen lehet a leggyakrabban használt mode=map paraméter megadása, de lehetőségünk van arra is, hogy a mapfájlban található extent értékektől eltérő paramétereket adjunk itt meg. Lássuk akkor, hogy milyen eredményt kapunk, ha ezeket a szövegrészleteket egymás után írva bemásoljuk egy böngésző keresőmezőjébe.

39. ábra: Megjelenítés a Mapserver eszköz segítségével

Ha minden paramétert helyesen adtunk meg, akkor eredményképpen megjelenik a megfelelő térképrészlet a böngészőnkben.

Felmerülhet azonban az igény, hogy a térképünknek csak egy bizonyos részét szeretnénk megjeleníteni. Ehhez közvetítenünk kell a Mapserver irányába azt a lekérdezést, mely leszűkíti a megjelenítendő területet az általunk kívánt kiterjedésre. Ahhoz hogy ezt elérjük, a lekérdezés paramétereihez adjuk hozzá a már ismert & karakter után a mapext= [bal alsó sarokhoz közelebbi x koordináta] [bal alsó sarokhoz közelebbi y koordináta][jobb felső sarokhoz közelebbi x koordináta][jobb felső sarokhoz közelebbi y koordináta] karaktersorozatot. Egy ilyen lehetséges kiterjedés meghatározása lehet például a mapext=0 0 3000 3000. Nézzük hogy ez milyen képet generál az előbbi Budapest szinte egészét mutató képből.



40. ábra: A térképünk megjelenítésének szűkítése

Látható, hogy a városnak csak a délkeleti része jelent meg, hiszen a maximális kiterjedés ezen kép esetében a 0 0 4665 4728 érték lenne.

V.4. A Mapserverrel generált kép beágyazása egy html oldalba

Az előbbi fejezetben bemutatásra került, hogy miként lehet legegyszerűbben egy térképes állományt a böngészőnkben megjeleníteni a Mapserver eszköz használatával, és hogy a megjelenítés hogyan korlátozható le egy adott területre.

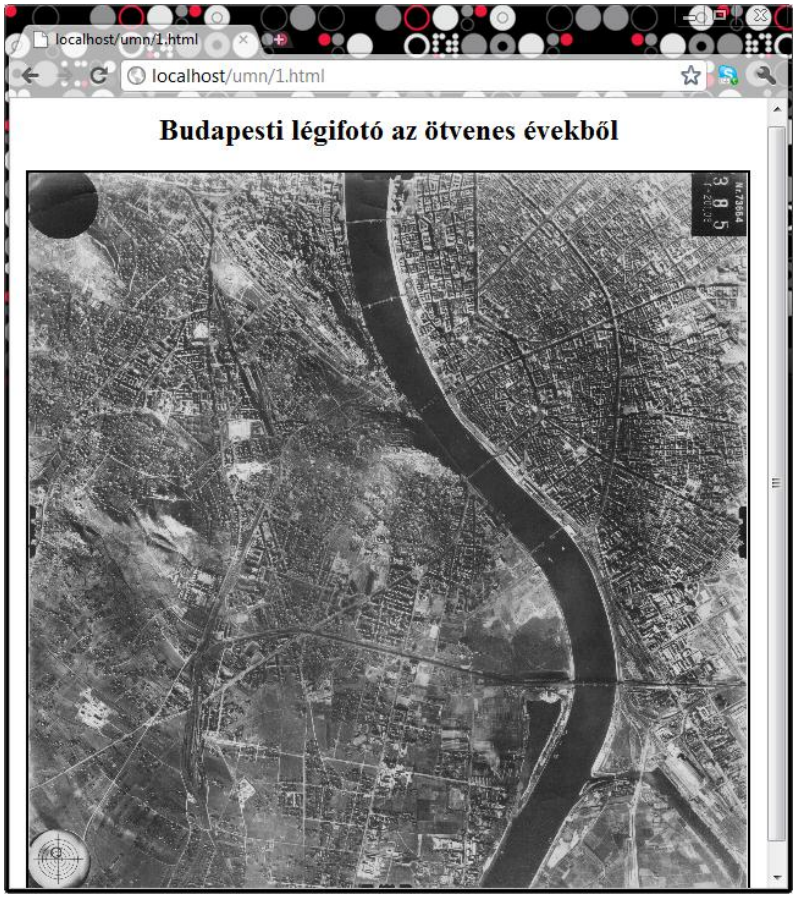
Láthattuk, hogy a Mapserver programnak a CGI-n keresztül adhatjuk át a kívánt nézet paramétereit. Annak érdekében, hogy ne kelljen minden egyes nézőkép-váltáshoz új adatokat beírnunk a böngészőnk keresőmezőjébe, kezeljük ezeket kliens oldalon, a html kódunkba beágyazott JavaScript kód segítségével. A html fájlunk body tag-jei közé adjunk hozzá egy egyszerű div elemet, melynek függőleges és vízszintes kiterjedését feleltessük meg a mapfájlunkban meghatározott SIZE értékekkel, és az azonosítójának válasszunk egy egyszerű és beszédes nevet, mondjuk legyen ez map_div.

A megjelenítést szabályozó JavaScript kódot a szokott gyakorlat szerint a head tag-ek közé írjuk. Több JavaScript könyvtárat is be kell linkelnünk az oldalunkhoz; ezek teszik majd lehetővé a térképünk egér-görgetővel történő nagyítási illetve kicsinyítési funkcióját, valamint a térkép vontatását az egérgomb lenyomva tartásakor. Azon függvény paramétereinek a megadása, melyet majd az oldalunk betöltésekor meghívunk, csak ezek után következik.

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript" src="../umn/js/jquery-1.4.4.min.js"></script>
<script type="text/javascript" src="../umn/js/jquery-ui.min.js"></script>
<script type="text/javascript" src="../umn/js/jquery.mousewheel.min.js"></script>
<script type="text/javascript" src="../umn/js/smm.js"></script>

<script>
    var probaMap;

    function init()
    {
        probaMap = new SamanMapserverMap(
                document.getElementById('map_div'),
        "http://localhost/cgi-bin/mapserv.exe?map=C:\\OSGeo4W\\apache\\htdocs\\umn\
\1.map&mode=map",
        0, 0, 4665, 4724);
    }

</script>
</head>

<body onload="init()">
<h2  style ='text-align:center '>Vereinigte Staaten von Amerika.I. Westligner
Teil</h2>
<div id="map_div" style='width:589px; height:596px; margin: 0px auto 0px auto;
border: 2px solid black;'></div>
</body>

</html>
```

41. ábra: A html kódunk

Ebben a script tag-ek között található JavaScript kódban elsőként deklarálunk egy ***probaMap*** változót. Ez után a függvényünkön belül ennek az új változónak értéket adunk; ez egy olyan típusú elem lesz, melyet az smm.js állományban deklaráltunk, és melynek neve ***SamanMapserverMap*** lesz. Ez három paraméter megadását kívánja meg tőlünk. Elsőként meg kell határoznunk annak a html elemnek az azonosítóját, melybe a lekérdezés meghívott eredményét tárolni szeretnénk; ez a korábban megadott map_div azonosítóval rendelkező div elem lesz. A második argumentum a korábban bemutatott egyszerű megjelenítéshez használt elérési útvonal, mellyel gyakorlatilag meghívjuk az adatunkat a szerverünkről. Harmadikként a kiterjedés (EXTENT érték a mapfájlunkban) paramétereit kell megadnunk, ezt szintén a mapfájlunkban határoztuk meg korábban. Mindhárom paraméter meghatározása után a html oldalunkat betöltve megjelenik a térképünk, melyet szabadon böngészhetünk az egér funkcionalitásának segítségével.



42. ábra: Térkép-megjelenítés html oldalon belül

Látható tehát, hogy a térképünkhöz, illetve a raszteres állományunkhoz kapcsolódó megfelelő paraméterek tudatában tetszőleges térképet, vagy egyéb térbeli vonatkozással rendelkező raszteres állományt meg tudunk jeleníteni egy html oldalon belül.

V.5. Automatizálási lehetőségek

Az előző fejezetekben áttekintettük, hogy miként kell egy raszteres állomány megjelenítéséhez elkészítenünk a mapfájlunkat, illetve a html kódunkat. Egy digitális térképtár esetében azonban a célunk az, hogy ezt a megjelenítést használjuk valamennyi raszteres állományunkhoz, mely az adatbázisunkban megjelenik. Nyilvánvaló, hogy a több ezer állomány esetében nem írhatjuk meg darabonként ezeket az állományokat, tehát szükségünk lesz valamilyen automatizálási módszerre.

Már első ránézésre is feltűnhet, hogy az egyes készítendő szöveges állományokban (a mapfájlban illetve a html kódban) nagyon sok az ismétlődő elem (melyek összessége szinte egy vázat alkot), és gyakorlatilag csak néhány olyan paraméter van, mely az egyes állományokat egyedivé teszi. Mivel ezeket a jellemző paramétereket az adatbázisunkban nagyszerűen tudjuk tárolni, ezért ésszerű lenne ezeket az adatokat az adatbázisunkból „hozzátenni” a meglévő vázunkhoz.

A Visual Basic .NET fejlesztői környezetben rendelkezésre áll az a lehetőség, hogy tetszőleges szöveges állományokat generáljunk. Mivel a LINQ to SQL segítségével az adatbázisunk egyes adataihoz is szabadon hozzáférhetünk az alkalmazások háttérkódjában, minden körülmény adott ahhoz, hogy ezeket a szöveges állományokat legeneráljuk.

V.6. A mapfájlok elkészítésének automatizálása

Vizsgáljuk meg elsőként, hogy melyek azok a paraméterek, melyek tömeges mapfájl generálás közben az egyes esetekben változhatnak, és hogy ezek az adatbázisból hogyan nyerhetőek ki.

Elsőként a NAME paramétert kell meghatároznunk, ez célszerű, ha megegyezik az adattáblánk elsődleges kulcsaként használt oszlop adott rekordra

vonatkozó aktuális értékével. Következőként a SIZE és EXTENT paramétereknek kell értékeket adnunk. Mivel azzal a feltételezéssel élünk, hogy az adatbázisunkban tárolva vannak az egyes raszteres állományaink magassági, illetve szélességi értékei pixelekben meghatározva, ezért az EXTENT értékhez 0 0 képszélesség, képmagasság értékeket kell beillesztenünk. A SIZE értékét célszerű az EXTENT paraméterből levezetni annak érdekében, hogy a befoglaló keretünk arányos legyen a kép oldalarányaival. Erre majd egy olyan iterációt fogunk használni, mely addig csökkenti a befoglaló keret méretét arányosan az EXTENT értékeken alapulva, míg az el nem éri egy általunk kitűzött méretbeli értéket. Változik még az adatforrásunk, vagyis a DATA paraméter értéke, mely meg fog egyezni az adattáblánk elsődleges kulcsának az aktuális rekordhoz kapcsolódó értékével, kiegészülve az ezután következő kiterjesztéssel, aminek az értéke ugyan változni fog, de ezt szintén az adatbázisból tudjuk majd lekérdezni.

A fenti leírást összegezve tehát mindenképpen tartalmaznia kell az adattáblánknak minden egyes rekordhoz a következő adatokat: az elsődleges kulcs értékét, a raszteres kép szélességét pixelekben, a raszteres kép magasságát pixelekben, illetve a kép kiterjesztését. Ha ezek az adatok rendelkezésre állnak, megkezdhetjük a mapfájlaink legenerálását, melyet majd egy asztali alkalmazással fogunk megvalósítani.

Hozzunk létre a Visual Basic .NET fejlesztői környezetben egy új Windows Forms alkalmazást, és elsőként nézzük, hogy miként tudunk lekérdezéseket végrehajtani a korábbi adatkezeléssel kapcsolatos fejezetekben már előkerült ***DataContext*** objektum segítségével a háttérkódban. Jelen esetben éljünk azzal a feltételezéssel, hogy már létrehoztuk a .dbml kiterjesztésű fájlunkat, melyben az adatbázisunk megfelelő tábláit lemappeltük.

Adjunk hozzá a felületünkhöz egy egyszerű ***Button*** kontrolt abból a célból, hogy a szöveges állományok legyártását tudjuk majd valamilyen eseményhez kötni, mely majd ennek a kontrolnak a klikk eseménye lesz.

Elsőként nézzük meg, hogy miképpen tudunk a ***DataContext*** objektumon keresztül egy adattáblánk tetszőleges rekordjának egy tetszőleges mezőjében felvett értékét megkapni. Ehhez elsőként deklarálnunk kell egy új ***DataContext*** objektumot az alábbi formában.

```
Dim dc As New EditDataContext
```

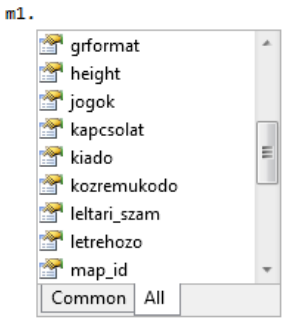
43. ábra: DataContext objektum deklarálása

Itt az ***Edit*** azonosító a .dbml fájlunk elnevezésére utal. Ez után következik a lekérdezés.

```
Dim m1 = (From m In dc.rastermaps _
          Where m.csoport = 7 _
          Select m).First
```

44. ábra: Egy lekérdezés a Data Context objektum segítségével

Egy új elembe „helyezzük” a lekérdezésünk eredményét, melynek jelen esetben az m1 elnevezést adjuk. Ezután következnek a lekérdezés lényegi elemei. Az egy adott adattáblára vonatkozó m elemhalmazból (mely jelen esetben a ***rastermaps*** táblánkra vonatkozik) kiválasztjuk az összes olyan elemet, melyre igaz az, hogy a csoport oszlopában a 7-es értéket veszi fel, és azok az elemek sokaságából, melyekre teljesül ez a feltétel, kiválasztjuk az elsőt. Így ezt az egy elemet fogja megtestesíteni az m1 objektum, és ennek megfelelően mint ennek az elemnek a tulajdonságai fognak megjelenni az adattáblánk egyes mezői.



45. ábra: Az adattáblánk mezői, mint tulajdonságok

Amennyiben tehát egy string-ben szeretnénk tárolni egy adott elem meghatározott tulajdonságát, akkor ezt az alább látható formában tudjuk megtenni.

```
Dim mag As String
mag = m1.height.ToString
```

46. ábra: Egy rekord mezőjének meghívása

Deklaráljuk tehát egy string típusú változót, majd adjunk értéket neki, mely az m1 elem egy kitüntetett tulajdonsága lesz. (A fenti esetben a mag azonosítóval rendelkező szöveges változóhoz a kiválasztott adattábla elemünk height = magasság értékét rendeltük hozzá.)

Ezzel tehát lehetővé válik egy elem bármilyen tulajdonságának a lekérdezése az adattáblánkból.

A fentiek ismeretében vizsgáljuk meg, hogy milyen lépéseket kell végrehajtsunk ahhoz, hogy a mapfájlunk változóit kinyerjük az adatbázisból, illetve hogy milyen módon tudjuk ezeket beleírni a készítendő szöveges állományba. Mivel ezt nem csak egyetlen elem esetében szeretnénk végrehajtani, hanem az adattáblánk összes elemére, ezért annak a lehetőségeit is számba vesszük, hogy milyen típusú iterációt célszerű alkalmaznunk. Természetesen ennek a megoldására több módszer is egyformán jó eredményt nyújthat, de ezek közül most csak egyet részletezünk.

Egyetlen elem esetében a következő lehet a megoldásunk. Valamilyen tetszőleges tulajdonsága alapján elsőként lekérdezzük ezt az elemet a Data Context objektumunkon keresztül. A fent közölt módszerrel string típusú változókba helyezzük a szükséges értékeket, majd a ***StreamWriter*** eszköz segítségével ezeket a mapfájlunk statikus elemeivel együtt beleírjuk egy szöveges fájlba.

Nézzük, hogy milyen lehetőségeink vannak, hogyha ezt a műveletet egy egész sokaság esetében szeretnénk végrehajtani. A művelet nagyon hasonló lesz, csak mindenképpen szükségünk lesz majd egy olyan elemre az adattáblánkban, mely az iterációhoz a továbblépést biztosítani fogja. Ehhez célszerű az adatbázisunkban mondjuk egy ***autoinc*** nevű mezőt létrehoznunk, mely az adattáblában lefele haladva egységnyi értékkel növekszik minden egyes rekord után. Az alább közölt ciklusmag többszöri végrehajtásáról esetünkben egy ***For-to*** ciklus k állandója fog gondoskodni, mely a k = 1 kezdőértéktől az összes elem számáig biztosítja a ciklusmagunk végrehajtását.

Vizsgáljuk meg ezek után, hogy ciklusunk magja hogyan alakul. Ezt két részre oszthatjuk. Az első részben az egyes elemek adatainak lekérdezése történik. Itt kérdezzük le a NAME, a SIZE, az EXTENT, valamint a DATA paraméterekhez szükséges értékeket.

```
12              Dim m1 = (From m In dc.rastermaps _
13                      Where m.autoinc = k _
14                      Select m).First
15
16              Dim fajlnev As String
17              Dim kiterj As String
18              Dim eleres As String
19              Dim cim As String
20              Dim mag As String
21              Dim szel As String
22              Dim sizemag As Integer
23              Dim sizeszel As Integer
24
25              mag = m1.height.ToString
26              szel = m1.width.ToString
27
28              sizeszel = m1.width
29              sizemag = m1.height
30
31              Do Until sizemag < 600
32                  sizemag = sizemag * 0.99
33                  sizeszel = sizeszel * 0.99
34              Loop
35
36              fajlnev = m1.map_id
37              kiterj = m1.grformat
38              eleres = "C:\OSGeo4W\apache\htdocs\umn\" & fajlnev & "." & "map"
39              cim = m1.cim
```

47. ábra: A ciklusmag első fele

A ciklus második felében a szöveges állományunk megalkotása történik. Ehhez mindenekelőtt egy ***StreamWriter*** eszközt kell deklarálnunk, mely a ***File*** objektum ***CreateText*** eljárásán keresztül adja át az adatot az argumentumban megadott elérési útvonalban megjelölt új állománynak. Ez az elérési út tartalmazza annak a mappának az elérési útvonalát, melyben a raszteres állományokat tároljuk, valamint a fájlnevet, mely meg fog egyezni az adattáblánk elsődleges kulcsával, valamint a kiterjesztést, mely a .map értéket fogja felvenni. Ez összességében azt jelenti, hogy a megadott mappában létrejön egy új szöveges állomány, melynek neve megegyezik a raszteres állomány fájlnevével, kiterjesztése a .map lesz, tartalma azonban teljesen üres lesz.

Az adatok feltöltése a ***StreamWriter*** eszköz ***Write*** eljárásával történik, és a zárójelek között megadott értékekkel töltődik fel az állományba. Amennyiben egyszerű szöveget szeretnénk az állományba „beleírni”, úgy a kívánt szövegrészletet idézőjelek közé kell tennünk. Ha string típusú változó értékét szeretnénk hozzáadni, akkor csak az illető azonosítót kell a zárójelek közé tennünk. Ha pedig sort szeretnénk léptetni, akkor a ***VbCrLf*** állandót kell meghívnunk. A ***Write*** eljárásba kerülő egyes elemeket az & jel segítségével tudjuk egymástól elválasztani.

```
41              Dim sw As StreamWriter = File.CreateText(eleres)
42
43              Dim ij As String = """
44
45              sw.Write("MAP" & vbCrLf & "  NAME ")
46              sw.Write(ij & fajlnev & ij & vbCrLf)
47              sw.Write("  STATUS ON" & vbCrLf)
48              sw.Write("  SIZE " & sizeszel.ToString & " " & sizemag.ToString & vbCrLf)
49              sw.Write("  EXTENT 0 0 " & szel & " " & mag & vbCrLf)
50              sw.Write("  UNITS DD" & vbCrLf)
51              sw.Write("  IMAGECOLOR 255 255 255" & vbCrLf & vbCrLf)
52              sw.Write("  OUTPUTFORMAT" & vbCrLf & "    NAME 'AGG'" & vbCrLf & "    DRIVER AGG/PNG" & vbCrLf &
"    IMAGEMODE RGB" & vbCrLf & "  END" & vbCrLf & vbCrLf)
53              sw.Write("  WEB" & vbCrLf)
54              sw.Write("        IMAGEPATH " & ij & "C:\OSGeo4W\apache\htdocs\umn\tmp\" & ij & vbCrLf)
55              sw.Write("        IMAGEURL " & ij & "/umn/tmp/" & ij & vbCrLf)
56              sw.Write("        TEMPLATE templ_mymap.html" & vbCrLf)
57              sw.Write("  END" & vbCrLf & vbCrLf)
58              sw.Write("  LAYER" & vbCrLf & "    NAME " & "nevtelen" & fajlnev & vbCrLf & "    TYPE RASTER" &
vbCrLf & "    STATUS DEFAULT" & vbCrLf)
59              sw.Write(vbTab & "DATA " & ij & fajlnev & "." & kiterj & ij & vbCrLf & "  END" & vbCrLf & vbCrLf)
60              sw.Write("END")
61
62              sw.Flush()
63              sw.Close()
```

48. ábra: A ciklusmag második része

Miután a szöveges állományunkat feltöltöttük a kívánt tartalommal a ***Write*** eljárás segítségével, két további argumentum nélküli eljárását kell meghívnunk a ***StreamWriter*** eszköznek, ezek a ***Flush***, illetve a ***Close***. Ezek arra szolgálnak, hogy formálisan is lezárják a szöveges adatok feltöltését a kívánt állományba.

Ha a ciklusunkat minden egyes elemre végrehajtjuk, akkor ez azt eredményezi, hogy minden raszteres állományhoz automatikusan létrejön a megfelelő mapfájl. Így már csak a html fájlokat kell elkészítenünk, melyeknek megalkotása nagyban hasonlít az imént közölt mapfájlok generáláshoz.

V.7. A html fájlok generálása

Az előző fejezetben áttekintettük, hogy egy adattábla adatai alapján hogyan tudunk raszteres állományokhoz tömegesen mapfájlokat készíteni. Mivel az egyes elemek megjelenítéséhez keretet adó html kód generálása is nagyon hasonló módon történik, ezért erről most csak vázlatosan értekezünk.

Ebben az esetben is a ***DataContext*** objektumon keresztül kérjük le a kívánt adatokat, ezek értékeit string változókban tároljuk, majd a ***StreamWriter*** eszköz segítségével írjuk bele a megfelelő helyen generált .html kiterjesztéssel rendelkező szöveges állományba. Mivel a ciklusmagunk első része megegyezik az előző fejezetben használt ciklusmagéval, ezért csak a ciklusmag második felét közöljük.

```
54              sw.Write("<!DOCTYPE html>" & vbCrLf)
55              sw.Write("<html>" & vbCrLf)
56              sw.Write("<head>" & vbCrLf)
57              sw.Write("<script type=" & ij & "text/javascript" & ij & " src=" & ij & "../umn/js/
jquery-1.4.4.min.js" & ij & "></script>" & vbCrLf)
58              sw.Write("<script type=" & ij & "text/javascript" & ij & " src=" & ij & "../umn/js/jquery-
ui.min.js" & ij & "></script>" & vbCrLf)
59              sw.Write("<script type=" & ij & "text/javascript" & ij & " src=" & ij & "../umn/js/
jquery.mousewheel.min.js" & ij & "></script>" & vbCrLf)
60              sw.Write("<script type=" & ij & "text/javascript" & ij & " src=" & ij & "../umn/js/smm.js" & ij &
 "></script>" & vbCrLf & vbCrLf)
61              sw.Write("<script>" & vbCrLf)
62              sw.Write("    var probaMap;" & vbCrLf & vbCrLf)
63              sw.Write("    function init()" & vbCrLf)
64              sw.Write("    {" & vbCrLf)
65              sw.Write("        probaMap = new SamanMapserverMap(" & vbCrLf)
66              sw.Write("          document.getElementById('map_div')," & vbCrLf)
67              sw.Write(vbTab & ij & "http://localhost/cgi-bin/mapserv.exe?map=C:\\OSGeo4W\\apache\\htdocs\\umn\
\" & fajlnev & ".map&mode=map" & ij & "," & vbCrLf)
68              sw.Write("  0, 0, " & szel & ", " & mag & ");" & vbCrLf)
69              sw.Write("    }" & vbCrLf & vbCrLf)
70              sw.Write("</script>" & vbCrLf)
71              sw.Write("</head>" & vbCrLf & vbCrLf)
72              sw.Write("<body onload=" & ij & "init()" & ij & ">" & vbCrLf)
73              sw.Write("<h2  style ='text-align:center '>" & cim & "</h2>" & vbCrLf)
74              sw.Write("<div id=" & ij & "map_div" & ij & " style='width:" & sizeszel.ToString & "px; height:"
& sizemag.ToString & "px; margin: 0px auto 0px auto; border: 2px solid black;'></div>" & vbCrLf)
75              sw.Write("</body>" & vbCrLf & vbCrLf)
76              sw.Write("</html>")
77
78              sw.Flush()
79              sw.Close()
```

49. ábra: A ciklusmagunk második fele

A ciklusunk megfelelő számú végrehajtásának az eredményeképpen létrejönnek a megjelenítést biztosító html állományaink, így már csak az ASP.NET-ben elkészített alkalmazásunk ***GridView*** vezérlőjével kell a kapcsolatot megteremtenünk.

V.8. A megjelenítés összekapcsolása a GridView vezérlővel

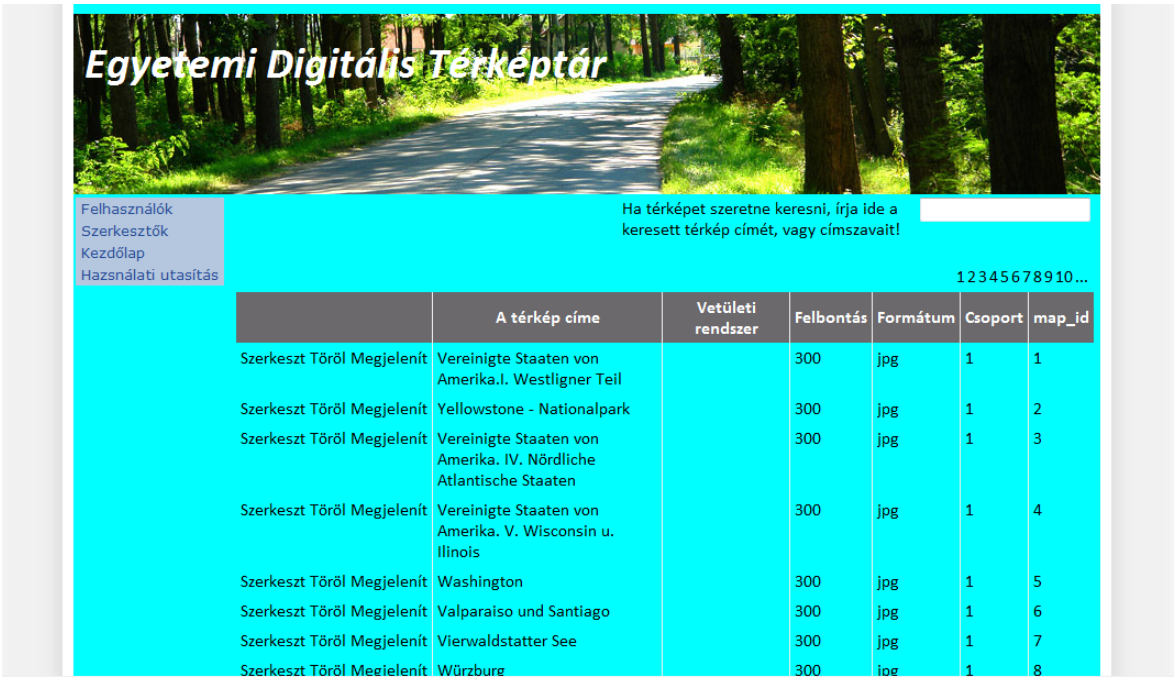
Az előző fejezetekben áttekintettük az egyes térbeli adatokat tartalmazó raszteres állományok megjelenítési lehetőségeit. Befejező lépésként az egyes rekordokat tartalmazó GridView vezérlőnket szeretnénk összekötni a megjelenítést biztosító html oldalakkal. Célunk, hogy a vezérlőben egy adott azonosítóval rendelkező elem kiválasztásával meghívásra kerüljön az a html oldal, melynek azonosítója (fájlneve) megegyezik a kiválasztott elemével.

```
37      Protected Sub GridView1_SelectedIndexChanged(ByVal sender As Object, ByVal e As EventArgs) Handles
GridView1.SelectedIndexChanged
38          Dim fajlnev As String
39          fajlnev = GridView1.SelectedValue.ToString
40
41          Dim cim = "http://localhost/umn/" & fajlnev & "." & "html"
42
43          Response.Redirect(cim)
44      End Sub
```

50. ábra: A megjelenítés meghívása

A ***GridView*** vezérlőben található egy tetszőleges elem kiválasztásakor meghívott eljárás során a fenti értékadások történnek meg, valamint meghívódik a ***Response.Redirect*** eljárás. Utóbbi a zárójelek között található URL címre irányítja a felhasználót. Ez esetben ez egy cim azonosítóval rendelkező string, mely tartalmazza a html fájlok helyét, és benne paraméterezve jelenik meg az egyes fájlok azonosítását szolgáló fajlnev paraméter, melynek értékét a ***GridVew.SelectedIndex*** határozza meg.

Természetesen az elkészült és működő alkalmazásunk színvonalát tovább emelhetjük egy szép megjelenés kialakításával, mely igazán barátságossá teheti a felhasználói környezetet.



	A térkép címe	Vetületi rendszer	Felbontás	Formátum	Csoport	map_id
Szerkeszt Töröl Megjelenít	Vereinigte Staaten von Amerika.I. Westligner Teil		300	jpg	1	1
Szerkeszt Töröl Megjelenít	Yellowstone - Nationalpark		300	jpg	1	2
Szerkeszt Töröl Megjelenít	Vereinigte Staaten von Amerika. IV. Nördliche Atlantische Staaten		300	jpg	1	3
Szerkeszt Töröl Megjelenít	Vereinigte Staaten von Amerika. V. Wisconsin u. Ilinois		300	jpg	1	4
Szerkeszt Töröl Megjelenít	Washington		300	jpg	1	5
Szerkeszt Töröl Megjelenít	Valparaiso und Santiago		300	jpg	1	6
Szerkeszt Töröl Megjelenít	Vierwaldstatter See		300	jpg	1	7
Szerkeszt Töröl Megjelenít	Würzburg		300	jpg	1	8

51. ábra: Az alkalmazásunk működés közben

VI. Összefoglalás

Diplomamunkámban megvizsgáltam annak a lehetőségét, hogy miként kapcsolható össze egy webes alkalmazás és egy nyílt forráskódú térképszerver annak érdekében, hogy egy hatékony eszközt biztosítson egy digitális térképtár működéséhez.

Az egyes térbeli adatokat tartalmazó raszteres állományainkhoz kapcsolódó adatokat, illetve adatbázist a Microsoft SQL Server 2008 R2 programcsomag segítségével kezeltük, mely ennek a célnak tökéletesen megfelelt. Az adatkezeléshez használt Management Studio könnyű kezelésének és átlátható működésének köszönhetően nagyban segítette a megvalósítást. Bemutattam annak a lehetőségét is, hogy miként használhatunk és konvertálhatunk egy más adatszolgáltatón található adatbázist; ezt az SQL Server csomag Migration Assistance for MySQL nevű eszközével valósítottam meg.

A digitális térképtárunk webes felületének a megalkotásához a .NET keretrendszert, illetve az ASP.NET szerveroldali alkalmazás-készítő eszközt használtam. A fejlesztői környezetben létrehozott vezérlők segítségével szemléltettem, hogy milyen adat-manipulációs lehetőségekkel ruházhatjuk fel a webes felhasználói felületünket.

A térképeink webes megjelenítéséhez a Mapserver eszközt használtam, melynek sokrétű és szerteágazó lehetőségei közül néhány alapvető felhasználási formát mutattam be. Egy tetszőleges html oldalba való beágyazási lehetőségeket is megvizsgáltam. A nagyszámú térképi adatból következő rengeteg szükséges szöveges állomány (mapfájl, illetve html kód) elkészítésére külön programokat írtam, ezzel lehetővé vált ezeknek a tömeges elkészítése, illetve generálása.

Az egyes munkarészek összekapcsolását az ASP.NET rendszerben dolgoztam ki, melynek eredményeképpen létrejött egy olyan alapvető funkciókkal rendelkező nagymennyiségű térképi tartalmat kezelő alkalmazás, mely jó alapot teremthet egy komolyabb funkcionalitással rendelkező ilyen rendszer kidolgozásának.

Források:

http://blogs.msdn.com/b/ssma/archive/2011/02/05/migrating-mysql-database.aspx

http://devportal.hu/content/aspnet16homework.aspx

http://mapserver.org/

http://www.asp.net/linq/videos/how-do-i-linq-to-sql-overview

http://davidhayden.com/blog/dave/archive/2008/02/21/LINQToSQLSQLServer2008.aspx

http://prog.hu/cikkek/836/JavaScript+alapok/oldal/2.html

http://elbandi.web.elte.hu/sz2/

http://www.w3.org/TR/html5/introduction.html#introduction

http://www.vbmysql.com/articles/vbnet-mysql-tutorials/the-vbnet-mysql-tutorial-part-3

Köszönetnyilvánítás

Szeretnék köszönetet mondani mindazoknak, akik segítséget nyújtottak jelen Diplomamunka megírásában. Elsősorban szeretném köszönetemet kifejezni Elek István egyetemi docensnek, aki türelmes odafigyeléssel és hasznos észrevételekkel segítette munkámat. Hálámat szeretném kifejezni továbbá Dombóvári Eszternek, aki a Mapserver alkalmazás kezdeti lépéseibe bevezetett, illetve a dolgozat írása alatt folyamatosan értékes tanácsokkal látott el. Köszönettel tartozom továbbá Gede Mátyás adjunktusnak, aki a problémára való nagyszerű rálátásával segített munkámban, valamint a megjelenítéshez rendelkezésemre bocsájtotta saját fejlesztésű JavaScript állományát, mely nagyban emelte az alkalmazásom színvonalát.

Tartalomjegyzék