SZAKDOLGOZAT

Gyarmati Ottó

Debrecen

2007

Debreceni Egyetem
Informatika Kar

Az ASP.NET

Témavezető:
Dr. Kuki Attila
Egyetemi adjunktus

Készítette:
Gyarmati Ottó
Programozó
matematikus

Debrecen
2007

Tartalomjegyzék

Bevezetés....4
1. A .NET felépítése....5
2. Az ASP és az ASP.NET összehasonlítása....7
3. Az ASP.NET és az IIS....9
4. Az ASP.NET kérések útja....10
5. Az ASP.NET fordítási modellje....12
6. A lapesemények....12
7. A nézetállapot....13
8. Az ASPX lapok struktúrája....14
9. A vezérlőelemek....18
9.1. A HTML vezérlők....19
9.1.1. A konténervezérlők....20
9.1.2. A beviteli vezérlők....22
9.1.3. Az ImageVezérlő....25
9.2. A Web vezérlők....25
9.2.1. Az alapvezérlők....27
9.2.2. Az adatérvényesítés és az érvényesítő vezérlők....33
9.2.3. A gazdag vezérlők....35
9.2.4. A listavezérlők....37
9.2.5. Az adatkezelő vezérlők....38

9.2.6. Az Xml vezérlő.....38
9.2.7. A MultiView és a View vezérlőelemek.....39
10. A Mono.....39
Irodalomjegyzék.....40

Bevezetés

Szakdolgozatomat a Microsoft által kifejlesztett, ASP technológia továbbfejlesztett változatából, ASP.NET-ből írom. Ez egy HTTP-kéréseket feldolgozó keretrendszer. Jelen szakdolgozat írásakor az ASP.NET 2.0-ás verziója van érvényben. Az ASP.NET elődjéhez képest hatalmas változáson ment át, ezzel kibővítve a webes alkalmazásfejlesztés lehetőségeit. A .NET keretrendszer segítségével egyszerűbb, míg maga az ASP technológia kibővítésével hatékonyabb lett az alkalmazás-fejlesztés. Annak ellenére, hogy az ASP.NET a nevét a Microsoft korábbi web fejlesztési technológiájáról, az ASP-ről kapta, a két verzió nagyban különbözik. A Microsoft teljes mértékben újratervezte az ASP.NET-et, aminek során a minden Microsoft .NET alkalmazás által közösen használt CLR tulajdonságait tartotta szem előtt. A programozó bármely, a .NET Keretrendszer által támogatott nyelven írhat ASP.NET kódot, de ez általában Visual Basic .NET vagy C# nyelven történik. Az ASP.NET működése gyorsabb is, mivel az egész weboldalt előfordítja egy vagy több DLL-be a webszerveren, így a weboldal gyorsabban fut a korábbi scriptelési technológiához képest.

Az ASP.NET megkönnyíti a fejlesztők áttérését a Windows alkalmazás fejlesztés területéről a webalkalmazás fejlesztés területére, mivel komponens-alapú fejlesztést tesz lehetővé hasonlóan a Windows felhasználói felületéhez. Egy web vezérlő - mint a gomb vagy a címke - nagyon hasonlóan működik a Windowsos megfelelőjéhez: a kód módosíthatja a paramétereit és válaszolhat az eseményekre. A vezérlők tudják, hogy hogyan jelenítsék meg saját magukat: ahogyan a Windows vezérlők saját magukat rajzolják a képernyőre, a web vezérlők HTML részleteket állítanak elő a felhasználónak visszaküldendő oldal részeként.

Az ASP.NET az eseményvezérelt GUI paradigma felé próbálja irányítani a fejlesztőt a hagyományos scriptelés helyett. A keretrendszer megpróbálja kombinálni a meglévő technológiákat (mint a JavaScript vagy VBScript) a belső komponensekkel (mint a Viewstate), hogy állapottartó környezetet biztosítson a web alapvetően állapotmentes környezetében.

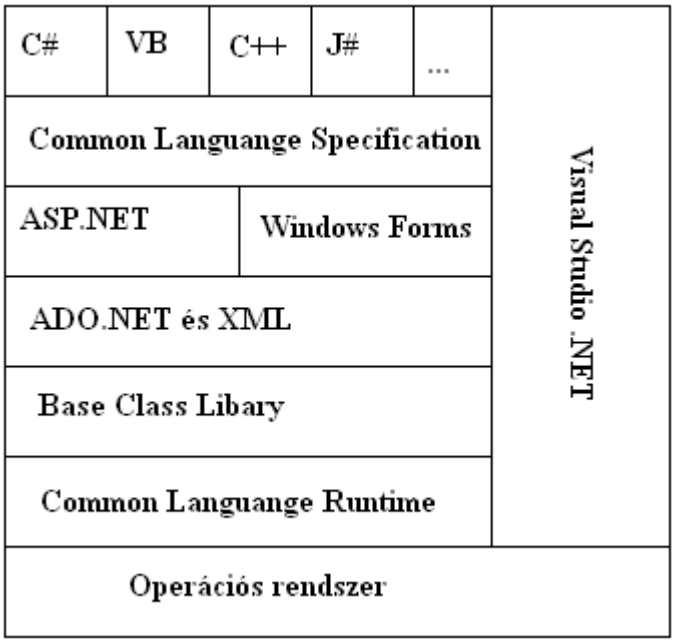
Az ASP.NET a .NET Keretrendszert infrastruktúraként használja. A .NET Keretrendszer egy felügyelt futtatási környezetet biztosít (mint a Java), JIT-tel rendelekző virtuális gépet és osztálykönyvtárat biztosítva.

1. A .NET felépítése

A .Net egy software, amely információkat, embereket, rendszereket és eszközöket kapcsol össze. Ez egy új generációs technológia, amely Web szolgáltatásokon alapul – kis alkalmazásokon, amelyek képesek kapcsolatba lépni egymással csakúgy, mint nagyobb méretű alkalmazásokkal az Internet-en keresztül.

Más megközelítésben a .Net egy programfejlesztési környezet, mely számtalan hasznos szolgáltatással segíti a programozók mindennapi munkáját.

A .Net keretrendszer vázlatos felépítése:



1.1 ábra: A .Net keretrendszer

Legalsó szinten van az operációs rendszer. Egy jó operációs rendszer nem engedi, hogy az ő általa felügyelt programok őt megkerülve kapcsolatba lépjenek egymással, csakis őrajta keresztül kommunikálhatnak, illetve használják a különböző erőforrásokat.

Az operációs rendszer felett van a CLR – Common Language Runtime -, amely a .Net keretrendszer legfontosabb része. A CLR biztosítja, hogy a .Net segítségével elkészített programok platformfüggetlenek legyenek. A CLR ellenőrzi a futtatandó kódot és adatokat, és biztosítja a különböző szolgáltatásokat az alkalmazás számára. A szolgáltatások biztosítása egy virtuális géppel történik. Ezt nevezzük felügyelt futtatásnak. A felügyelt futtatás során a futtatókörnyezet felügyelt adatokat kezel. A futtatókörnyezet helyezi el az adatokat a memóriában, és szükség szerint fel is szabadítja a Garbage Collector (hulladékgyűjtő)

segítségével. A felügyelt futtatás során a futtatórendszer felügyeletet hajt végre. A futtatandó kódot a CIL (Common Intermediate Language) köztes nyelven megfogalmazott kód reprezentálja. A felügyelt futtatás során a JIT (Just In Time) fordító alakítja át a CIL kódot natív kódra. Tehát nem az IL kód kerül futtatásra, azt még egyszer át kell alakítani futás előtt egy platform-támogatott kódra. Ezt a folyamatot nevezzük JIT fordításnak. Ez a fordítás történhet akkor, amikor épp installáljuk a programot, vagy közvetlenül a futás előtt, és biztosít, hogy a lehető leghatékonyabban fog futni a kód. A .NET több különböző JIT fordítóval rendelkezik, amelyek különböző környezetekre vannak specializálva, attól függően, hogy a fordítás vagy a futás legyen a leggyorsabb.

A .Net egy új API-nak tekinthető. A programozónak elvileg semmilyen más API-t nem kell ismernie, hiszen ő már nem használja az adott operációs rendszer API-ját. A második réteg tartalmazza a .Net API nagy részét, ez a réteg a Base Class Libary (BCL). Itt találhatóak azok a szolgáltatások, melyeket a programozó felhasználhat programjának fejlesztése során. A régebbi API-k csak eljárásokat és függvényeket tartalmaznak, ezzel szemben a BCL strukturált, névterekbe és osztályokba szervezett sok ezer szolgáltatást tartalmaz. Ezáltal a felhasználás hatékonyabb és áttekinthetőbb lett.

A következő réteg az ADO.NET (Active Data Object) és az XML, amely a 21. századi adatelérési technológiákat tartalmazza. Ezeket a technikákat használják fel az alkalmazások az adatok tárolására, elérésére és módosítására. A rétegek segítségével tudnak az alkalmazások adatokat tárolni illetve elérni különböző háttértárolókon.

A következő réteg kétfelé válik. Az egyik a Windows Forms, ekkor az alkalmazásunkat hagyományos interaktív grafikus felületen valósítjuk meg. Ez tartalmazza azon API készletet, melyek segítségével grafikus felületű, ablakos, interaktív alkalmazásokat készíthetünk. A réteg másik fele az ASP.NET, ekkor web-es felületen valósítjuk meg az alkalmazásunk felhasználói felületét. Nem kifejezetten interaktív programok írására való. Ezen programok futtatásához valamilyen web szerver, kliens oldalon pedig egy internetes tallózó program szükséges.

A következő réteg a CLS (Common Languange Specification). Ez a réteg definiálja azokat a jellemzőket, melyeket a .Net által támogatott programozási nyelvek, a fejlődésük során különböző módon értelmeznek. Ez a réteg rögzíti az alaptípusok méretét, tárolási módját, a tömböket.

A CLS réteg fölött helyezkednek el a különböző programozási nyelvek és fordítóprogramjaik. A .NEt nem épül rá egyetlen nyelvre sem, nyelvfüggetlen. Elvileg bármely olyan programozási nyelven lehet .Net-es programot írni, amely tud a CLR virtuális gép kódjára fordítani, vagyis ismeri a CLS követelményeit.

Összefoglalva azért jó .Net-ben programozni, mert a megírt alkalmazásunk független lesz az operációs rendszertől, a számítógép hardware-től. Nem kell új programozási nyelvet sem megtanulnunk, ha azt támogatja a .Net. Kihasználhatjuk az automatikus memória-menedzselés szolgáltatásait (Garbage Collector). Felhasználhatjuk a programfejlesztéshez az igen széles szolgáltatást nyújtó rendszert, a Base Class Libary-t, ami igencsak lecsökkenti a fejlesztési időt.

Összességében a .Net keretrendszerről elmondhatjuk, hogy nyelvsemleges, ami azt jelenti, hogy egy feladatot bármely .Net nyelvvel megoldhatjuk, illetve ezeket a különböző nyelven megírt modulokat összekapcsolhatjuk. A .Net rendszersemleges, tehát minden olyan rendszer alatt futtathatók az alkalmazások, amely rendelkezik a .Net futáskörnyezetével. Ezt pedig elvileg minden rendszer számára el lehet készíteni. A .Net platformsemleges is, vagyis mindegy, hogy egy alkalmazást egy PC-n, mobil készüléken, esetleg PDA-n futtatjuk, az eredmény ugyanaz lesz.

2. Az ASP és az ASP.NET összehasonlítása.

A webalkamazások felhasználói felületét leggyakrabban a weblapokon elhelyezett szerveroldali vezérlőkkel valósítható meg. Ezek a szerveroldali vezérlők helyettesítik a szerveren a hagyományos HTML elemeket és szerver oldali kódból programozhatók. Tulajdonságaikat megszabhatjuk, eseményeket, eseménykezelőket, metódusokat rendelhetünk hozzá, amit szerver oldali kódból vezérelhetünk. Legfontosabb szerepük a böngésző által megértett és értelmezett HTML kód generálása.

Az ASP (Active Server Page) a Microsoft által megalkotott technológia. Nagyon közkedvelt, sok web szerver használja A klasszikus ASP és az ASP.NET a webalkalmazások fejlesztését támogatja. ASP.NET a .Net technológia egy olyan komponense, amely segítségével jelentősen egyszerűsödik a webalkalmazások írása.

Egy ASP egy HTML lap tartalma lehet, ami lényegében egy script-nyelvi kódrész, ezzel bővítve egy HTML lap szolgáltatását. Ha egy lapot letöltünk, akkor a script kód

végrehajtódik, a kód előállít egy kimenetet, melyet visszaküld a kliensnek HTML lapként. A statikus HTML és a script kód keverése lehetővé teszi a Web lapok testre szabását. A kóddal generálható egy szokásos HTML oldal, úgy, hogy figyelembe vesszük a felhasználói bemenetet. A script kód a hagyományos ASP lapokon interpreteres, ami nem biztosítja a legjobb teljesítményt.

Az ASP.NET lap kódja már le van fordítva, ezáltal nagyobb lesz a teljesítmény kihasználás. Ez a gyakorlatban annyit jelent, hogy, ha először töltünk le egy ASP.NET lapot, akkor az lassabb lesz mint egy ASP lap, de újbóli letöltéskor már gyorsabban fog az ASP.NET lap megjelenni. Ennek az a magyarázata, hogy az ASP lap minden letöltéskor értelmezve lesz, addig az ASP.NET lap első letöltésekor le lesz fordítva, majd a következő letöltésnél már a lefordított változata lesz végrehajtva. Ez rengeteg előnnyel jár. Ezekről még később részletesebben lesz szó az ASP.NET fordítási modelljéről szóló fejezetben.

Mivel a szerveroldali kód le van fordítva, ezért nincs különbség a lefordított komponensek és a lapba ágyazott kódrészletek végrehajtása között. Nincs különbség a HTML elemeket és szerveroldali kódtömböket váltakozva tartalmazó lapok, illetve a szerveroldali kódot egy nagy tömbbe gyűjtő lapok között.

A lefordított szerveroldali kód mások nagy előnye az, hogy az összes szintaktikai hiba még fordítás alatt kiderül, nem pedig futás közben. Ugyanolyan hibakeresési eljárásokat használhatunk, mint egy hagyományos alkalmazásnál.

Egy ASP.NET lap esetén egy tapasztaltabb webfejlesztőnek az tűnhet fel igazán, hogy itt minden egy osztály, szemben a klasszikus ASP-vel. Ennek az a magyarázata, hogy az ASP.NET a .NET szerves része, ami egy objektumorientált technológia.

Lényeges eltérés még az ASP-től a folyamatmodell (process modell). Az ASP.NET üzemfolyamata (worker process) teljesen el van különítve az IIS (Internet Information Service) folyamatától. Az ASP.NET webalkalmazásokat egyáltalán nem kötelező IIS szerverre telepíteni.

ASP-ben meg vannak határozva a használható script nyelvek, ez általában a VBA vagy a JScript. ASP.NET-ben pedig bármely .NET által támogatott nyelven lehet programot írni. Tehát használható mind a C++, C#, vagy akár a COBOL is.

Az ASP lapokban a HTML és a kód keverve van, ami igen megnehezíti a programírást. El lehet különíteni a kódot a HTML elemektől egy kis trükkel úgy, hogy a kódot egy

különálló fájlba eltároljuk, és úgynevezett <include> tagok használatával az ASP lapban beépítve fel tudjuk használni. Ezt nem igazán támogatja maga az ASP sem. Az ASP.NET megoldotta ezt a problémát az úgynevezett mögöttes kód (code behind) koncepcióval. Ezáltal elkülönül a lap kinézetét meghatározó forráskód, és egy külön fájlban lesz a Visual Studio.NET-ben megírt rész. Mindegyik Web Form létrhoz két fájlt a web szerveren: Egy fő fájlt, amely tartalmazza megjelenítendő felület szerkezetét, ennek .aspx a kiterjesztése, és a mögöttes kódot, amely a fejlesztő által választott programozási nyelven van megírva, ennek a kiterjesztése függ a választott programozási nyelvtől (C# esetén .asxp.cs, Visual Basic esetén .aspx.vb…). A mögöttes kód használatának egyik nagy előnye az, hogy mivel egy lap tartalmát és kinézetét különböző emberek fejlesztik, így könnyebb a lapok fejlesztése.

ASP.NET 2.0 magasabb szintre emeli ASP.NET-et. A leggyakrabban használt szolgáltatásokon túl még több szolgáltatást tol a keretrendszerbe, ezzel tökéletesítve az ASP.NET-et. Ilyen pluszszolgáltatás az engedélyezési alrendszer. (Az ASP.NET korábbi verziójának elfogadható, könnyen kezelhető hitelesítési modellt tartalmaz, de a fejlesztőknek gyakran saját hitelesítési rendszereket kellett megvalósítaniuk az egyes webhelyeken.)

3. Az ASP.NET és az IIS

Miután egy felhasználó az URL begépelése után megnyomta az Enter billentyűt, a böngésző egy HTTP GET kérést küld a célwebhely felé. A kérés az útvonalválasztókon keresztül megérkezik a webkiszolgálónkra, ahol egy szoftver figyeli - a 80-as porton - beérkező kérést. Microsoft platformon a 80-as portot figyelő szoftver általában az IIS (Internet Information Service). Az ASP.NET az IIS 5.0-ás és a 6.0-ás verzióival működik. Ha IIS 5.0-át használunk, akkor az IIS tartja nyilván a fálj kiterjesztések és a kéréseket értelmező ISAPI (Internet Services API) DLL-ek leképezéseit. Amikor beérkezik egy kérés a webhelyre, akkor az IIS a kérést a megfelelő ISAPI DLL- hez továbbítja. ISS 6.0 esetben, a 80-as portot egy kernelszintű program a HTTP.SYS figyeli, és ez továbbítja a kéréseket a megfelelő ISAPI DLL-nek.

Minden ISAPI DLL három jól definiált belépési ponttal rendelkezik. Ezek közül a legfontosabb a HttpExtensionProc. Amikor egy kérés beérkezik, és azt az ISS értelmezni tudja, akkor az IIS az összes környezeti információt az EXTENSION_CONTROL_BLOCK struktúrába csomagolja. Az IIS meghívja a kapcsolódó DLL HttpExtensionProc metódusát, és

átadja neki az EXTENSION_CONTROL_BLOCK mutatóját. Az ASP.NET-alkalmazások futtatása során az IIS több megkülönböztetett fájlkiterjesztést ismer. A kiterjesztéseket magukba foglaló kérések az ASP.NET ISAPI DLL-jének az aspnet_isapi.dll-nek továbbítódnak. Ilyen kiterjesztések például a .aspx, .asax, .asmx, .ashx. Ha a keresett oldal neve megfelel az ASP.NET egyik kiterjesztésének, akkor az IIS a kérést az asp_netisapi.dll-hez továbbítja. Ez a dll tartalmazza a szükséges exportokat, mint például a httpExtensionProc metódus.

4. Az ASP.NET kérések útja

A kérések folyamata az ASP.NET futószalagon vezet keresztül. Ezt a futószalagot az ASP.NET dolgozófolyamata kezeli. A dolgozófolyamat IIS 6.x verzióban a w3wp.exe, 5.0 verzióban pedig az asp-net_wp.exe. A dolgozófolyamat helyettesítő folyamatként működik, amely az ASP.NET történéseit szolgálja ki.

Ha egy kérés beérkezik az ASP.NET-futtatórendszer által kezelt AppDomain osztályba, az ASP.NET a HttpWorkerRequest osztály segítségével tárolja a kérés információit. Ezt követően a futtatórendszer az információkat a HttpContext osztályba csomagolja. A HttpContext osztály, az aktuális kérés HttpRequest és HttpResponse objektumait is beleértve, a kéréssel kapcsolatos összes információt magában foglalja. A futtatórendszer előállítja a HttpApplication osztály egy példányát, és kivált néhány alkalmazásszintű eseményt. Ilyen esemény lehet például a BeginRequest vagy az AuthenticateRequest. Az eseményekről a futószalaghoz csatlakozott HttpModules modulok is értesülnek. Ezek után dönt arról azASP.NET, hogy milyen kezelő alkalmas a kérés kezelésére. Létrehoz egy ilyen kezelőt, és felkéri a kérés feldolgozására. Miután a kezelő foglalkozik a kéréssel, az ASP.NET a HttpApplication objektumon és a HttpModules modulon keresztül kivált néhány utófeldolgozási eseményt. Ilyen például az EndRequest esemény.

A futószalag néhány része közvetlenül elérhető, ezek a futószalagon keresztülhaladó kérések hasznos eszközei. Ilyen eszközök például a HttpApplication, HttpContext, HttpModules, HttpHandler.

A HttpApplication objektumok feladata az alkalmazásszintű adatok tárolása és az alkalmazásszintű események kezelése. Erre azért van szükség, mert a HTTP kezelők rövid életűek, a kérés kezelésének ideje alatt a rendszerben maradnak, azután eltűnnek. Ez egyszerű

alkalmazások esetén jól működik, de ezekkel az illékony kezelőkkel lehetetlen alkalmazás szintű funkcionalitást biztosítani.

A Httpcontext osztály központi helyet foglal el. Segítségével a futószalagon keresztülhaladó aktuális kérés bizonyos részeihez férhetünk hozzá. Ezzel az aktuális kérés összes tulajdonsága a rendelkezésünkre áll. Valójában a HttpContext összetevők rendszerint a futószalag más részeit jelentik, de a kérés kezelését nagymértékben megkönnyíti, hogy ezek az információk egyetlen helyen állnak rendelkezésünkre. A HttpContext az alábbi tulajdonságokat foglalja magába:

- A környezet Response objektumának hivatkozása, hogy a kimenetet elküldhessük az ügyfélnek.
- A Request objektum hivatkozása, hogy magáról a kérésről kereshessünk információkat.
- A központi alkalmazás hivatkozása, hogy megállapíthassuk az alkalmazásállapotot.
- Az alkalmazásszintű gyorsítótár hivatkozása. Ez adattárolásra és az adatbázis felesleges körbejárásainak elkerülésére szolgál.

A HttpModule-ok nagyobb teljesítményű apparátust biztosítanak bizonyos alkalmazásszintű feladatok elvégzéséhez. Az ASP.NET magában foglal néhány előre definiált HtpppModule-t. A munkamenetállapotot, a hitelesítést és felhatalmazást a HttpModule-ok kezelik. A HttpModule-ok elkészítése elég egyértelmű, és a modulok segítségével az összetett alkalmazásszintű műveletek kezelése egyszerű feladat. Ha például saját hitelesítési sémát kell írnunk, a HttpModule nagyban megkönnyíti a munkánkat.

A HttpHandler a futószalagon érkező kérés utolsó megállóhelye. Az interfészt megvalósító osztályokat az IHttpHandler kezelőnek tekinti. Amikor a kérés a futószalag végére érkezik, az ASP.NET megvizsgálja a konfigurációs fájlt, és megnézi, hogy az adott fájlkiterjesztést leképeztük-e a HttpHandler kezelőre. Ha igen, akkor az ASP.NET betölti a megfelelő kezelőt, és a kérés IHttpHandler.ProcessRequest metódusának meghívásával végrehajtja a kérést. Az ASP.NET előredefiniált HttpHandler-eket foglal magában, ilyen például a System.Web.UI.Page vagy a System.Web.Services.WebServices kezelők, de készíthetünk teljesen új, HttpHandler-t is.

5. Az ASP.NET fordítási modellje

A Microsoft egyik leglényegesebb fejlesztése az ASP fejlesztői környezetével kapcsolatban az, hogy a webes kéréskezelő keretrendszert osztályokból építette fel. A kéréskezelés osztályalapú architektúrába helyezése olyan webkezelési keretrendszer létrehozását teszi lehetővé, amely fordítható. Az ASP.NET-lapok első lehívása során a rendszer lefordítja azokat különböző szerelvényekre. Ebből az architektúrából származó hatalmas előny az, hogy a laphoz történő következő hozzáféréskor a lapot közvetlenül a szerelvényből lehet betölteni. Mivel a .NET keretrendszer a lapokat szerelvényekre fordítja, nemcsak a teljesítmény növelhető, hanem megbízhatóság is. Továbbá a webes kérések keretrendszerének fordítása robusztusabb és konzisztensebb hibakeresést tesz lehetővé.

Az ASP.NET az .aspx fájlokat automatikusan fordítja, ehhez nem kell mást tennünk, mint megkeresni a kódot tartalmazó .aspx fájlt. Ekkor az ASP.NET az oldalt egy osztályra fordítja. A fordítás során eredményül kapott szerelvényeket az ASP.NET egy ideiglenes könyvtárba másolja.

6. A lapesemények

A lapunk a Page osztály egy leszármazottja lesz. A Page osztály több eseménnyel rendelkezik, melyeket a lapunkat képviselő osztály is örököl. Ezeket az eseményeket nevezzük lapeseményeknek (page load). Ha egy lapot letöltünk, akkor az bizonyos eseményeket idéz elő. Ezek az események időrendben a következők: Init, Load, PreRender, Unload.

Az Init esemény az első. Ebben az eseményben az ASP.NET inicializálja a lapot első letöltés esetén. Ismételt letöltés esetén pedig visszaállítja a lapot, ekkor történik meg az úgynevezett lapállapot visszaállítása, amiről később még szó lesz. Az esemény kezelője a Page_Init eljárás.

A következő esemény a Load esemény. Ezen esemény alatt a lap létrehozza és inicializálja, illetve visszaállítja a lapban definiált vezérlőket. A Load esemény kezelője a Page_Load eljárás. Leggyakrabban arra használjuk, hogy megadjuk vele a lap megjelenésére szükséges utasításokat. A Load esemény minden vezérlő esemény előtt fut le. A Load eseményben szoktuk használni a Page osztály IsPostBack tuljdonságát. Az ASP.NET ezzel a

tulajdonsággal tartja számon, hogy a lap először, ekkor az IsPostBack értéke false, esetleg ismételten lett letöltve, ekkor az IsPostBack értéke true.

A PreRender esemény akkor fut le, ha az összes vezérlő eseménye lezajlott. Ennek az eseménynek a kezelője a Page_PreRender eljárás. Ritkán használt lapesemény.

Az Unload esemény akkor következik be, miután a lap megjelent. Ezt az eseményt arra szokás használni, hogy a lap megjelenésével feleslegessé vált erőforrásokat felszabadítsuk. Ilyen erőforrás lehet például egy adatbázis-kapcsolat. Az Unload esemény kezelője a Page_Unload. Ha lehívjuk a lapunkat a böngészőben, akkor a Page_Unload eljárásban megadott szöveg nem jelenik meg, mivel az esemény a lap megjelenése után zajlik le.

Mind a négy lapesemény kezelője ugyanazzal a két paraméterrel van ellátva: az object sender objektum az eseményt előidéző objektum, ez ebben az esetben a Page objektum. A másik paraméter az EventArgs e objektum az eseménnyel kapcsolatos jellemzőket továbbító objektum. Ezekről bővebben lesz még szó.

Minden ASP.NET vezérlő eseményekkel rendelkezik, ezeket nevezzük vezérlőeseményeknek. Ezek segítségével adatokat tudunk küldeni a szervernek, azután újból le tudjuk tölteni a lapot. A lap újbóli letöltésekor a Load esemény kezelőjében ezeket a szerverre küldött adatokat fel tudjuk használni a lap újbóli megjelenítésénél. A lap és vezérlőesemények együttes működése képzi az ASP.NET alkalmazások dinamikus mivoltának alapját.

Minden eseménykezelőnek vannak argumentumai. Az első egy object típusú változó, ami a System.Object rövidneve. Ez a változó képviseli az eseményt előidéző vezérlőt. Például, ha több gombot helyezünk el egy lapon, akkor meg tudjuk mondani ennek a paraméternek a segítségével, hogy melyik gombra is kattintottunk. Rajta keresztül férhetünk hozzá a vezérlő tulajdonságaihoz, illetve metódusaihoz. A második paraméter egy System. EventArgs típusú változó, vagy ennek leszármazottja. Ez az objektum az előidézett eseményről tárol információkat. Egyes esetekben nem hordoz semmilyen lényeges adatot, más esetben pedig igen fontos információkat tartalmaz.

7. A nézetállapot

Valahányszor előidézünk egy eseményt, vagyis visszaküldjük a lapot a szerverre, akkor adatokat küldünk a lap állapotáról. Ezt nevezzük visszaküldésnek (post back). Az előidézett

esemény újra lekéri a lapot a szerverről. A szerver a visszaküldés során kapott információt feldolgozott formában visszaküldi a böngészőnek. A visszakapott információból a böngésző fel tudja építeni a lapot, úgy, hogy a különböző vezérlők állapotát megtartja. Mivel a szervertől visszakapott kód tiszta HTML kód, ezért a nézetállapot megőrzése csakis a szerveren történhet. De ha ez így van, akkor a szervernek rengeteget kellene dolgoznia, és igen sok memóriára lenne szüksége.

A valóságban úgy működik ez a nézetállapot megtartása, hogy ha egy lapot lekérünk, akkor a szerver elküldi a HTML kódját a böngészőnek. A lap eredeti állapotáról úgynevezett pillanatfelvételt készít, és a lapot reprezentáló ViewState tulajdonságába tárolja. A ViewState tulajdonság a Control osztály tulajdonsága, mivel a Page osztály - ami az ASP.NET lapot reprezentálja - ezen Control osztály leszármazottja, örökli ezt a tulajdonságot. A ViewState tulajdonság típusa System.Web.UI.StateBag, ami egy szótár, amely név6érték párokként tárolja az információt. Tehát a lap összes vezérlőjének az állapota bekerül a Page objektum ViewState tulajdonságába. Ezután az ASP.NET bizonyos algoritmusok segítségével a ViewState értéket egy karakterláncca alakítja, amit egy <input type=”hidden” name=”__VIEWSTATE”/> HTML elem value attribútumához hozzárendel. Ez a rejtett HTML elem belekerül a böngészőhöz küldött HTML kódba. A ViewState állapotát megőrző memóriaszeletet az ASP.NET felszabadíthatja. Tehát a lap nézetállapota el lett rejtve a visszaküldött HTML kódba. Ha a lekért lapnak megváltoztatjuk a nézetállapotát (például kitöltünk egy szövegmezőt), majd visszaküldjük a szerverre alapot (például egy gomb megnyomásával), akkor nemcsak a vezérlők állapota, hanem az elrejtett __VIEWSTATE nevű HTML elem is felkerül a szerverre. A szerver a __VIEWSTATE segítségével össze tudja hasonlítani a lap korábbi nézetállapotot az új nézetállapottal. Ezek után előállítja az új lap HTML kódját, kiegészítve a lap nézetállapotát leíró karakterjánccal.

Ezzel a módszerrel lehet a nézetállapotot megőrizni az ismételt letöltések között, úgy, hogy a szerver teljesítménye jelentősen csökkenne.

Alapértelmezésben a nézetállapot megőrzése be van kapcsolva, de ezt program szinten kikapcsolhatjuk a @ Page direktíva EnableViewState attribútumával.

8. Az ASPX lapok struktúrája

Ebben a fejezetben az aspx lapok struktúrája kerül bemutatásra. Egy ASP.NET lap az alábbi elemekből áll:

- Lapdirektívák, vagy direktívák (page directive)
- Szerveroldali beillesztő direktívák (server-side include directive)
- Impotáló direktívák (import direktive)
- Szerveroldali megjegyzések (server-side comment)
- Deklarációs kódtömbök (code declaration block)
- Megjelenítési kódtömbök (code render blok)
- Egyszerű szöveg
- HTML elemek (HTML tag)
- HTML vezérlők (HTML control)
- Web vezérlők (web control)

A lapdirektívákkal meghatározhatjuk az ASP.NET lapok lefordításának módját. A direktívákat a kódban a <%@ és a %> elemek között helyezzük el. Egy direktívát bárhol elhelyezhetünk a kódban, de általában a kód elején szokott szerepelni. A lapdirektívák szintakszisa a következő:

<%@ direktíva attribútum=”érték” [attribútum=”érték”…] %>

Az ASP.NET számos direktívát ismer.

A @ Page lapdirektíva lap-specifikus attribútumokat definiál, melyek az ASP.NET értelmező és fordító működését határozzák meg. Elég sok attribútuma van, ezek közül a legfontosabb az Inherits, Codebehind, Src, Language, Trace és az EnableWievState. Az Inherits attribútum az ASP.NET lapot reprezentáló osztályt határozza meg. A Codebehind pedig ezen osztályt definiáló állományt adja. Ez a két attribútum szükséges ahhoz, hogy a Visual Studio.NET a .aspx és a .aspx.cs állományból álló ASP.NET alkalmazást lefordítson. Ekkor a webszerverre való telepítésnél a .aspx állományt és a bin mappában lévő lefordított állományt kell telepíteni. Ha a webszerverre a .aspx és a .aspx.cs állományokat telepítjük, akkor ez a két állomány a szerveren lesz lefordítva, illetve értelmezve. Azt, hogy ez a két állomány összetartozik a @ Page direktíva Src attribútumával tehetjük meg. A Language

attribútummal már találkoztunk. A lap programozási nyelvét határozza meg. Alapértelmezésben ez a nyelv a Visual Basic.NET. A @ Page direktíva egy hasznos attribútuma a Trace. Ennek értéke true vagy false. Alapértelmezés szerint false. Ha az attribútum értéke true, akkor követhetjük a lap megjelenítését, úgy, hogy a megjelenés után az ASP.NET hozzáad egy táblázatot a kódhoz, mely fontos információkat tartalmaz a végrehajtott kódról. A táblázatban nyomkövetési üzeneteket is megjeleníthetünk, a Trace objektum Write és Warn metódusai segítségével. Minkét metódus egyetlen paramétere a megjelenítendő karakterlánc. A különbség csak annyi, hogy a Warn pirosan, míg a Write feketén jeleníti meg a paraméterként átadott szöveget. At EnableViewState attribútmmal is találkoztuk már. Ennek segítségével tudjuk program szinten ellenőrizni a nézetállapot működését. Ha az attribútum értéke true, akkor bekapcsoljuk, ha pedig false, akkor kikapcsoljuk a nézetállapot megőrzését. A @ Page direktíva a leggyakrabban használt direktíva.

A beillesztő direktívák lehetővé teszik, hogy egy állományban elhelyezzük azokat a kódrészeket, amelyeket többször, esetleg más alkalmazásokban is fel szeretnénk használni. Mindegy, hogy vegyes kódot vagy mögöttes kódot tartalmazó állományokkal dolgozunk. Kétféle beillesztő direktíva létezik. Az egyik relatív elérési út megadásával adja meg a beillesztendő állományt. Ennek a formális alakja a következő:

<!-- #include file = „relatív elérési út” -->

A beillesztendő állomány elérési útját a beillesztő állományhelyéhez viszonyítva kell megadnunk. A másik beillesztő direktíva a beillesztendő állomány virtuális elérési útját használja:

<!-- #include virtual = „virtuális elérési út” -->

A virtuális elérési út lehet abszolút vagy relatív. Felhasználhatjuk az egy mappával feljebb ../ jelzést, akár többszörösen is. A beillesztett állományok neve és kiterjesztése tetszőleges lehet. Az állomány arra a helyre lesz beillesztve, ahol a beillesztő direktíva van. A beillesztés a kód értelmezése vagy végrehajtása előtt történik, ezért a beillesztő direktíva file vagy virtual attribútuma nem tartalmazhat semmilyen kódrészletet.

Az importáló direktívák a névterek importálásánál nyújtanak segítséget. A .NET keretrendszerben az osztályok névterekbe vannak csoportosítva. Ha egy osztályt a névtere magadása nélkül adnánk meg, akkor hibát kapnánk, mert az ASP.NET lapokban nem minden

osztályt használhatunk a névtere megadása nélkül. Bizonyos névtereket nem kell importálni, mert azokat az ASP.NET automatikusan importálja. Ha azonban a névteret minden alkalommal teljes egészében meg kellene adni, akkor egy áttekinthetetlen kódot kapnánk eredményül. Egy importáló direktíva csak egy névteret importálhat. Ha több névteret szeretnénk importálni, akkor minden névtérre alkalmazni kell az importáló direktívát. Az importáló direktíva általános alakja a következő:

<%@ Import NameSpace=”névtér” %>

A következő névtereket importálja automatikusan az ASP.NET:

- System
- System.Collections
- System.Collections.Specialized
- System.Configuration
- System.IO
- System.Text
- System.Text.RegularExpressions
- System.Web
- System.Web.Caching
- System.Security
- System.SessionState
- System.Web.UI
- System.Web.UI.HtmlControls
- System.Web.UI.WebControls

A szerveroldali megjegyzések általában dokumentációs célokat szolgál. A megjegyzésnek szánt szöveget vagy kódrész a <%-- és --%> elválasztó karakterek közé kell írnunk. Egy megjegyzés több sort is elfoglalhat. Dokumentációs funkcióján túl a szerveroldali megjegyzéseket hibakeresési célból is lehet használni. Ekkor elrejtjük a szerver elől azokat a kódrészeket, melyek a hibát okozhatják, vagy amelyek a hibakeresésnél nem jelentősek. A szerveroldali megjegyzéseket a szerver nem küldi el a böngészőhöz.

Deklarációs kódtömbnek nevezzük az ASP.NET állománynak azt a részét, ahova az alkalmazás logikáját deklaráló kódrészletet írjuk. A deklarációs kódtömböt elvileg bárhova írhatjuk az ASP.NET állományon belül és akár több ilyen tömböt is használhatunk. Ha több deklarációs kódtömböt használnánk, akkor az a forráskód megértésének a rovására menne, ezért célszerű csak egy kódtömböt használni, és azt az állomány egy jól elkülönített részébe írjuk. Ilyen rész lehet például a <body> HTML elem előtti rész vagy az állomány vége. A deklarációs kódtömböt a <script runat=”server”> és a </script> elemek közé kell zárni. A <script> elem runat attribútumát mindig meg kell adni, és értéke csakis „server” lehet. Ezen túl még két opcionális attribútummal rendelkezik: a language és az src. A language attribútummal a deklarációs kódtömb nyelvét adjuk meg. Ha ezt az attribútumot elhagyjuk, akkor a @ Page direktíva által meghatározott nyelv lesz az alapértelmezett. A language attribútum értékének meg kell egyeznie a @ Page direktíva által meghatározott nyelvvel, ellenkező esetben értelmezési hibát idézünk elő. Az src attribútum segítségével tudunk elérni egy másik állományban lévő kódot. Egy ASP.NET lapban lévő deklarációs kódra szükség lehet más lapoknak is. Ekkor a kódot egy külön állományba célszerű elhelyezni. Ha szükségünk van ebben az állományban lévő kódra, akkor a src attribútumnak kell értékül adni az adott állomány nevét. A deklarációs kódtömb csak deklarációkat tartalmazzon. Ne tartalmazzon olyan utasításokat, melyek azonnali végrehajtást igényelnek. A deklarációs kódtömbben csak deklaráljuk a kódot, ami később kerül végrehajtásra. Másképpen kifejezve, a deklarációs tömbök csak osztályok, változók, és eljárások deklarációját tartalmazhatja. Az itt deklarált elemek később lesznek felhasználva. A deklarációs kódtömböket egyszerűen kódtömbnek, vagy programkódnak is szokás nevezni.

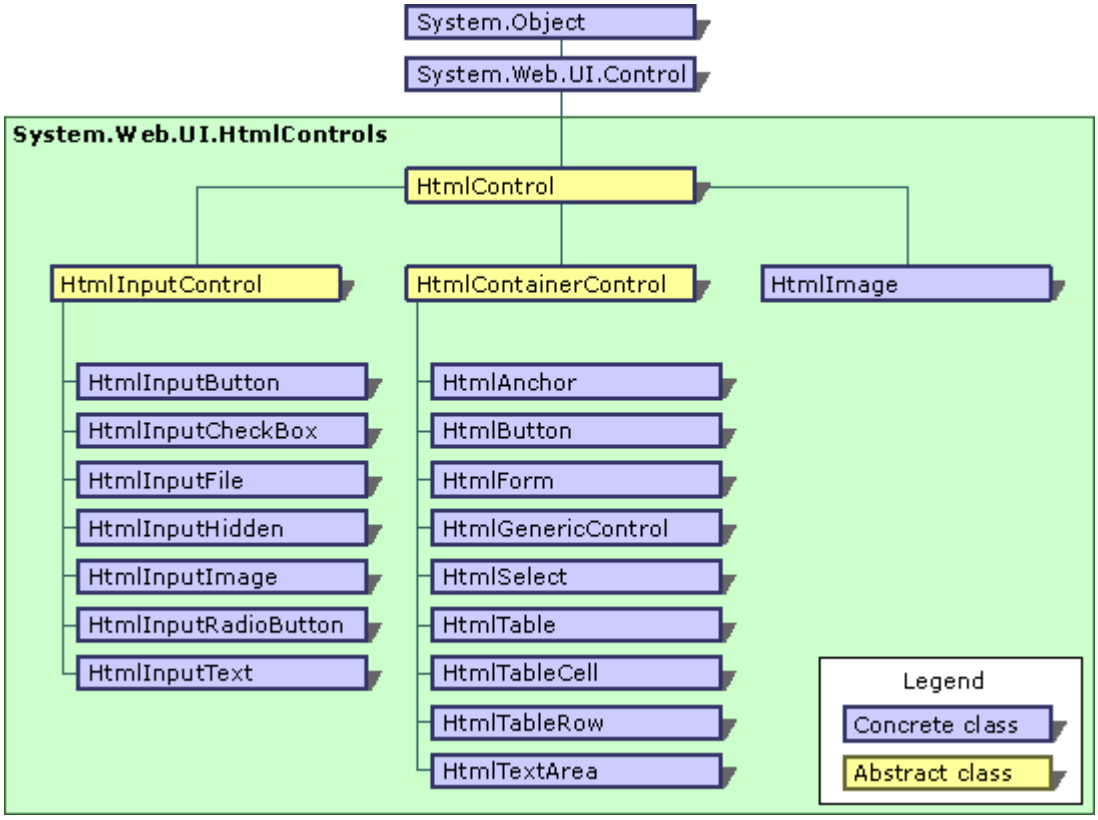
A deklarációs kódtömbök mellett szükségünk van olyan kódrészletekre is, melyek nem deklaratív, hanem végrehajtandó utasításokat tartalmaz. Ezeket a kódrészleteket megjelenítő kódtömbökbe kell írnunk. A megjelenítő kódtömbökbe végrehajtható kódot írhatunk. Kétféle megjelenítő kódtömböt különböztetünk meg: az inline, vagyis sorbeli kódot (inline code) illetve az inline kifejezést (inline expression). Az inline kódot a <% és %> elválasztó karakterek közé kell írnunk. Az inline kód egy vagy több utasítást hajt végre. Ezeket az utasításokat a @ Page direktííva által meghatározott nyelven kell írni. Az inline kifejezést a <%= és %> elválasztó karakterek közé kell írni. Az inline kifejezés egy változó vagy függvény értékét jeleníti meg.

9. A vezérlőelemek

Az ASP.NET kétféle szerveroldali vezérlőelemet ismer: HTML vezérlők és web vezérlők. A HTML vezérlők megfelelnek az ismert HTML 4.0 elemeknek. Általában azt mondhatjuk, hogy bármely hagyományos HTML elemet átalakíthatunk egy megfelelő HTML vezérlővé úgy, hogy az attribútumai közé felvesszük a runat = ”server” attribútumot. Ezt az attribútumot és esetleg más, csak a HTML vezérlők szintaxisától elfogadott attribútumot, a böngésző nem ismeri. Ezért mielőtt a szerver a lapot elküldi a lapot kérő böngészőhöz, lefordítja a kódjukat HTML 4.0 kódra.

9.1. A HTML vezérlők

Minden HTML vezérlő egy deklaratív definícióval vagy deklarációval és egy osztálydefinícióval rendelkezik. A delarációt nyitó részt a < és > zárójelek közé kell tenni. Első helyen az elemnév áll, ezt követik az attribútumok, attribútum = ”érték” alakban. Minden vezérlőnek tartalmaznia kell természetesen a runat = ”server” attribútumot.



A HTML vezérlők hierarchiája

Az ASP.NET HTML vezérlők a System.Web.UI.HtmlControls névtérhez tartoznak. Ez a névtér tartalmazza a HtmlControl osztályt, melytől az összes HTML vezérlő származik, illetve még két osztályt a HtmlTableCellCollection és a HtmlTableRowCollection. A HtmlControl osztály a System.Web.UI névtér Control osztályától származik, amely a System.Object osztálytól származik. A HtmlControl osztálynak három leszármazottja van, ezek a HtmlContainerControl, a HtmlImage és a HtmlInputControl osztályok.

A HtmlControl osztály az összes HTML vezérlők alaptulajdonságait biztosítja. Tulajdonságait, metódusait és eseményeit az összes HTML vezérlő örökli. Az osztály absztrakt, tehát az osztálynak nem felel meg egy HTML elem sem. Az osztály legfontosabb tulajdonságai:

- Attributes: visszaadja a HTML vezérlő attribútumait tároló gyűjteményt. A gyűjtemény név/érték párokban tárolja az attribútumokat.
- Disabled: Meghatározza, hogy a vezérlő aktív-e. Értéke true, ha aktív, ez az alapértelmezett is, false, ha nem aktív.
- Style: Visszaadja a HTML vezérlő CSS (cascading style sheet) stílusát tároló gyűjteményt. A gyűjtemény a stíluselemeket név/érték párokban tárolja.
- TagName: A HTML elem neve, amely a < karakter után álló szó.

9.1.1. A konténervezérlők

A HTML konténervezérlői nagyon hasonlítanak a HTML elemekhez, de sok új tulajdonsággal rendelkeznek, melyek segítségével a megjelenésüket és a viselkedésüket programszinten ellenőrizhetjük. Konténervezérlők a nyitó rész mellett tartalmaznak záró részt is, ennek alakja: </HTMLvezérlő>. A konténervezérlőknél a nyitó és záró részek között helyezkedik el a tartalom. A HtmlContainerControl osztály mindazoknak a HTML vezérlőknek a tulajdonságait definiálja, melyeknek egy záró elemük van, mint például a . Ez az osztály is absztrakt. Az osztály két fontos tulajdonsággal rendelkezik:

- InnerHtml: meghatározza a vezérlő tartalmát, figyelembe véve a HTML formázást. Nem kódolja a HTML specifikus karaktereket.

- InnerText: Meghatározza a vezérlő tartalmát HTML formázás nélkül. A HTML specifikus karaktereket specifikusan kódolja.

A HtmlContainerControl osztályból számos HTML vezérlő származik:

- HtmlAnchor
- HtmlButton
- HtmlForm
- HtmlGenericControl
- HtmlSelect
- HtmlTable
- HtmlTableCell
- HtmlTableRow,
- HtmlTextArea.

A HtmlAnchor segítségével program szinten ellenőrizhetjük a HTML <a> elemet. A vezérlő nyitó és záró része között egy tetszőleges szöveg lehet. Az osztály legfontosabb tulajdonságai:

- HRef: meghatározza a kapcsolat URL-jét.
- Name: meghatározza a könyvjelző neevét.
- Target: meghatározza a célablakot. A Target attribútum értékei a következők lehetnek: _blank, ekkor a célablak egy új ablak keret nélkül, _parent, ekkor a célablak ugyanaz az ablak szülőkerettel, _self, ekkor a célablak ugyanaz az ablak, ugyanazzal a kerettel, _top, ekkor a célablak ugyanaz az ablak, de keret nélkül. Bármely más érték megegyezik a _blank értékkel.
- Title: meghatározza a gyorstippet, és a böngésző által használt címet.

Az osztályhoz tartozik egy esemény, a ServerClick esemény, amely akkor lesz kiváltva, amikor a vezérlőre kattintunk. Ezen esemény kezelője az OnServerClick metódus. A vezérlő nagy előnye, hogy képes postback (lap visszaküldése a szerverre) eseményekre reagálni.

A HtmlButton vezérlő a <button> HTML elemnek felel meg. A vezérlő nyitó és záró része közötti tartalom egy szöveg, egy képállomány URL-je, vagy egy másik vezérlő lehet. Különböző stílusdefinícióval a vezérlő megjelenítését megváltoztathatjuk.

A HtmlForm vezérlő a <form> HTML elemnek felel meg. Minden vezérlőt, amelyik a szerver postback szolgáltatását igénybe akarja venni, be kell ágyazni egy HTMLForm vezérlőbe. Egy ASP.NET lap maximum egy HtmlForm elemet tartalmazhat.

A HtmlGenericControl vezérlő mindazokat a HTML elemeket reprezentálja, melyeket nem képvisel egyéni HTML vezérlő. Ilyen HTML elem a <body>, a , vagy a . Ha a az osztály konstruktorát paraméter nélkül hívjuk meg, akkor az egy HTML elemet hoz létre. A konstruktor paramétereként megadható egy sztring, így a meghatározhatjuk a létrehozandó HTML elemet.

A HtmlSelect vezérlő a <select> HTML elemnek felel meg. Ez a vezérlő támogatja az adatkötést. Ezáltal a vezérlő opcióit adatforrásból is fel lehet tölteni. A vezérlő eseménye a ServerSelect, mely akkor lesz előidézve, amikor a kliens kiválaszt egy opciót az opciógyűjteményből. A ServerSelect nem vált ki postback-et, ezért, hogy a ServerSelect végre legyen hajtva egy másik vezérlővel, például HtmlButton-al, kell előidézni a postback eseményt.

A HtmlTable vezérlő a <table> HTML elemet képviseli. A <table> HTML elem egy összetett elem, mely több alárendelt HTML elemből tevődik össze. Ennek megfelelően, programozási szinten is több HTML vezérlő és osztály segítségével tudjuk ellenőrzésünk alá vonni a HtmlTable vezérlő összes összetevőjét. Ezek a HTML vezérlők a HtmlTableRow, a HtmlTableRowCollection, a HtmlTableCell, és a HtmlTableCellCollection. A HtmlTable vezérlő által reprezentált tábla sorait a vezérlő Rows tulajdonsága adja vissza. A Rows tulajdonság csak olvasható és értéke egy HtmlTableRowsCollection típusú gyűjtemény. A gyűjtemény elemei HtmlTableRows vezérlők. Ez a vezérlő képviseli a <tr> HTML elemet. Egy HtmlTableRow vezérlő által reprezentált sor celláit a vezérlő Cells tulajdonsága adja vissza. Ez a tulajdonság szintén csak olvasható, értéke pedig egy HtmlTableCellCollection típusú gyűjtemény. A gyűjtemény elemi HtmlTableCell vezérlők. Ez a vezérlő képviseli a <td> HTML elemet.

A HtmlTextArea a <textarea> HTML elemet képviseli. A vezérlő nyitó és záró része között egy tetszőleges tartalmat helyezhetünk el. A vezérlő eseménye a ServerChange, amely akkor lesz előidézve, amikor a kliens megváltoztatja a textarea által meghatározott tartalom tartalmát. A HtmlSelect vezérlőhöz hasonlóan a HtmlTextArea sem vált ki postback eseményt,

így a ServerChange eseménykezelő végrehajtásához egy másik vezérlővel kell a postback-et előidézni.

9.1.2. A beviteli vezérlők

A HtmlInputConrol osztály a beviteli mezők alaposztálya. Ez az osztály a HtmlContainerControl osztályhoz hasonlóan szintén absztrakt. A legfontosabb tulajdonságai:

- Name: meghatározza az osztály azonosító nevét.
- Type: Visszaadja a beviteli vezérlő típusát, melynek értéke a következők valamelyike lehet: text, password, checkbox, radio, button, submit, reset, file, hidden, image.
- Value: meghatározza a hozzárendelt értéket, amely a HTMLInputControl osztálytól származtatott vezérlőtől függ.

A HtmlInputControl osztályból származó HTML vezérlők a következők:

- HtmlInputButton
- HtmlInputCheckBox
- HtmlInputFile
- HtmlInputHidden
- HtmlInputImage
- HtmlInputRadioButton
- HtmlInputText

A HtmlInputButton vezérlő segítségével programból érhetjük el az <input type = ”button”>, az <input type = ”submit”> illetve az <input type = ”reset”> HTML elemeket. A HtmlInputButton előidézhet egy postback eseményt. A vezérlő eseménye a ServerClick esemény, melynek a kezelőjét az OnServerClick attribútummal adhatjuk meg. A type = ”button” és a type = ”submit” típusok esetben, ha rákattintunk a vezérlőre, akkor a vezérlő visszaküldi a <form> HTML elem tartalmát a szerverre. A szerverre küldött adatokat eseménykezelő eljárásokban feldolgozhatjuk. Az adatok feldolgozása a Page_Load eljárásban is megtörténhet. Ha a HtmlInputButton vezérlő típusa type = ”reset”, akkor a vezérlőre kattintva a vezérlő visszaállítja a <form> tartalmát az eredeti állapotára. Nem váltódik ki a ServerClick esemény, és a lap sem lesz visszaküldve a szerverre.

A HtmlInputCheckBox vezérlő az <input type = ”checkbox”> HTML elemnek felel meg. Ez nem más, mint a klasszikus jelölőnégyzet. A vezérlőnek van egy opcionális attribútuma a checked. Ha megadjuk, akkor a jelölőnégyzet ki lesz jelölve, ha nem adjuk meg, akkor nem. A megváltoztatjuk a vezérlő állapotát, akkor előidézzük a vezérlő eseményét, a ServerChange-et. A vezérlő állapotát a jelölőnégyzet bejelölésével illetve a bejelölés eltávolításával változtathatjuk meg. A ServerChange esemény kezelőjét az onserverchange attribútummal határozhatjuk meg. A vezérlő eseményének előidézése nem jár együtt a postback-el. Ekkor a postback-et egy HtmlButton, vagy egy HtmlInputButton segítségével idézhetünk elő. A lap ismételt letöltése alkalmával végrehajtódik a ServerChange esemény kezelője. A vezérlő állapotát a Checked tulajdonsággal ellenőrizhetjük, melynek értéke true ha be van jelölve, illetve false ha nincs.

A HtmlIputFile vezérlő segítségével állományokat tölthetünk fel a szerverre. Ezzel a vezérlővel az <input type = ”file”> HTML elemet ellenőrizhetjük vele programozási szinten. A HtmlInputFile vezérlő MIME (Multipurpose Internet Mail Extensions) attribútumával meghatározhatjuk az elfogadott MIME típusokat. Több MIME típus esetén a különböző típusokat vesszővel kell elválasztani. A vezérlő legfontosabb tulajdonsága a PostedFile tulajdonság, amely visszaadja a felküldött állományt képviselő HttpPostedFile típusú objektumot. A felküldött állományt ezen az objektum segítségével érhetjük el és dolgozhatjuk fel a szerver oldalon. A PostedFile tulajdonság csak olvasható. A HttpPostFile osztálynak négy attribútuma és egy metódusa van. Az attribútumok a ContentLength, amely visszaadja a feltöltött állomány méretét bájtban, a ContentType, feltöltött állomány MIME típusát adja vissza, a FileName visszaadja az állomány abszolút elérési útját a kliens gépen, és az InputStream visszaad egy Stream objektumot, amely segítségével olvasni lehet az állományt még a feltöltés előtt. Az egyetlen metódusa a HttpPostFile-nak a SaveAs, mely segítségével elmenthetjük az állományt a szerverre. A HttpInputFile nem idéz elő postback-et, ennek érdekében HtmlButton-t, vagy HtmlInputButtont használhatunk. Ahhoz, hogy a HtmlInputFile vezérlő a postback-el együtt ténylegesen elküldje a kiválasztott fájlt, be kell ágyazzuk egy HtmlForm vezérlőbe, melynek enctype attribútuma ”multipart/form-data”.

A HtmlInputImage vezérlő nagyon hasonlít a HtmlInputButton vezérlőre, a különbség az, hogy a HtmlInputImage egy gomb helyett egy képet jelenít meg. Ez a vezérlő az <input type = ”image”> HTML elemet képviseli. A vezérlő esemény kezelője a ServerClick esemény. Ennek

az a sajátossága, hogy második paramétere nem EventArgs típusú, hanem ImageClickEventArgs. Ennek a típusnak az x és y attribútumai visszaadják annak a képpontnak a koordinátáit, ahova a felhasználó a képre kattintott.

A HtmlInputRadioButton vezérlő a klasszikus <input type = ”radio”> választógombhoz biztosít hozzáférést a szerverkódban. Ezeket a típusú vezérlőket rendszerint csoportosítani szokták, ezt a name attribútummal lehet megtenni. Azok a HtmlInputRadioButton vezérlők, amelyeknek name attribútuma ugyanaz, ugyanahhoz a csoporthoz tartoznak. Ebben az esetben a csoportnak maximum egy választógombja lehet kiválasztott állapotban. Ha egy választógombnak nem adjuk meg a name attribútumát, akkor az egymagában képez egy csoportot. Ha megváltoztatjuk egy csoportban az egyik választógomb attribútumát, akkor kiváltjuk a vezérlő ServerChange eseményét. Ezt az onserverchange attribútummal határozhatjuk meg. A ServerChange esemény itt sem vált ki postback-et. Egy HtmlInputRadioButton vezérlőnek az opcionális checked attribútumával határozhatjuk meg, hogy ki legyen-e választva. A lap ismételt letöltésekor végrehajtódik a ServerChange esemény kezelője. A vezérlő állapotát a Checked tulajdonsággal ellenőrizhetjük, amelynek értéke true, ha a választógomb ki van jelölve, illetve false, ha nincs. Egy csoporton belül több választógombnak is megadhatjuk a checked attribútumot, de csak az a választógomb lesz kiválasztva, amelyiknek utoljára adtuk meg a checked attribútumot.

A HtmlInputText vezérlő a leggyakrabban használt HTML vezérlők egyike. A vezérlő segítségével az <input type = ”text”> illetve az <input type = ”password”> HTML elemeket ellenőrizhetjük programozási szinten. A vezérlő type attribútuma ”text”, vagy ”password” lehet. Így határozhatjuk meg, hogy a vezérlő mely HTML elemet képviseli. Ha az attribútum értéke ”password”, akkor a beadott szöveg a vezérlőben pontokkal, vagy csillagokkal lesz helyettesítve. A vezérlő maxlength attribútumával meghatározhatjuk a bevihető szöveg maximális hosszát karakterekben. A vezérlő nem vált ki postbak-et, ezt a szokásos módon egy HtmlButton, vagy HtmlInputButton, esetleg HtmlInputImage segítségével lehet megtenni.

9.1.3. Az ImageVezérlő

A harmadik HtmlControl osztályból származó osztály a HtmlImage. Segítségével a klasszikus <img> HTML elemhez kapunk kódbeli hozzáférést. Deklarációjában megadhatjuk azokat az attribútumokat, amelyeket az <img> HTML elem esetében is használhatunk: id, alt,

border, height, src, width. A képek megjelenítésénél a HtmlImage vezérlő ugyanúgy működik, mint az <img> HTML elem. Előnye az utóbbival szemben az, hogy a programkódból nemcsak a megjelenítendő kép tulajdonságait (például méretét), hanem a képforrást is megváltoztathatjuk.

9.2. A Web vezérlők

A HTML vezérlők mellett a web vezérlők képezik azt az új eszköztárat, amivel az ASP.NET új lehetőségeket nyit a dinamikus weblapok programozásában. A web vezérlők nem felelnek meg feltétlenül HTML elemeknek, habár nagyon sok közülük az ismert HTML elemeket bővítik vagy általánosítják. Léteznek azonban olyan web vezérlők, melyek nem felelnek meg HTML elemeknek és egészen újszerű lehetőségeket biztosítanak a dinamikus ASP.NET lapok számára. A böngésző - csakúgy, mint a HTML vezérlőket - a web vezérlőket sem ismeri, tehát mielőtt a szerver egy lapot elküld a lapot kérő böngészőhöz, lefordítja kódjukat HTML 4.0 kódra. A web vezérlők majdnem mindegyik HTML vezérlő szerepét átvehetik, ezen felül okosabbak is. Lényegében a web vezérlők a nehézsúlyú vezérlők, melyek csak akkor használatosak, ha igazán szükség van a nagyobb tudásra.

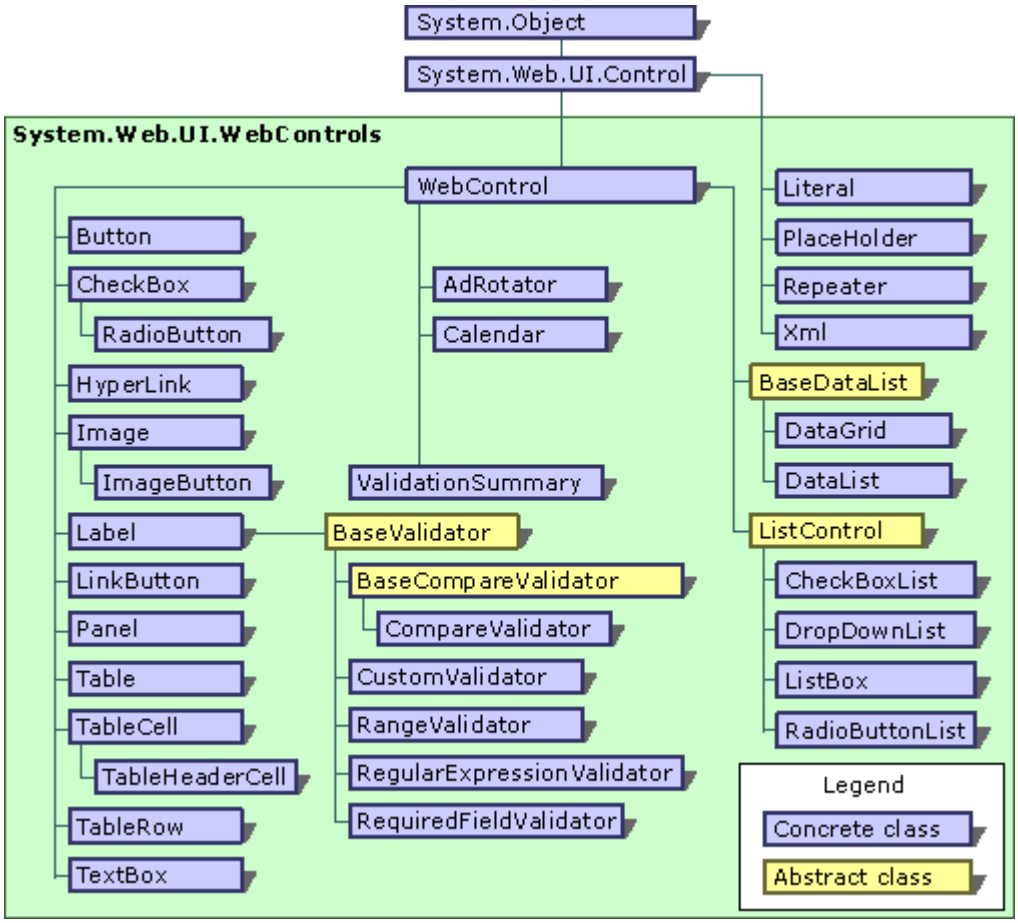
Az ASP.NET web vezérlői a System.Web.UI.WebControls névtérhez tartoznak. Ez a névtér tartalmazza a WebControl osztályt, amelytől majdnem az összes web vezérlő származik. A vezérlők vizuális megjelenését meghatározó tulajdonságokat a vezérlők a WebControl osztálytól öröklik. Az osztály által biztosított tulajdonságokat, a vezérlők alaptulajdonságainak (base property) nevezzük. Ezek a tulajdonságok a következők:

- BackColor: meghatározza a háttérszínt. Értéke egy ismert szín neve lehet, vagy egy RGB érték.
- BorderStyle: meghatározza a keret stílusát.
- BorderWidth: meghatározza a keret szélességét pixelben kifejezve.
- Font-Bold: meghatározza, hogy a szöveg félkövér legyen-e. Értéke true vagy false lehet.
- Font-Italic: meghatározza, hogy a szöveg félkövér legyen-e.
- Font-Name: meghatározza a használt karakterkészlet nevét.

- Font-Names: meghatároz egy karakterkészlet családot. Ha a kliensgépen a család első karakterkészlete nincs telepítve, akkor a második lesz alkalmazva, ha az sincs, akkor a következő és így tovább.
- Font-Overline: meghatározza, hogy a szöveg felett legyen-e vonal.
- Font-Size: meghatározza a szöveg méretét, pontban vagy pixelben kifejezve.
- Font-Strickeout: meghatározza, hogy a szöveg át legyen-e húzva.
- Font-Underline: meghatározza, hogy a szöveg alá legyen-e húzva.
- ForeColor: meghatározza a szöveg színét.
- Height: meghatározza a vezérlő magasságát pixelben kifejezve.
- TabIndex: meghatározza a tabulátor sorrendet
- ToolTip: meghatározza a gyorstippet.
- Width: meghatározza a vezérlő szélességét pixelben kifejezve.

Természetesen ezeket a tulajdonságokat programkódból is el lehet érni. Ekkor a hozzájuk rendelt, és a visszaadott értékek a tulajdonságnak megfelelő objektumok.

A web vezérlők ezen tulajdonságain túl még két általános tulajdonság van. Ezek a Visible és az Enabled tulajdonságok. A Visible nem felel meg egyetlen HTML attribútumnak sem. Ha értéke false, akkor az ASP.NET nem írja a visszaküldött lap HTML kódjába a vezérlőnek megfelelő HTML kódot, vagyis a vezérlő nem jelenik meg a lapon. Az Enable a disabled HTML attribútumnak felel meg. Ha egy adott vezérlő Enable attribútumának értéke false, akkor a vezérlőt megjelenítő HTML kódot az ASP.NET a disabled attribútummal látja el.



A web vezérlők hierarchiája

9.2.1. Az alapvezérlők

A WebControl osztálytól származnak közvetve vagy közvetlenül az alapvezérlők:

- Button
- CheckBox
- RadioBox
- HyperLink
- Image
- ImageButton
- Label
- LinkButton
- Panel
- Table

- TableCell
- TableHeaderCell
- TableRow
- TextBox

Az alapvezérlőkhöz tartoznak még a Literal és a PlaceHolder, amelyek nem a WebControl-tól származnak, hanem a System.WebUI.Control osztálytól.

A Button ImageButton és LinkButton vezérlőket nvezzük együttesen gombvezérlőknek. A Button vezérlő az egyik legfontosabb és egyben legegyszerűbb vezérlő. A kliensalkalmazás és a szerver közötti kommunikációban a kliens a Button vezérlő segítségével jelzi, hogy befejezte a mondanivalóját, és kész a válasz fogadására. A vezérlő megfelel az <input type= ”submit”> HTML vezérlőnek. A Button vezérlő visszaküldheti a lapot a szerverre és kiválthat különböző eseményeket (event) is, ezekre a szerver reagál. Egy Button vezérlő Text tulajdonságával meghatározhatjuk a gombon megjelenő feliratot. A CommandName és a CommandArgument attribútumok segítségével a tulajdonság értékét a vezérlő a Command esemény kezelőjének továbbítja. A CausesValidation argumentum értéke true vagy false lehet, és ettől függően a szerverre küldött adatok érvényessége ellenőrzés alá kerül vagy nem. Alapértelmezett értéke true. A Button vezérlő két eseménnyel rendelkezik: a Click és a Command. Ezek az események akkor lesznek kiváltva, amikor a gombra kattintunk és a lap vissza lesz küldve a szerverre. A Command esemény esetében, ha a CommandName és/vagy CommandArgument tulajdonságok meg vannak adva, akkor értékük is felkerül a lapra. A button vezérlő két metódust tartalmaz: az OnClick és az OnCommand metódusok. Ezek a Click és a Command események kezelői.

A CheckBox vezérlő az ismert <input type= ”checkbox”> HTML vezérlőnek felel meg. A CheckBox vezérlővel egy egyszerű igen/nem értékű opciót építhetünk be a lapunkba. Több CheckBox vezérlővel egy kölcsönösen egymást nem kizáró opciókat tartalmazó csoportot valósíthatunk meg. A vezérlő Checked tulajdonságával megállapítható, hogy a jelülőnégyzet ki van-e jelölve. Értéke true vagy false lehet. Az AutoPostBack tulajdonsággal meghatározhatjuk, hogy, ha a vezérlő Checked tulajdonsága megváltozik, akkor a lap vissza legyen-e küldve a lapra vagy nem. Értéke szintén true vagy false lehet, alapértelmezett értéke false. A Text meghatározza a vezérlő szövegét, a TextAlign meghatározza a vezérlő szövegének kiigazítását. Értéke Left vagy Right lehet, alapértelmezett értéke Right. A vezérlő

eseménye a CheckedChanged, amely akkor lesz kiváltva, amikor a jelölőnégyzet állapota megváltozott. Az esemény kezelője az OnCheckedChanged.

A RadioButton vezérlő a CheckBox vezérlőtől származik. Ez a vezérlő az <input type= ”radio”> HTML vezérlőnek felel meg. Hasonlóan a CheckBox vezérlőhöz, a RadioButton-al is egy egyszerű igen/nem értékű opciót építhetünk be a lapunkba. A különbség a két vezérlő között az, hogy a RadioButton vezérlő rendelkezik a GroupName tulajdonsággal. Ezzel a tulajdonsággal csoportosíthatjuk a különböző RadioButton vezérlőket. Az egy csoporthoz tartozó RadioButton vezérlők GroupName tulajdonsága ugyanazzal az értékkel rendelkezik. Az egy csoportba tartozó RadioButton vezérlők egy kölcsönösen egymást kizáró opciócsoportot hoznak létre.

A HyperLink vezérlő segítségével egy ASP.NET. lapot egy másik weblaphoz kapcsolhatunk. HTML megfelelője az ismert <a> elem. A vezérlő nem rendelkezik metódussal vagy eseménnyel. A HiperLink vezérlő több tulajdonsággal rendelkezik. Az ImageUrl tulajdonság a megjelenítendő kép URL-je. A NavigateUrl a kapcsolt weblap URL-je. A Target tulajdonság a kapcsolat célablaka vagy célkerete (frame).

A tulajdonság lehetséges értékei: a _top, _self, _parent, _search vagy _blank.

A Text tulajdonság a kapcsolat által megjelenített szöveg. A HyperLink vezérlő nagy előnye a HTML kapcsolatban szemben az, hogy tulajdonságait programkódból is megadhatjuk és így dinamikus kapcsolatokat tudunk létrehozni.

Az Image vezérlő a klasszikus <img> HTML elemnek felel meg. Egy egyszerű vezérlőről van szó, ami egy képet jelenít meg. A vezérlőnek nincsenek metódusai és eseményei. A vezérlő fontosabb tulajdonságai közé tartozik az AlternatívText, amely meghatározza a megjelenő szöveget, ha a böngésző nem tudja megjeleníteni a képállományt; az ImageAlign, amely meghatározza a megjelenített kép kiigazítását illetve az ImageURL, amely meghatározza a képállomány Url- jét.

Az ImageButton vezérlő nagyon hasonlít a Button vezérlőre. Az ImageButton egy képállományt jelenít meg, mely rákattintásra, ugyanúgy viselkedik, mint egy parancsgomb. Az ImageButton vezérlő főbb tulajdonságai közé tartozik az AlternatívText, CommandName, CommandArgument, CausesValidation, ImageAlign, ImageURL.

Az AlternatívText meghatározza a megjelenő szöveget, ha a képállományt a böngésző nem tudja megjeleníteni. A CommandName és a CommandArgument tulajdonságértékét a vezérlő

a Command esemény kezelőjének továbbítja. A CausesValidation értéke true vagy false lehet és ettől függően a szerverre küldött anyagok érvényessége ellenőrzés alá lesz vetve vagy nem. Alapértelmezett értéke true. Az ImageAlign meghatározza a megjelenítendő kép kiigazítását. Az ImageURL meghatározza a képállomány Url-jét. A vezérlő Click és Command eseményekkel rendelkezik, amelyek akkor lesznek kiváltva, amikor a vezérlőre kattintunk. Ha a Command esemény kiváltódik és a CommandName és/vagy a CommandArgument tulajdonságok meg vannak adva, akkor értékük a lappal együtt fel lesznek küldve a szerverre. Ezeket az értékeket az eseményt kezelő OnCommand metódus figyelembe veheti. A vezérlő rendelkezik két metódussal is: az OnClick a Click esemény kezelője, míg az OnCommand a Command esemény kezelője.

A Label vezérlő segítségével HTML formázott statikus szöveget jeleníthetünk meg. A Lable vezérlő HTML megfelelője a HTML vezérlő.

A LinkButton vezérlő egy <a> HTML elemnek felel meg. A LinkButton vezérlő a gomb vezérlők csoportjához tartozik. A LinkButton vezérlő Text tulajdonsága meghatározza a szöveget melyet a vezérlő egy kapcsolattal lát el. A vezérlő CommandName és a CommandArgument tulajdonságainak az értékét a vezérlőhöz tartozó Command esemény kezelőjének továbbítja. A LinkButton CausesValidation attribútumának értéke true vagy false lehet és ettől függően a szerverre küldött adatok érvényessége ellenőrzése alá lesz vetve vagy nem. A tulajdonság alapértelmezett értéke true. A vezérlő Click és Command eseményekkel rendelkezik, amelyek akkor lesznek kiváltva, amikor a vezérlőre kattintunk. Ha a Command esemény kiváltódik és a CommandName és/vagy a CommandArgument tulajdonságok meg vannak adva, akkor értékük a lappal együtt fel lesznek küldve a szerverre. Ezeket az értékeket az eseményt kezelő OnCommand metódus figyelembe veheti. A vezérlő rendelkezik két metódussal is: az OnClick a Click esemény kezelője, míg az OnCommand a Command esemény kezelője.

A Panel vezérlő nem felel meg egyetlen hagyományos HTML elemnek sem, HTML szintű realizálása több HTML elemből áll. Ez a vezérlő egyszerre több vezérlőt képes megjeleníteni illetve elrejteni. Különösen alkalmas arra, hogy programkódból vezérlőket adhassunk hozzá a lapunkhoz. A Panel vezérlőnek nincsenek eseményei és metódusai. Három fontosabb tulajdonsággal rendelkezik, ezek a BackImageURL, a HorizontalAlign, és a Wrap. A BackImageURL meghatározza a háttérkép URL-jét, a HorizontalAlign meghatározza a

vezérlő tartalmának kiigazítását. A Wrap értéke true vagy false lehet, és ettől függően a vezérlő, ha szükséges szóvégre megtöri a sort vagy sem. Alapértelmezett értéke true. A Panel vezérlő konténerként működik. A nyitó és záró részei között elhelyezett vezérlők a Panle vezérlő tartalmát képezik. Ha a Panel vezérlő Visible tulajdonsága true, akkor a panelben tartalmazott vezérlők láthatóak lesznek, ellenkező esetben nem. A Panel vezérlő a nyitó és záró részei közé helyezett vezérlőket a lap végrehajtásakor az ASP.NET a Panel vezérlő gyermekvezérlő-gyűjteményéhez adja hozzá. Programkódból ezt a gyűjteményt a Controls tulajdonsággal érhetjük el. A tulajdonság egy System.Web.UI.ControlCollection típusú objektumot ad vissza.

A Table, a TableRow, a TableCell, és a TableHeaderCell vezérlők rendre a <table>, a <tr>, a <td>, és a <th> HTML elemeknek felelnek meg. Ezek a vezérlők szorosan összefüggnek egymással. Segítségükkel programkódból dinamikus táblákat építhetünk. A TableHeaderCell vezérlő a TableCell vezérlő leszármazottja. Ezek a vezérlők nem rendelkeznek metódusokkal vagy eseményekkel. A Table vezérlő legfontosabb tulajdonságai a BackImageURL, a CellPadding, a CellSpacing, a GrindLines, a HorizontalAlign és a Rows. A BackImageURL meghatározza a háttérkép URL, jét, a CellPadding a cellák közti távolságot határozza meg pixelben. A GrindLines tulajdonsággal tudjuk befolyásolni, hogy milyen rácsvonalak legyenek megjelenítve. Értéke lehet Horizontal, ekkor csak a vízszintes irányú rácsvonalak lesznek megjelenítve, Vertical, ekkor csak a függőleges irányúak, None, esetén egyik irányú sem, illetve Both, ekkor mindkétirányú rácsvonalak meg lesznek jelenítve. A HorizontalAlign vezérlő meghatározza a tábla vízszintes kiigazítását. A tulajdonság értékei a Center, a Justify, a Left, a Notset vagy a Right lehet. Alapértelmezett értéke a NonSet. A Rows tulajdonság visszaadja a táblasorok gyűjteményét.

A TableRow vezérlő a következő tulajdonságokkal rendelkezik: A Cells, amely visszaadja a táblasornak megfelelő gyűjteményt. A HorizontalAlign, amely meghatározza a táblasor tartalmának vízszintes kiigazítását a táblasoron belül. Értékei megegyeznek a Table vezérlőnél felsorolt értékekkel. A VericalAlign, amely meghatározza a táblasor tartalmának függőleges kiigazítását a táblasoron belül. A tulajdonság a Bottom, Middle, NotSet, illetve a Top értékeket veheti fel. Alapértelmezett értéke a NotSet.

A TableCell és a TableHeaderCell vezérlők tulajdonságai: ColumnSpan, HorizontalAlign, RowSpan, Text, VerticalAlign, és a Wrap. A HorizontalAlign, a VerticalAlign tulajdonságok

megegyeznek a fent leírtakkal. A ColumnSpan meghatározza a cella által kifeszített táblaoszlopok számát. Alapértelmezett értéke 0. A RowSpan meghatározza a cella által kifeszített táblasorok számát. Alapértelmezett értéke 0. A Text meghatározza a cella tartalmát. A Wrap attribútum értéke true vagy false lehet, és attől függően, ha szükséges szóvégre megtöri a cella tartalmának sorait. Alapértelmezett értéke true.

A TextBox, vagy szövegmező vezérlő a TextMode tulajdonság értékétől függően három különböző HTML elemnek felelhet meg. Ha a TextMode értéke MultiLine, akkor a TextBox vezérlő a <textarea> HTML elemnek felel meg, ha az értéke Password, akkor a TextBox vezérlő az <input type = ”password”> HTML elemnek felel meg, illetve ha az értéke SingleLine, akkor a TextBox vezérlő az <input type = ”text”> HTML vezérlőnek felel meg. A TextBox vezérlő ezen kívül rendelkezik még az AutoPostBack, a Columns, a MaxLength, a ReadOnly, a Rows, illetve a Wrap tulajdonságokkal. Az AutoPostBack értéke true vagy false lehet, és ettől függően, ha a vezérlő Text tulajdonsága megváltozik, automatikusan vissza lesz küldve a lap a szerverre. Alapértelmezett értéke false. A Columns tulajdonság meghatározza a vezérlő szélességét karakterben kifejezve. A MaxLength meghatározza a szövegmezőbe beírható karakterek számát. Ha a TextMode értéke MultiLine, akkor a MaxLength attribútum értéke nincs semmilyen kihatással a vezérlőre. A ReadOnly vezérlő értéke true vagy false lehet. Ha értéke true, akkor a szövegmező Text tulajdonságának értéke nem változtatható meg, ha false, akkor megváltoztatható. A Rows tulajdonságnak, csak akkor van értelme, ha a vezérlő TextMode tulajdonságának értéke MultiLine. Ekkor a Rows értéke meghatározza a vezérlő magasságát karakterben kifejezve. A Text tulajdonság meghatározza a szövegmező szövegét. A Wrap tulajdonság értéke true vagy false lehet. Ha true, akkor a szövegmező, ha szükséges a szóvégre megtöri a sort, ha false, akkor nem. A TextBox vezérlő rendelkezik egy eseménnyel, ez a TextChanged esemény, mely akkor lesz kiváltva, ha a vezérlő Text tulajdonsága megváltozik. A Text tulajdonság megvizsgálására közvetlenül a fókusz elvesztése után kerül sor. A TextChanged esemény kezelője az OnTextChanged metódus.

A Literal vezérlő egyszerűen megjeleníti a Text tulajdonság által meghatározott szöveget. A Text tulajdonsággal megadott szöveget az ASP.NET betű szerint, azaz minden változás nélkül , belefűzi a lap HTML kódjába pontosan arra a helyre, ahol a Literal vezérlő van. Ha a meghatározott szövegnek HTML értelmezése van, akkor a böngésző természetesen értelmezni fogja ezt a szöveget, és a HTML értelmezésnek megfelelően fogja megjeleníteni. Ez a

megjelenítési lehetőség sok előnnyel jár, de ugyanakkor veszélyt is hordoz magában. Literal vezérlővel egy szkriptet is befűzhetünk a lap HTML kódjába. Ha a Literal vezérlő Text tulajdonságát a lap látogatója adja meg, például egy szövegmező segítségével, akkor lapunk ki van téve bizonyos veszélyeknek.

A PlaceHolder vezérlő elsődleges célja, hogy elősegítse a vezérlők programkódból történő elhelyezését a lapon. Természetesen nincs HTML megfelelője, hanem a tartalmát képező vezérlőknek megfelelő HTML elemek adják meg. Saját tulajdonságokkal, eseményekkel vagy metódusokkal nem rendelkezik.

9.2.2. Az adatérvényesítés és az érvényesítő vezérlők

Az aktív párbeszédben, ami egy web alkalmazás végrehajtása alatt a kliens és a szerver között létrejön, a web alkalmazás a kliens által megadott adatokat feldolgozza, és a válaszát ezen adatok függvényében fogalmazza meg. Ezért a kliens által megadott adatok helyessége kulcsfontosságú, ugyanis helytelen értékű vagy típusú adatok a web alkalmazás lefagyását okozhatják vagy akár a webszerver működését is károsan befolyásolhatják. A felhasználó által megadott adatok helyességének ellenőrzése mindig a web alkalmazást fejlesztőjének feladata. A fejlesztőnek gondoskodnia kell arról, hogy a webalkalazást megvédje a hibás vagy káros adatoktól és ugyanakkor kódjában visszajelzéseket kell beépítenie a felhasználó részére, melyben felhívja a figyelmét a hibás adatokra.

Mielőtt az adatok felhasználásra kerülnek, feltétlenül ellenőriznünk kell az adatok helyességét, azaz az adatokat érvényesíteni (validate) kell. Ezt a folyamatot nevezzük adatérvényesítésnek. Az ASP.NET előtt a fejlesztőnek két lehetősége volt az adatok érvényesítésére. Ellenőrizhette az adatokat a kliensoldalon vagy a szerveroldalon. Mindkét esetnek megvannak a maga előnyei és hátrányai. A kliensoldali érvényesítés előnye a sebesség. Mivel az érvényesítő eljárás a kliensoldalon, azaz a böngészőben történik, a lapot nem kell visszaküldeni a szerverre. Az adatérvényesítés ebben az esetben JavaScript szkriptekkel történik. A módszer hátránya éppen ebben rejlik. A különböző eredetű és verziójú böngészők különböző JavaScript verziókat támogatnak. A szerveroldali adatérvényesítés független a használt adatérvényesítőtől. A felhasználó által megadott adatok felkerülnek a szerverre és ott történik az érvényesítésük. Ez a módszer valamelyest lelassítja a

folyamatot, de cserébe abszolút támogatást biztosít. Nagy adatmennyiség esetében a folyamat annyira lelassulhat, hogy ez a módszer használhatatlanná válik.

Az ASP.NET bevezetése előtt, a fejlesztőknek választaniuk kellett a két módszer között. Az ASP.NET adatérvényesítésre szakosított vezérlőket vezet be, az úgynevezett érvényesítő vezérlőket(validation controll). Az érvényesítő vezérlők automatikusan generálnak úgy kliens oldali, mint szerveroldali adatérvényesítő kódot. Ha a lapot lekérő böngésző támogatja a JavaScript szkripteket, akkor a böngészőhöz küldött kód tartalmazza a megfelelő JavaScript adatérvényesítő szkripteket. Figyelembe kell vennünk azonban, hogy az automatikusan generált kliensoldali adatérvényesítő kód, nem minden böngészővel működik kifogástalanul. Ha a lapot lekérő böngésző nem támogatja a JavaScript szkripteket, akkor az ASP.NET szerveroldali adatérvényesítésnek megfelelő kódot generál.

A Label vezérlőtől származnak, a BaseValidator absztrakt osztály közvetítésével, az úgynevezett érvényesítő vezérlők (validator control): a CompareValidator, a CustomValidator, a RangeValidator, a RegularExpressionValidator, és a RequiredFieldValidator. A BaseValidator osztály rendelkezik az EnableCrientScript tulajdonsággal, melynek értékei true vagy false. Az érvényesítő vezérlők EnableClientScript tulajdonságának alapértelmezett értéke true, mely aktiválja a kliensoldali adatérvényesítést. Ha egy bizonyos vezérlő estében ki akarjuk kapcsolni a kliensoldali adatérvényesítést, akkor elegendő, ha az EnableClientScript tulajdonság értékét false-ra állítjuk. A BaseValidator egy máik fontos tulajdonsága a ControllToValidate tulajdonság. Az érvényesítendő vezérlőket egy vagy több érvényesítő vezérlőhöz kell kötnünk ahhoz, hogy értékét érvényesítsük. A BaseValidator osztály akkor indítja el az érvényesítő folyamatot, ha a felhasználó rákattint a lapon elhelyezett Button, ImageButton, LinkButton web vezérlőkre vagy HtmlButton, HtmlInputButton vagy HtmlInputImage HTML vezérlőkre.

A Page osztály Validators tulajdonsága a lapon elhelyezett érvényesítő vezérlők gyűjteményét adja vissza. A fent említett vezérlők egyikére kattintunk, akkor ezzel meghívjuk a Page osztály Validate metódusát. Ez a metódus gondoskodik arról, hogy a Validators gyűjteménybe foglalt összes érvényesítő tényező elvégezze érvényesítési feladatát. A Validate metódus végrehajtását nevezzük érvényesítő feladatnak. Ha az érvényesítő folyamat alatt mindegyik érvényesítő vezérlő sikerrel végezte el feladatát, akkor a Validate metódus true értékkel látja el a Page osztály IsValid tulajdonságát. Ha legalább egy érvényesítő tényező

nem tudta sikeresen elvégezni feladatát, akkor az IsValid tulajdonság értéke false. Ügyeljünk arra, hogy mindig vizsgáljuk meg az IsValid tulajdonság értékét. Ha az IsValid tulajdonság értékét a visszaküldő gomb Click eseményének kezelőjében vizsgáljuk meg, akkor biztosak lehetünk abban, hogy a Page osztály Validate metódusa már végre lett hajtva, tehát az IsValid tulajdonság már megkapta a megfelelő értéket. Ha viszont az IsValid tulajdonság értékét a Page_Load eseménykezelőben vagyunk kénytelenek megvizsgálni, akkor az IsValid tulajdonság még nem kapta meg az érvényesítő folyamatból adódó értékét. Ez azzal magyarázható, hogy egy érvényesítő folyamat mindig a gomb Load és Click események között zajlik le. Ebben az esetben kénytelenek vagyunk kódból meghívni a Page osztály Validate metódusát, még mielőtt megvizsgálnánk az IsValid tulajdonság értékét.

Szintén az érvényesítő vezérlők csoportjába szokták sorolni a WebControl vezérlőtől közvetlenül származó ValidationSummary vezérlőt.

9.2.3. A gazdag vezérlők

Közvetlenül a WebControl vezérlőtől származnak az úgynevezett „gazdag” vezérlők (rich control). Ezek az AdRotator és a Calendar.

Az AdRotator a hirdetésre használt képeket felváltva jeleníti meg. A vezérlő több tulajdonsággal rendelkezik. Az AdvertisementFile meghatározza a hirdetéseket definiáló XML állomány helyét. A KeywordFilter meghatározza a kiválasztó szűrő értéket, a hirdetéseket definiáló XML állományban megadott hirdetések közül csak a szűrő által kiválasztott hirdetések jelennek meg. A Target attribútum meghatározza a hirdetést megjelenítő ablakot vagy keretet. Lehetséges értékei: _top, _self, _parent, vagy _blant. A vezérlő egyetlen eseménye az AdCreated esemény. Mely akkor lesz kiváltva, amikor a vezérlő egy hirdetést kiolvasott a hirdetéseket definiáló XML állományból, de még a hirdetés magjelenítése előtt. Az AdCreated esemény kezelője az OnAdCreated metódus.

A hirdetéseket definiáló állomány egy XML állomány. Ennek felépítése a következő:

```
<Advertisements>
    <Ad>
        <ImageUrl> Kép Url-je</ImageUrl>
        <NavigateUrl> Kapcsolat Url-je </NavigateUrl>
        <AlternateText> Alternatív szöveg </AlternateText>
```

```
            <Keyword> aulcsszó </Keyword>
            <Impressions> Gyakoriság </Impression>
        </Ad>
    </Advertisement>
```

Az <ImageUrl> megszabja a hirdetés képállományának URL-jét. A <NavigateUrl> a hirdetés kapcsolatát rögzíti. Mindkét elemben teljes Url-t, az alkalmazás törzsmappájához viszonyított relatív elérési utat vagy a hirdetéseket definiáló XML állományt tartalmazó mappához viszonyított relatív elérési utat használhatunk. Az <AlternateText> által meghatározott szöveg a hirdetés gyorstippjében jelenik meg, illetve, a képmegjelenítést nem támogató böngészőknél a képet helyettesíti. A <Keyword> elem segítségével csoportosíthatjuk a hirdetéseket, és a KeywordFilter tulajdonsággal megszabhatjuk, hogy melyik hirdetéscsoportot akarjuk használni. Az <Impressions> elemmel megszabjuk a hirdetések megjelenítésének gyakoriságát. A gyakoriságot egy számmal adjuk meg, mely az állományban definiált összes hirdetéshez viszonyítva adja meg a hirdetés relatív gyakoriságát. A hirdetéseket definiáló állományban egyéni elemeket is definiálhatunk. Az egyéni elemekkel megadott attribútumokat a vezérlő egy gyűjteménybe vonja össze, ahonnan lekérdezhetjük és felhasználhatjuk.

A hirdetők érdekeltek abban, hogy hirdetésük nézettségét statisztikailag is ellenőrizhessék. Az AdRotator AdCreated eseményének kezelőjével könnyen ellenőrizhetjük a hirdetésének nézettségét. Az eseménykezelő második paramétere egy AdCreatedEventArgs objektum. Ez az objektum rendelkezik egy AdProperties tulajdonsággal, mely egy szótárban adja vissza a hirdetéseket definiáló XML állományban megadott egyéni attribútumokat. Így könnyen számon tarthatjuk, hogy melyik hirdetést hozta létre a vezérlő AdCreated esemény kiváltásával. Az eredményeket szöveges állományokban tárolhatjuk, hogy bármikor felhasználhassuk.

A Calendar vezérlő egy dinamikus naptárt hoz létre a vezérlőt deklaráló lapon. A Calendar vezérlőt sok esetben használhatjuk. Leggyakrabban segédeszközként használjuk dátumok beadásánál, de segítségével többé-kevésbé bonyolult naptárakat is létrehozhatunk. A vezérlő számtalan tulajdonsággal rendelkezik. A CellPadding meghatározza a cellák tartalmának távolságát a vezérlő keretéhez, pixelben kifejezve. A CellSpacing meghatározza a cellák közti távolságot pixelben kifejezve. A DayNameFormat meghatározza a nap/év formátumát.

Lehetséges értékei: FirstLetter, FirstTwoLetter, Full vagy Short. Alapértelmezett értéke Short. A FirstDayOfWeek attribútum meghatározza a hét elő napját. Ez a nap a naptár első oszlopában kap helyet. Lehetséges értékei: Default, illetve Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, és Sunday. Alapértelmezett értéke Default. A NextMonthText meghatározza a következő hónapra utaló kapcsolat szövegét, a PrevMonthText meghatározza az előző hónapra utaló kapcsolat szövegét. A NextPrevFormat meghatározza a Next és a Previous kapcsolatok formátumát. Lehetséges értékei: CustomText, FullMonth és ShortMonth. Alapértelmezett értéke a CustomText. A SelectedDate visszaad egy DateTime típusú értéket, mely a kiválasztott napot határozza meg, a SelectedDate egy DateTime típusú értékeket tartalmazó gyűjteményt ad vissza, amely a kiválasztott napokat tartalmazza. A SelectionMode meghatározza, hogy napokat, heteket vagy hónapokat választhatunk ki a naptárban. Lehetséges értékei: None, Day, ekkor napok választhatók, DayWeek, ekkor napok és hetek választhatók, DayWeekMonth, ekkor napok, hetek és hónapok választhatók. Alapértelmezett értéke None, ekkor nem választható semmi. A SelectMonthText meghatározza a hónap kiválasztására használt szöveget, a SelectWeekText a hét kiválasztására használt szöveget határozza meg. A ShowDayHeader meghatározza a napok nevét tartalmazó fejléc megjelenítését. Lehetséges értékei true vagy false. Az alapértelmezett értéke true. A ShowGrindLines meghatározza az elválasztó vonalak megjelenítését. Lehetséges értékei true vagy false. Az alapértelmezett értéke true. A ShowNextPrevMonth meghatározza az előző és a következő hónapra utaló kapcsolat megjelenítését. Lehetséges értékei true vagy false. Az alapértelmezett értéke true. A ShowTitle meghatározza a naptár címének a megjelenítését. Lehetséges értékei true vagy false. Az alapértelmezett értéke true. A TodaysDate megahatározza az aktuális napot a naptárban, a VisibleDate meghatározza az aktuális hónapot a naptárban. Értékük DateTime típusú. A vezérlő három eseménnyel rendelkezik. Ezek a DayRender, a SelectionChanged, VisibleMonthChanged. A DayRender esemény akkor lesz kiváltva, amikor a vezérlő létrehoz egy napnak megfelelő cellát. Az esemény kezelője az OnDayRender metódus. A SelectionChanged esemény akkor lesz kiváltva, amikor a felhasználó egy új napot, hetet vagy hónapot választ ki. Az esemény kezelője az On SelectionChanged metódus. A VisibleMonthChanged vezérlő akkor lesz kiváltva, amikor a felhasználó az előző vagy a következő hónapra utaló kapcsolatra kattint. Az esemény kezelője az On VisibleMonthChanged.

9.2.4. A listavezérlők

A WebControl vezérlőtől a ListControl absztrakt osztály közvetítésével, származnak a listavezérlők. Ezek a CheckBoxList, a DropDownList, a ListBox és a RadioButtonList. Mindegyik listavezérlő egy listába összegyűjtött listaelemekből áll. A listának és a listaelemeknek a megjelenése és funkcionalitása mindegyik listavezérlő esetében más. A CheckBoxList vezérlő listájában a listaelemek jelölőnégyzet alakjában jelennek meg, és ezekből akárhányat kijelölhetünk. A RadioButtonList listaelemei választógombok, és ezek közül mindig csak egyet jelölhetünk ki. A ListBox listája egy klasszikus lista formájában jelenik meg, és a listából egy vagy több elemet jelölhetünk ki. A DropDownList listaelemei egy legördülő listát töltenek ki, és ezek közül mindig csak egyet jelölhetünk ki.

A listavezérlők a közös tulajdonságaikat a ListControl absztrakt osztálytól öröklik. A legfontosabb tulajdonság az Items. Ez a tulajdonság egy ListItemCollection típusú gyűjteménybe adja vissza a ListControl listáját. A ListItemControl ListItem típusú elemeket tartalmaz. Az Item gyűjtemény a megszokott gyűjteménymetódusokkal rendelkezik, mint például a Count, amely visszaadja a listába foglalt elemek számát, vagy az Add metódus, mely segítségével új elemet adhatunk a listához. A listából eltávolítani a Remove metódussal lehet. A gyűjtemény elemeit egyszerű indexeléssel érhetjük el, vagy Enumerator (felsorolás) használatával.

9.2.5. Az adatkezelő vezérlők

Szintén a WebControl osztálytól származnak az adatkezelő vezérlők (data control). Ezek közvetlenül a BaseDataList absztrakt osztálytól származnak. Ezek a vezérlők a DataGring és a DataList. A Repeater vezérlőt is az adatmegjelenítő vezérlőkhöz soroljuk, bár ez az osztály a System.Web.UI.Control osztálytól származik. Az adatkezelő vezérlők képessége nagyon sokrétű és majdnem, hogy korlátlan lehetőségeket biztosítanak az adatok megjelenítésére.

9.2.6. Az Xml vezérlő

Végül ehhez a névtérhez tartozik még az XML vezérlő. A .Net keretrendszer sok olyan osztályt tartalmaz, melyek segítségével könnyen és hatékonyan kezelhetjük az XML adatokat.

Ezek az osztályok több névtérben vannak definiálva. A legfontosabb osztályok az XmlTextReader, XmlTextWriter, az XmlDocument és az XmlTranszform. Az XmlTextReader lehetővé teszi az XML állományok olvasását. Az olvasás csak az állomány elejétől kezdve, az állomány vége felé lehetséges. Ezt nevezzük egyirányú navigálásnak. Az állományt nem alakítja át egy DOM (Document Object Model) objektumra. Az XmlTextWriter lehetővé teszi az Xml állományok írását. Az XmlDocument lehetővé teszi egy Xml állomány reprezentálását egy DOM objektum segítségével, mely kétirányú navigálást biztosít az XML adatok között. Az XmlTransform lehetővé teszi az XML dokumentumok transzformálását XSL stíluslapokkal.

9.2.7. A MultiView és a View vezérlőelemek

Ezen vezérlőelemek segítségével könnyen kezelhetők a nagy mennyiségű adatgyűjtésben résztvevő felületek. A MultiView a panelszerű vezérlőelemek (View vezérlőelemek) tárolójaként viselkedik. Segítségével a benne lévő különböző nézetek végiglapozhatók.

A MultiView alkalmanként csak egy nézetet jelenít meg, a legfőbb feladata, pedig, hogy a nézetek készletét szabályozza. Az, hogy éppen melyik nézetet jelenítse meg a MultiView, az ActiveViewIndex tulajdonságát kell állítani. Az első elem indexe a 0, a másodiké az 1, és így tovább. A MultiView is tartalmazhat vezérlőelemeket, ezek alapesetben minden egyes View-al együtt megjelennek.

10. A Mono

A Mono egy olyan szoftver, amely segítségével .NET alkalmazásokat tudunk fejleszteni és futtatni Linux, Solaris, vagy esetleg Mac OS X operációs rendszereken. Segítségével használhatjuk a .NET keretrendszert, így Mono segítségével nem csak Microsoft platformon van lehetőségünk ASP.NET webalkalamzást írni, hanem akár Linuxon is. A megírt alkalmazást akár közzé is tehetjük, mert a Mono segytségével ASP.NET kéréseket is ki tud szolgálni a webszerverünk.

ASP.NET kéréseket kétféleképpen tudjunk feldolgozni. Az egyik módszer Apache segítségével mod_mono-t használva. Ez egy modul, amely lehetővé teszi az Apache-nak,

hogy az ASP.NET alkalmazásokat szolgáljanak ki. A másik lehetőség, ha egy xsp használunk. Ez egy egyszerű webszerver, amelyet C#-ban írtak. Ha elindítjuk az xsp programot, akkor az figyeli a 8080-as portot, de természetesen átállíthatjuk tetszőleges a figyelést egy megfelelő portra.

11. A Demó alkalmazás

A szakdolgozathoz készített alkalmazás egy bejelentkező és egy regisztrációs lapot tartalmaz, illetve egy fő lapot, amely csak a választott login nevünkön üdvözöl minket. Az alkalmazás futtatásához a Visual Studio Development Serverét használtam. Természetesen akármilyen ASP.NET kérést kiszolgálni képes webszerver alkalmas.

Az alkalmazás első lapján egy login screen található. Itt megadhatjuk a nevünket és a jelszavunkat, illetve, ha nem vagyunk regisztrálva, akkor regisztrálhatjuk magunkat a Register linkre kattintva. Ebben az esetben a Register.aspx lapra navigálunk, amelyen egy MultiView vezérlőt helyeztem el. A vezérlő három View vezérlőt tartalmaz. Az első View vezérlőn a személyes adatokat kell megadnunk. A *-al jelölt mezők kitöltése kötelező. Ez a lapon megfigyelhetjük, hogyan működnek a ASP.NET érvényesítő vezérlői. Ha egy mezőt nem töltünk ki, vagy a jelszó megerősítése hibás, esetleg rossz a formátuma az emil címnek, akkor a lap ez jelzi. Ha a „Next” gombra kattintunk, és minden rendben van, akkor a második View vezérlő jelenik meg. Ezen a lapon néhány érdekesebb vezérlő működését figyelhetjük meg. Kiválaszthatjuk egy DropDown vezérlővel, hogy milyen országban lakunk, ha ez megtörtént, akkor az adott országban lévő városok közül kiválaszthatjuk, egy ListBox-ban, hogy melyik városban lakunk. Ha a kiválasztott ország USA, akkor megjelenik egy új ListBox vezérlő, amely az államokat tartalmazza. Egy RadioButton vezérlő segítségével kiválaszthatunk egy képet, illetve CheckBox-ok segítségével megadhatjuk, hogy mi a hobbink. Ezeket az információkat eltárolhatjuk és később felhasználhatjuk. Az utolsó View egy összegzés, a megadott adatokról, ha mindent rendben találtunk, akkor elkészíthetjük az account-unkat. Ekkor a program ellenőrzi, hogy időközben nem foglalta-e le valaki a kívánt felhasználónevet. Elég kicsi rá az esély, de nem lehetetlen. Ha időközben foglalt lett a név, akkor az első View lesz megint aktív, és más nevet kell választanunk. Természetesen a kurzor a megfelelő mezőben áll, amelyet a SetFocus() segítségével lehet beállítani. Ha kész az account akkor megjelenik egy gomb, amely segítségével visszanavigálhatunk a belépési oldalra, és

bejelentkezhetünk. Ha begépeljük a megfelelő felhasználó nevet és jelszót akkor a főlap üdvözöl minket. Az, hogy egy adatot (például egy mező értékét) egy másik lapon olvashassuk, ahhoz Session objektumhoz kell hozzáadni a küldő lapon a küldendő értéket (amelyet egy kulccsal fogunk azonosítani a másik lapon), és be kell állítani, hogy mely lapnak küldjük.

```
String Data;
/*.........*/
Session.Add("Key", Data);
Server.Transfer("Default.aspx");
```

Ekkor a Data változó értékét tudjuk átküldeni a Default.aspx lapnak. Az értéket a másik lapon a kulcs segítségével tudjuk elérni.

```
String Data = (String)Session["Key"];
```

Természetesen több értéket is hozzá adhatunk a laphoz, és a kulcs segítségével hozzájuk is férhetük, de csak azon a lapon, amelynek küldtük.

Irodalomjegyzék

George Shepherd: ASP.NET 2.0 Step by step

Hatvany Béla Csaba: ASP.NET vezérlők programozása

Visual Studio.NET – The .NET Framework Black Book

MSDN Libary

Felhasznált internet oldalak

www.mono-project.com

www.msdn.com

www.prog.hu

www.wikipedia.com