ASP.NET INTERNETES ALKALMAZÁSFEJLESZTÉS

2010.02.19.

Elméleti áttekintés

Bemutatkozás

- Tarcsi Ádám
- e-mail: ade@inf.elte.hu

Tematika

- Elméleti áttekintés
- A .NET keretrendszer
- A Visual Studio .NET fejlesztői környezete
- ASP.NET (C#)
  - Webűrlapok használata
  - Szerveroldali Web és HTML vezérlőelemek
  - Page objektum és az eseménykezelés
- Adatkezelés: XML, ADO.NET és LINQ
- Web szolgáltatások
- Web alkalmazások konfigurálása, hitelesítése és telepítése
- AJAX (?)

Követelmények - értékelés

- Gyakorlati dolgozat:
- A feladatok a következő témaköröket ölelhetik fel:
  - Szerveroldali programozás ASP.NET és ADO.NET technológiával
  - Egyszerű Web Service létrehozása

Ajánlott irodalom

- Scott Mitchell: ***Tanuljuk meg az ASP.NET 2.0 használatát 24 óra alatt.*** Kiskapu Kiadó, Budapest, 2007
- Dr. Hatvany Béla Csaba: ***ASP.NET vezérlők programozása.*** ComputerBooks, Budaspest, 2006.
- George Shepherd: **Microsoft *ASP.NET 2.0 lépésről lépésre.*** Szak Kiadó, Bicske, 2006

Előfeltételek

- HTML
- XML szerkesztés
- Kliensoldali programozás

Bevezetés, témafelvetés

- Hálózati, szerver és kliens oldali megoldások (TCP/IP és HTTP protokoll és működése)
- WEB-es prezentációs megoldások (HTML nyelv, CSS), web-grafika
- WEB szerverek, böngészők, kliens oldali WEB programozás alapjai (pl. JavaScript)
- Adatbázis-kezelés (a relációs modell, adatmodellezés, SQL)
- Rendszerek közti adatkommunikáció „önleíró” dokumentum nyelven = XML (XML felépítése, használata, kapcsolódó technológiák érintőlegesen: DTD, XSD, XSL ill. XSLT)
- XML alapú adatbázisok (XML adattárolás alapjai, lekérdező nyelvek: XPath, XQuery)
- Multimédiás adatbázisok (nagy méretű multimédiás anyagok tárolása adatbázisokban, visszakeresés, hatékonyság)
- Programozási módszertan
- WEB programozás (módszerek, beágyazott script-nyelvek, általában PHP)
- Vállalati környezetre tervezett webes fejlesztői környezetek (pl.: .Net, Java)
- Multimédiás WEB programozás – bináris tartalmak (stream-ek, header, letöltés, feltöltés)
- Multimédiás WEB programozás – vektorgrafikus és programozott tartalmak (SVG, Flash)

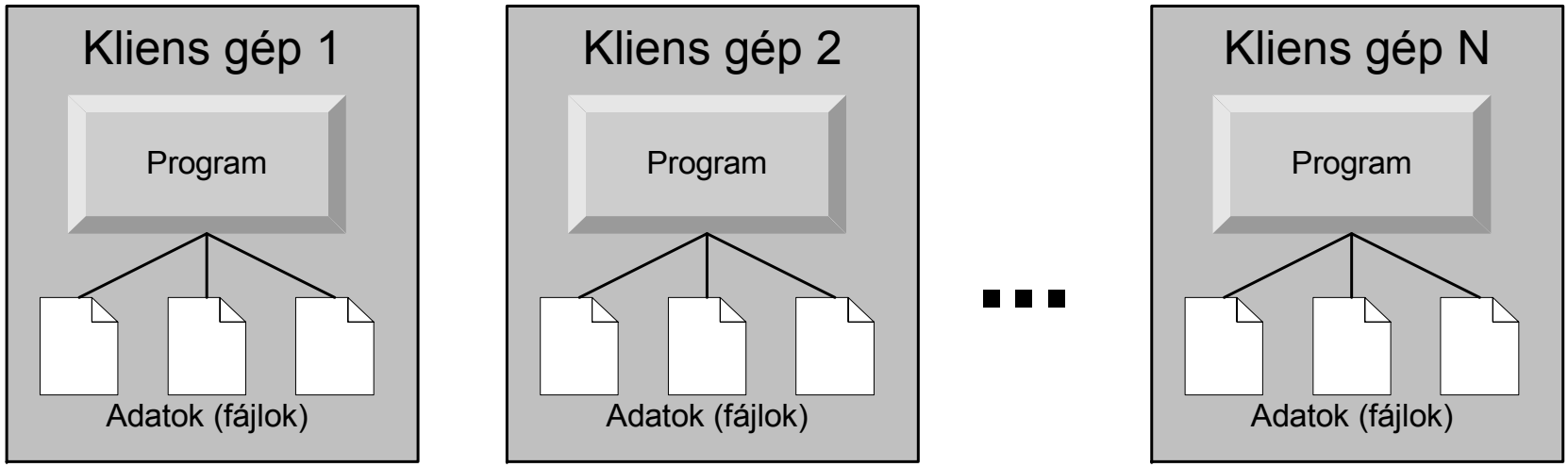
Bevezetés, témafelvetés

- Informatikai biztonság (adatvédelem, kommunikációs vonalak védelme, védelem illetéktelen behatolásokkal szemben, meghibásodások elleni védelem)
- Szoftvertervezés
- Projektmenedzsment
- Szoftvertesztelés

9

Architektúrák - evolúció

Egygépes (standalone) alkalmazások



- A program teljes egészében a munkaállomáson fut.
- Az adatok ugyanitt tárolódnak.
- Egyszerre csak egy felhasználó használhatja.
- Semmilyen hálózati kapcsolat nincs, a különálló programok közti adatszinkronizáció meglehetősen nehézkes.

Egyszerű kliens-szerver alkalmazások 1.

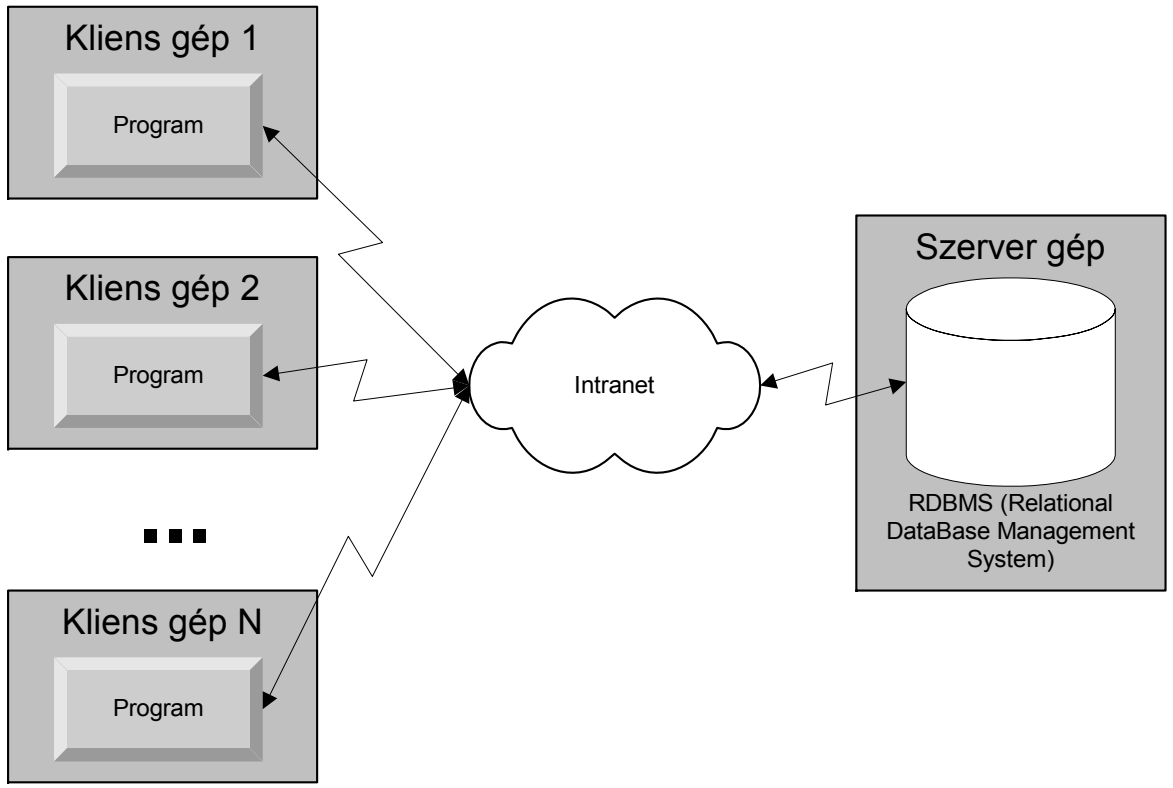
- Egy vagy több szerver gép erőforrásait (jellemzően adatait) megosztja a kliensek között. Jobb esetben on-line.

- Az alkalmazás egy része (adatbázis-kezelő rsz.) a szerven fut.

- Az alkalmazás logikát implementáló rész a kliens gépeken fut. „***vastag kliens rendszerek***”

- Egy adatbázist többféle kliens program is használhat.

- Egyszerre több konkurens felhasználó használhatja.



Egyszerű kliens-szerver alkalmazások 2.

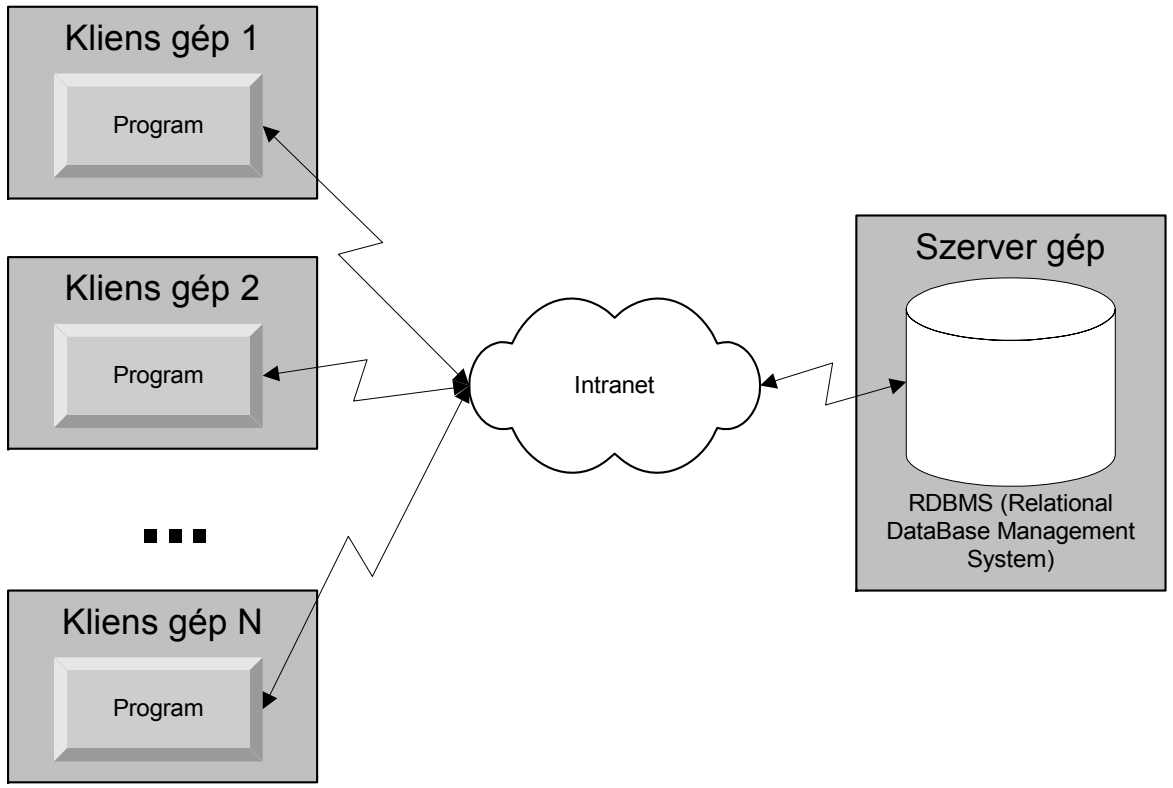
- Jellemzően intranet-es alkalmazásoknál használatos.

- Terheli a kliens gép erőforrásait.

- Gyakran mindenféle driver-ek telepítését igényli a kliens gépeken

- Verziófrissítés alkalmával az összes kliens-en frissíteni kell a programot.

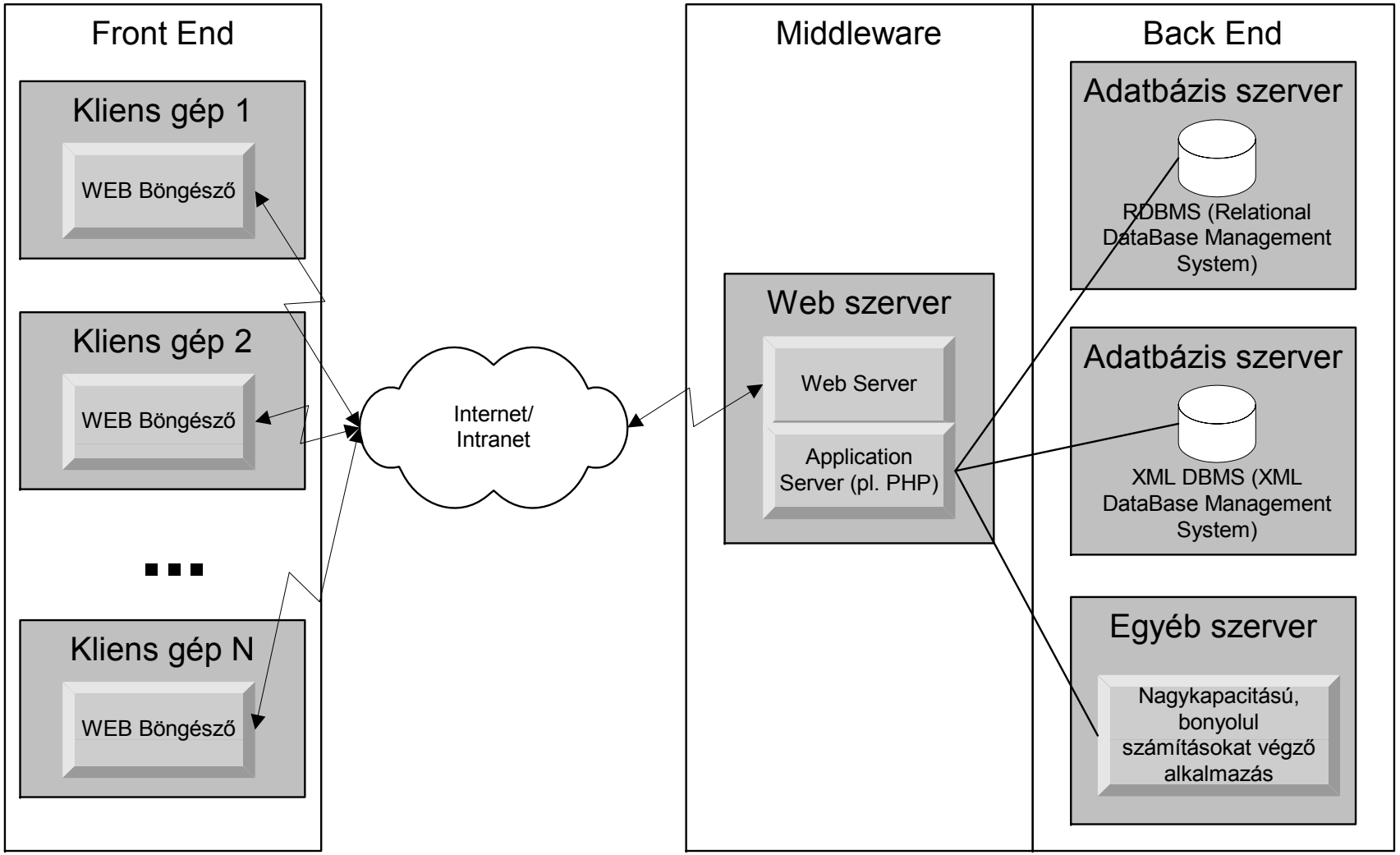
- A RAD (Rapid Application Development) sok eszközzel támogatott, számos jó vizuális fejlesztőkörnyezet: gyorsan „összekattint-gathatunk” és leprogramozhatunk komoly alkalmazásokat.



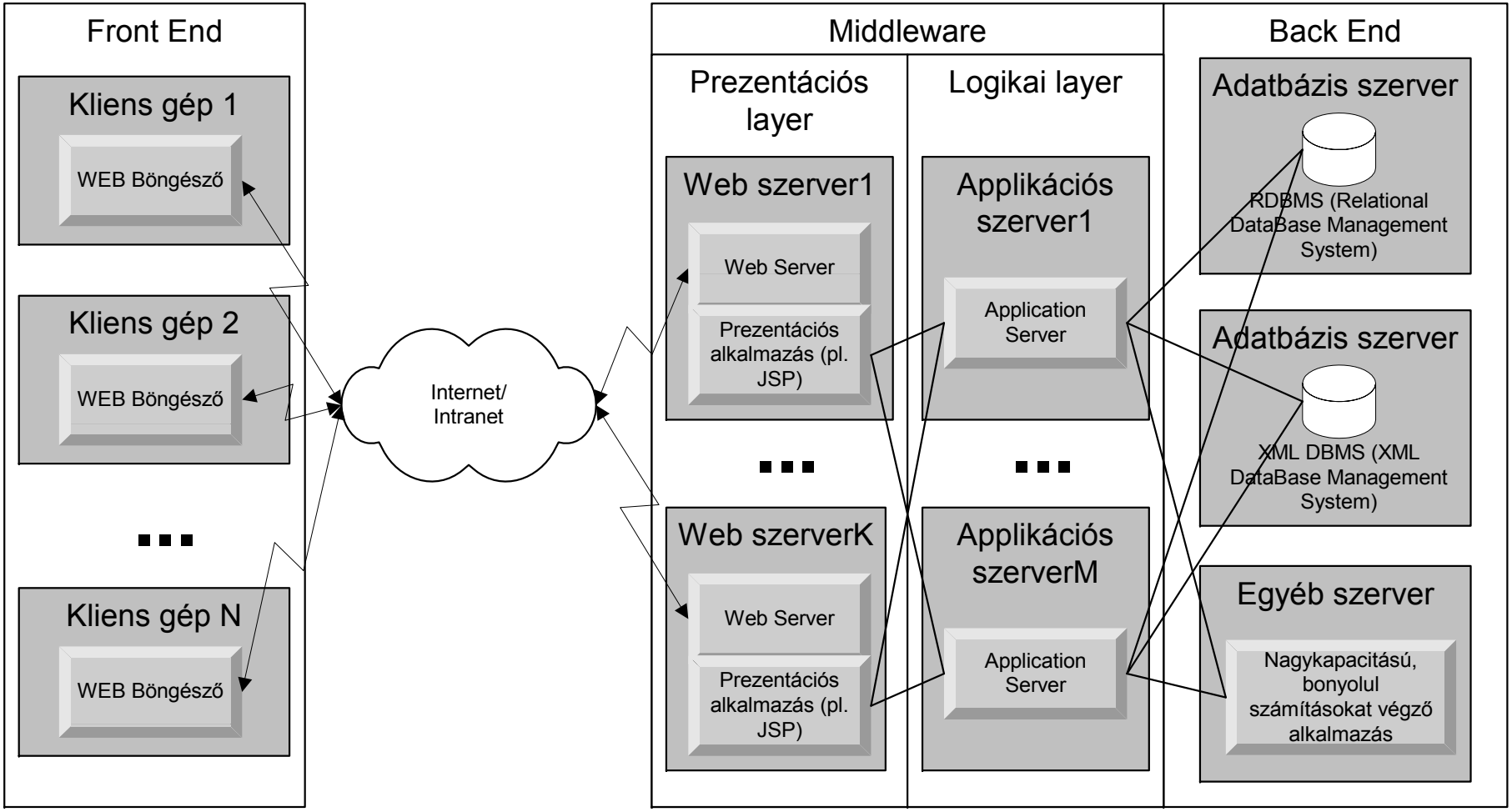
Többrétegű (multitier) hálózati alkalmazások

- Minimálisan három réteg létezik:
  - Front End = kliens oldali felhasználói réteg (általában egy WEB böngészőben)
  - Middleware = szerver oldali prezentációs és logikai réteg (általában egy WEB szerveren beágyazott script-ekben összeolvasztva a megjelenítés és az egyszerűbb logika)
  - Back End = hátsó szerver oldali nagykapacitású tároló (adatbázis szerver) vagy számoló réteg

Háromrétegű architektúra



Többrétegű architektúra



Többrétegű architektúra jellemzői

- Load balancing, terhelésmegosztás.
- Tervezést támogató környezetek: .Net, Java J2EE.
- Architektúra felosztás-összevonás logikai szinten.
- A rendszer logikai architektúrája (tervezés, programozás) független a számítógépes megvalósítástól, hálózattól.
- A logikai réteg tovább osztható.
- Nagyon sok konkurens felhasználó kiszolgálására optimalizálva

Többrétegű architektúra jellemzői

- Kliens gép: böngésző, a logika – többnyire – a szerveren található – vékony kliens architektúra
- Minimális logika a klienseken: a beviteli adatok validálására, a lapok speciális megjelenítésére (pl. JavaScript).
- A szerveren elkülönül az adattárolás, a logika és a prezentáció → eltérő szerepkörök
- Az egyes szintek önmagukban is tesztelhetőek.
- A rendszer egyes komponensei több célra vagy újra felhasználhatók.

Többrétegű architektúra jellemzői

- A vékony kliensek miatt nagyon gyenge kliens gépek is elegendők.
- A technológia platform-független.
- A kliensekre nem kell drivert telepíteni.
- A verziófrissítés csak a szervert érinti, a klienseket nem.
- Sajnos egyelőre elég kevés eszköz támogatja a RAD-ot (Rapid Application Development), a környezet kevés segítséget nyújt a programozónak a megoldási lehetőségek kiválasztásában
  - → „Házi szabványok”, saját keretrendszerek készülnek.
- Nehezebb tesztelni

Keretrendszerek

- Alapelemeket biztosító programozási rendszerek, melyek összetett feladatokat képesek támogatni.
- Egy vázat adnak alkalmazásaink számára.
- Bizonyos filozófiának megfelelő szabályok gyűjteménye
- Gyakori, ismétlődő műveleteket támogatnak, sablonokat biztosítanak:
  - munkamenetek,
  - adatbázis kapcsolatok,
  - hibakezelés, stb.

Miért van szükség keretrendszerekre?

- Ömlesztett kód: HTML(, CSS), logika (pl.: PHP), egyben van
- Ömlesztett logika: vezérlés, adatelérés, megjelenítés, feldolgozás, alkalmazáslogika egyben van
- Csoportmunka nem lehetséges
- Konfiguráció kódba épített
- Karakterkódolás eltérő lehet a HTML-ben, kódban és adatbázisban, nincs egységesen kezelve
- Több belépési pontja van az alkalmazásnak, ami biztonsági és jogosultsági kérdéseket vet fel

Keretrendszerek

- Előnyök
  - Fejlesztési időt takarítható meg
  - Rákényszerítik a programozót valamilyen kódolási standard-re → átláthatóbb kódok
  - Valamilyen szabályrendszert alkalmazását várják el.
  - Támogatják az újra felhasználhatóságot
  - Egységesebb alkalmazásfejlesztés
  - Szétválasztott kód és logika
  - Meghatározott könyvtárszerkezet
  - Csoportmunka támogatott: külön fileok az egyes szakembereknek (designer, adatbázisos, felületfejlesztő, stb.)
  - Rétegek szétválasztása
- Hátrányok
  - Tanuláshoz idő kell
  - Szabályok korlátokat is jelentenek

Keretrendszerek - példák

- Vastag-kliens RAD: Visual Studio, Borland Delphi, …
- PHP: CodeIgniter, Symfony, Zend Framework, CakePHP, …
- PHP keretrendszerek tartalomkezelő szolgáltatásokkal: Drupal, Joomla!, Wordpress, …
- JavaScript: JQuery, Prototype, script.aculo.us, Yahoo YUI, ExtJS

Front Controller

- Minden oldalt érintő közös műveletek egy helyen szerepelnek:
  - Konfiguráció beolvasása
  - Autentikáció
  - Autorizáció
  - Input paraméterek előfeldolgozása, szűrése
  - Karakterkódolás
- 1 belépési pont: pl.: index.php
- Hogyan jelezhető, melyik oldalt kell megjeleníteni?
  - hidden mező
  - GET paraméterrel: pl.: index.php?oldal=main

Model-View-Controller (Modell-Nézet-Vezérlő - MVC)

- Összetett, sok adatot kezelő alkalmazásokban elvárható:
  - Az adatok (modell) és a felhasználói felület (nézet) szétválasztása.
  - A nézet ne befolyásolja az adatkezelést.
  - Az adatok átszervezhetőek legyenek a felhasználói felület változtatása nélkül.

MVC – 2.

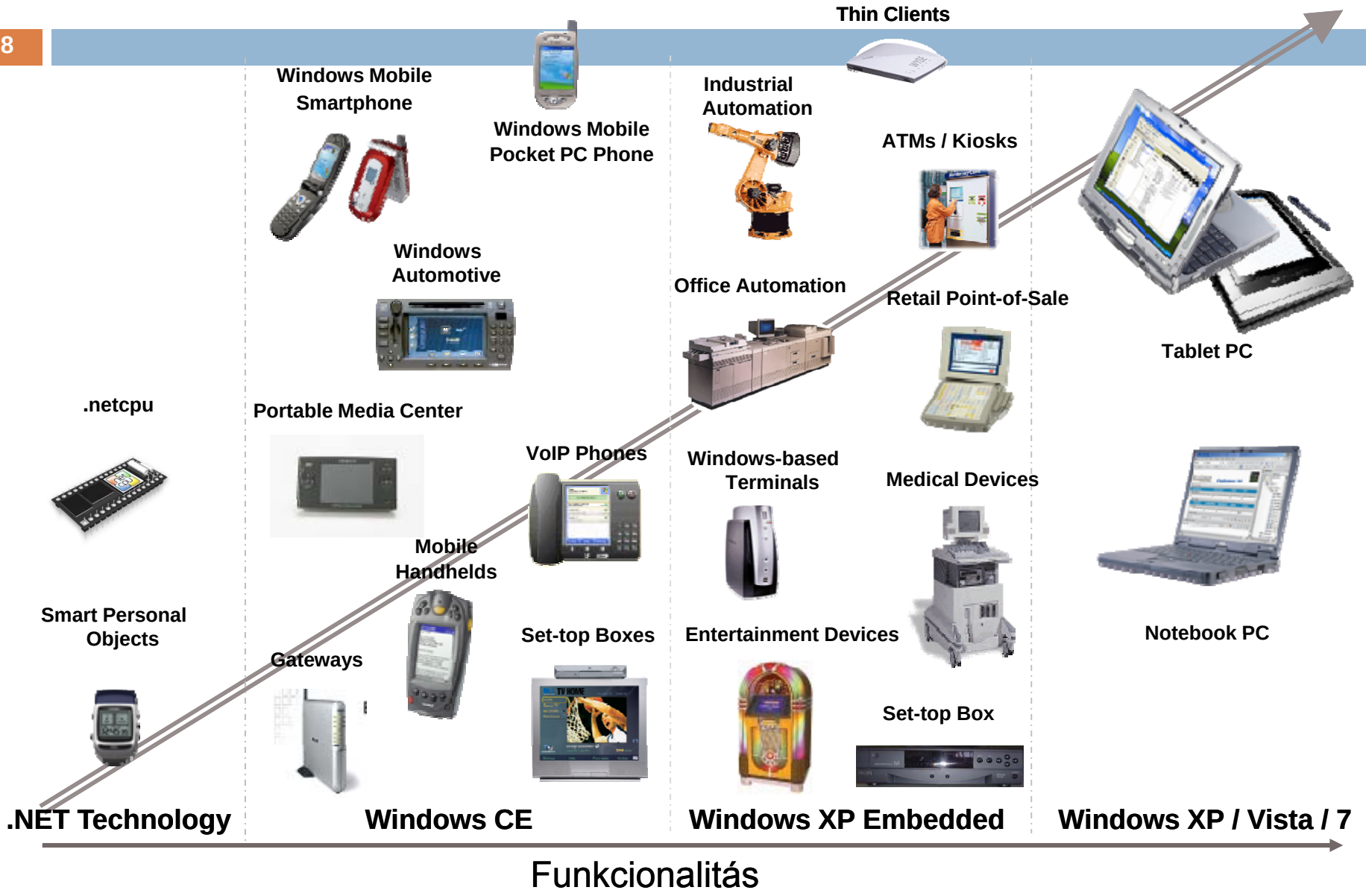
- Az MVC tervezési minta elkülöníti az adatok elérését és az üzleti logikát az adatok megjelenítésétől és a felhasználói interakciótól egy közbülső összetevő, a vezérlő bevezetésével.
  - Model: az adatokat, praktikusan az adatbázisbeli táblák, néha beleértik ezek elérését segítő kódot vagy magát az üzleti réteget is
  - View: megjelenítésért szükséges réteg
  - Controller: vezérlést végzi, értesíti a Modelt és a Viewt is a változásokról, kezeli az input adatokat.
- Össze szokás kapcsolni a FrontController mintával

.NET Framework

.NET Framework

- Keretrendszer
- RAD: Gyors alkalmazás fejlesztést biztosít
- Platformfüggetlenség
- Hálózati transzparencia
- Kiszolgálóoldali és asztali fejlesztésekhez
- A SUN Java Virtual Machine (JVM) konkurense

Fejlesztési platformok



.NET történelem

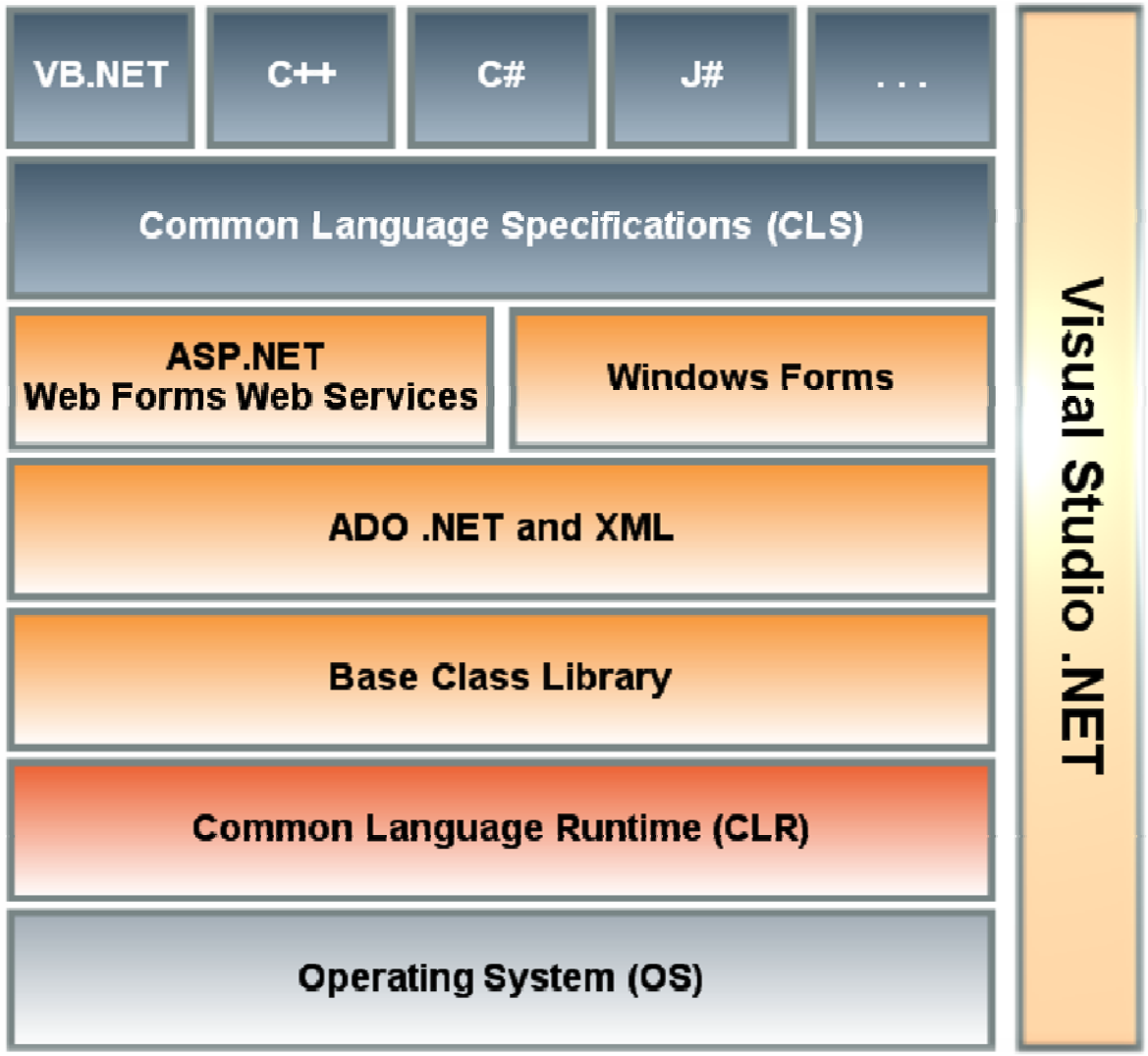
- 1996 Colusa Software felvásárlása
- 1998 Megromló kapcsolat a SUN-nal.
- 2000 .NET 1.0 beta 1
- 2002 Elérhető a .NET mint szoftverplatform (1.0)
- 2005 Elérhető a .NET 2.0
- 2006 Elérhető a .NET 3.0
- 2007 Elérhető a .NET 3.5
- 2010 .NET 4.0

.NET nyelvek

- Több mint 150 nyelv!
- Bizonyos nyelvek speciális feladatokra tervezték
- Az alkalmazás egyes részei eltérő nyelven is készülhetnek külön szerelvényeket (DLL) használva

C#	VisualBasic
J#	Visual C++
E#	Lisp
Php.NET	Python for .NET
Delphi.NET	Pascal for .NET
A# (Ada)	S# (Smaltalk)

.NET Framework



Common Language Runtime

- Egységes futási környezetet biztosít a .NET komponensek számára
  - függetlenül attól, hogy azokat milyen nyelven írták.
- Elvégzi a memóriakezelést,
- Biztonságos futási környezetet biztosít,
- Hozzáférést ad az operációs rendszer szolgáltatásaihoz.
- A CLR alatt futó kódot felügyelt kódnak nevezzük.

Base Class Library

- Osztálykönyvtárak, melyek alapból részei a .NET-nek.
- Objektum-orientáltak
- Általunk kiterjeszthető típusok

Common Type System

- Meghatározza, hogy a CLR hogyan definiálja és használja a típusokat.
- A CTS típusai minden nyelvben jelen kell legyenek
  - Például a CTS Int32 típus VB.NET-ben Integer és C#-ban int típusnak fel meg.

Common Language Specification (CLS)

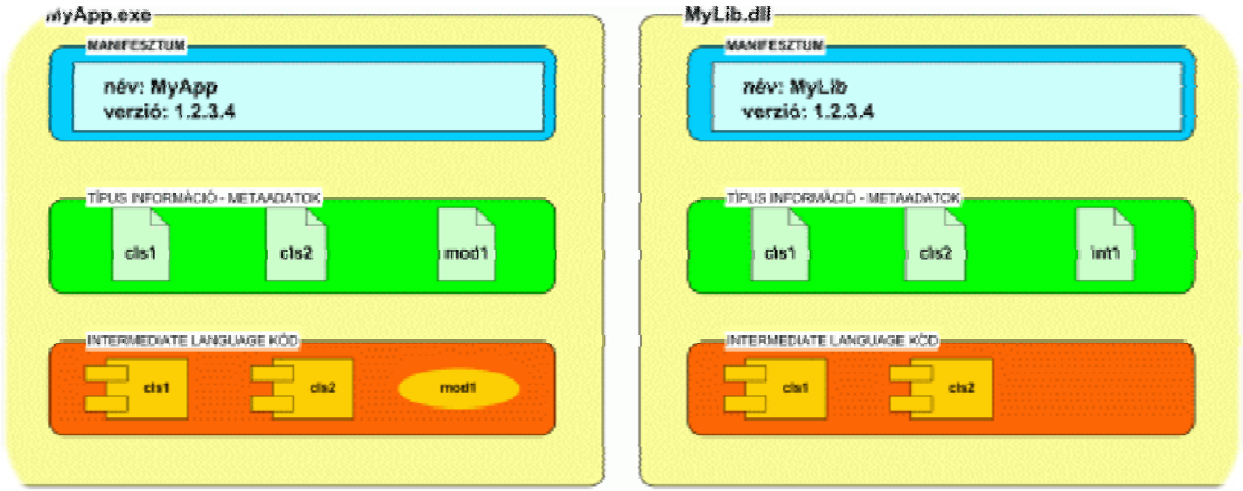
- A CLS általános szabályokat fektet le, amelyeket minden .NET-nyelv be kell tartson.

MS IL - köztes kód

- A .NET alatt megírt különböző nyelvű programokból a fordítóprogram először létrehoz egy egységes nyelvű köztes kódot. (IL-kód)
  - Elméletileg két különböző nyelvben létrehozott azonos szemantikájú szerkezetek, azonos IL-kódot eredményeznek.
- Az IL-kód gépi kódra fordítását a futásidejű (JIT) fordító végzi el.
  - A létrehozott exe vagy dll állományban tehát tulajdonképpen az IL-kód szerepel kiegészítve egy gépi kóddal, amely a JIT fordítót hívja meg azért, hogy az lefordíthassa az IL- kódot gépi kódra.
- A futásidejű fordítás ugyan időt vesz igénybe, de számos előnnyel jár.
  - Pl. lehetőség van arra, hogy a számítógép processzortípusának megfelelő kód jöjjön létre, kihasználva az adott processzor utasításkészletének előnyeit.

Assembly, metadata, manifest

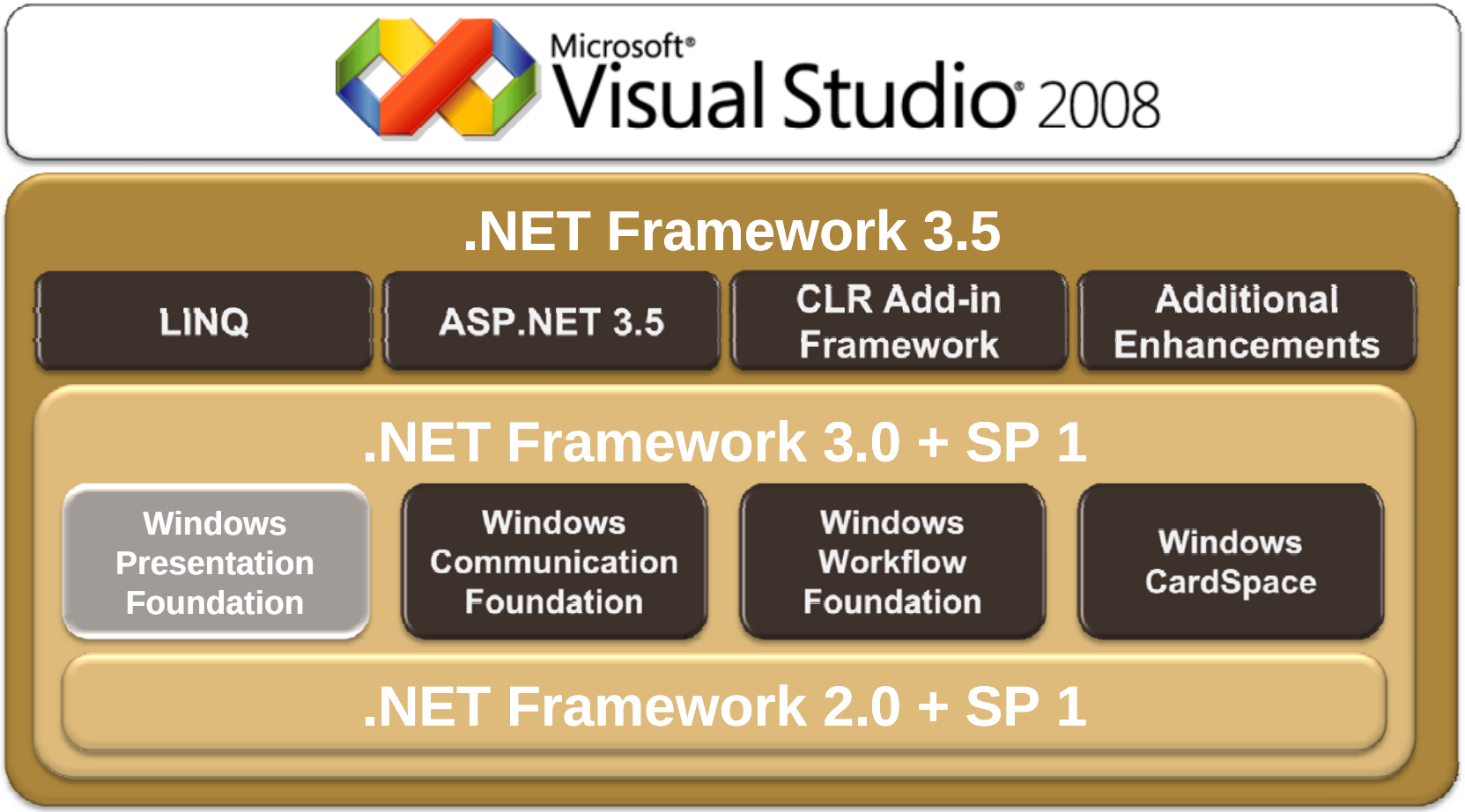
- **Assembly:** A lefordított .NET alkalmazás egy *.exe vagy *.dll állományba kerül, bináris alakban van, de nem gépkódot, hanem IL-kódot tartalmaz.
- **Manifest:** Név, verziószám, más assembly-től való függőségek, attribútumok
- **Metaadatok:** assembly osztályairól, programok esetében a programmodulokról, könyvtárak esetében a könyvtárinterfészről találhatók az adatok típusára vonatkozó információk.
- **IL kód:** Köztes nyelvi kódot tartalmazza



.NET jellemzői

- Konzisztens, egyszerűsített programozási modell
- Széles platform-elérés
- Programnyelvek integrálása
- Automatikus memóriamenedzsment
- Típusbiztonság
- Könnyebb hibakeresés
- Kivételkezelés
- Biztonság

.NET Framework változatok



Visual Studio

- .NET fejlesztői keretrendszer
- RAD
- Csoportmunkát támogatja

Visual Studio változatai

	Express	Standard	Pro	Team
Windows or Web Designers	+	✓	✓	✓
Code Editors and IntelliSense	✓	✓	✓	✓
Programming Languages	+	✓	✓	✓
Remote Data Access	–	✓	✓	✓
Mobile Device Development	–	✓	✓	✓
User Experience	Simplified	Simplified	✓	✓
Server Development/Debugging	–	–	✓	✓
SQL Server 2005 Development	–	–	✓	✓
Office Development	–	–	–	✓
Architecture, Development, Testing, and Project Management Tools	–	–	–	✓

Visual Studio történelem

- 1997 – Visual Studio 97
- 1998 – Visual Studio 6.0
- 2002 – Visual Studio .NET
- 2003 – Visual Studio .NET 2003
- 2005 – Visual Studio 2005
- 2008 – Visual Studio 2008
- 2010 – Visual Studio 2010

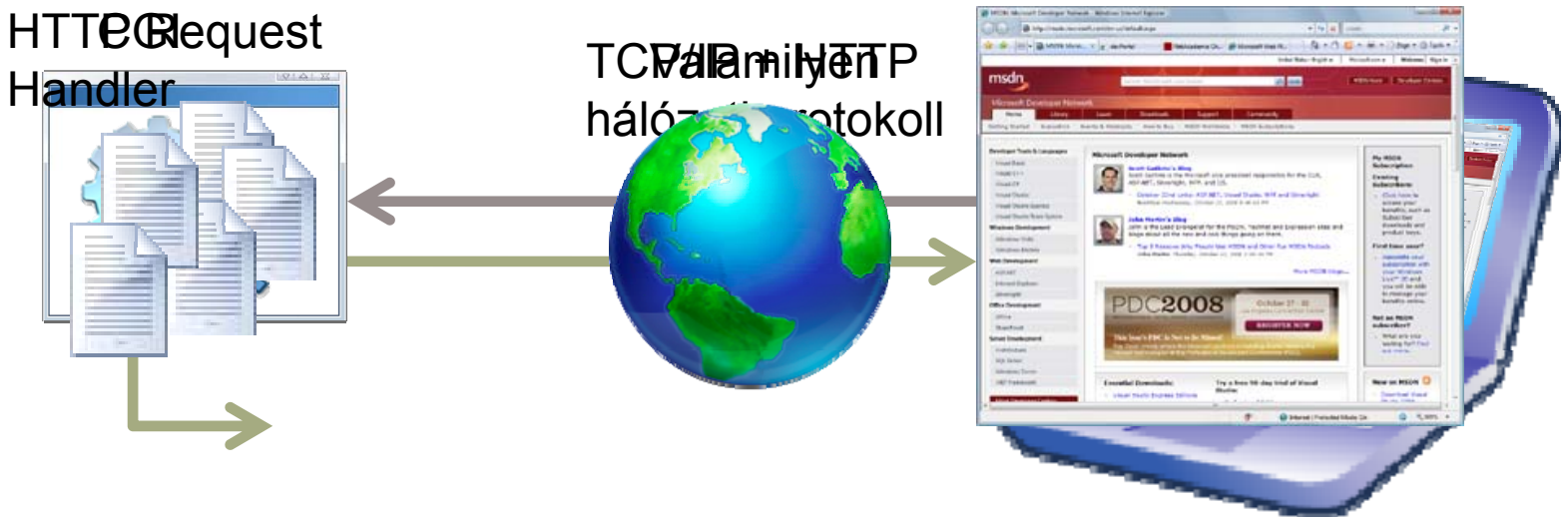
Internetes alkalmazásfejlesztés

ASP.NET

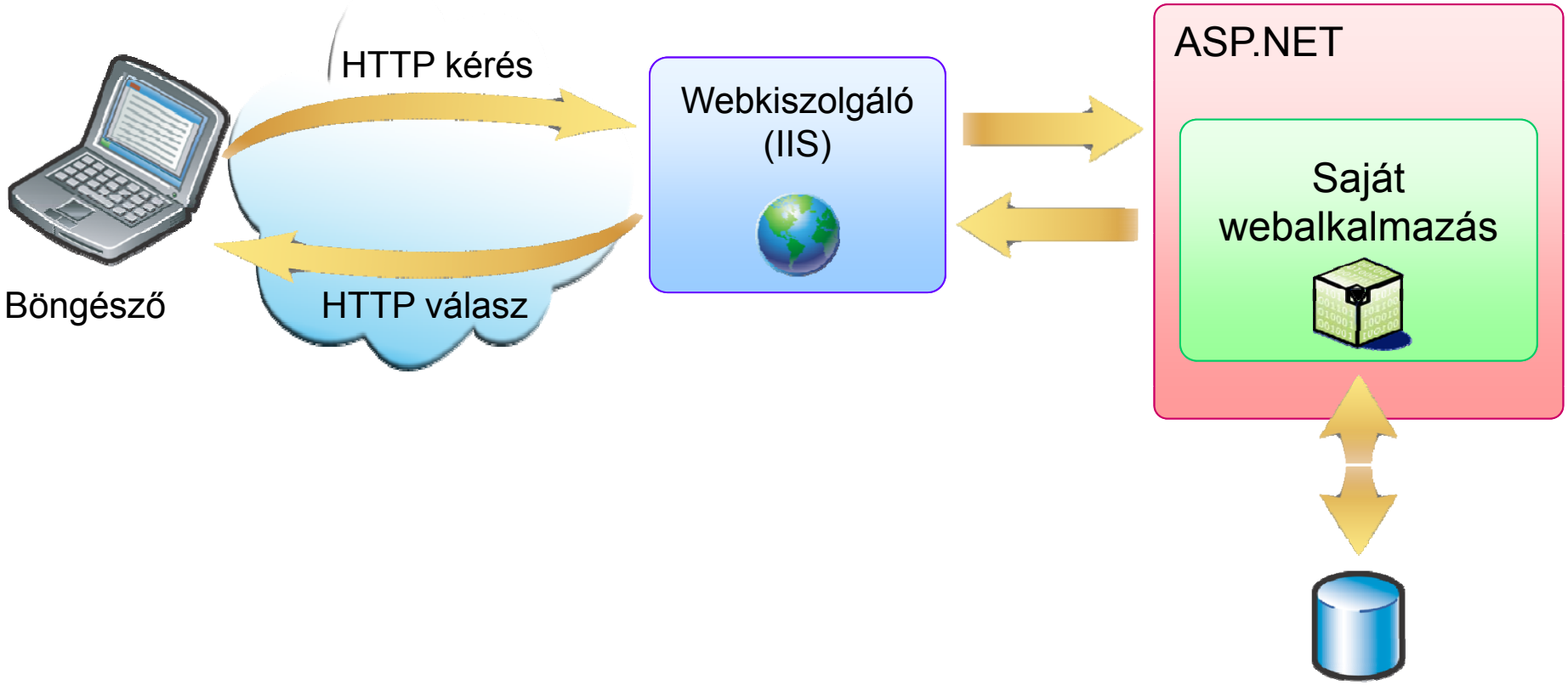
HTML előállítási módszerek

- Statikus weblapok
  - a HTML fájlok a szerveren
- Dinamikus weblapok
  - CGI alkalmazások: teljesen kódolva állítják elő
  - Sablonból szövegrészlet cserével: régi ASP, PHP
  - Sablonból objektummodell építésével: ASP.NET

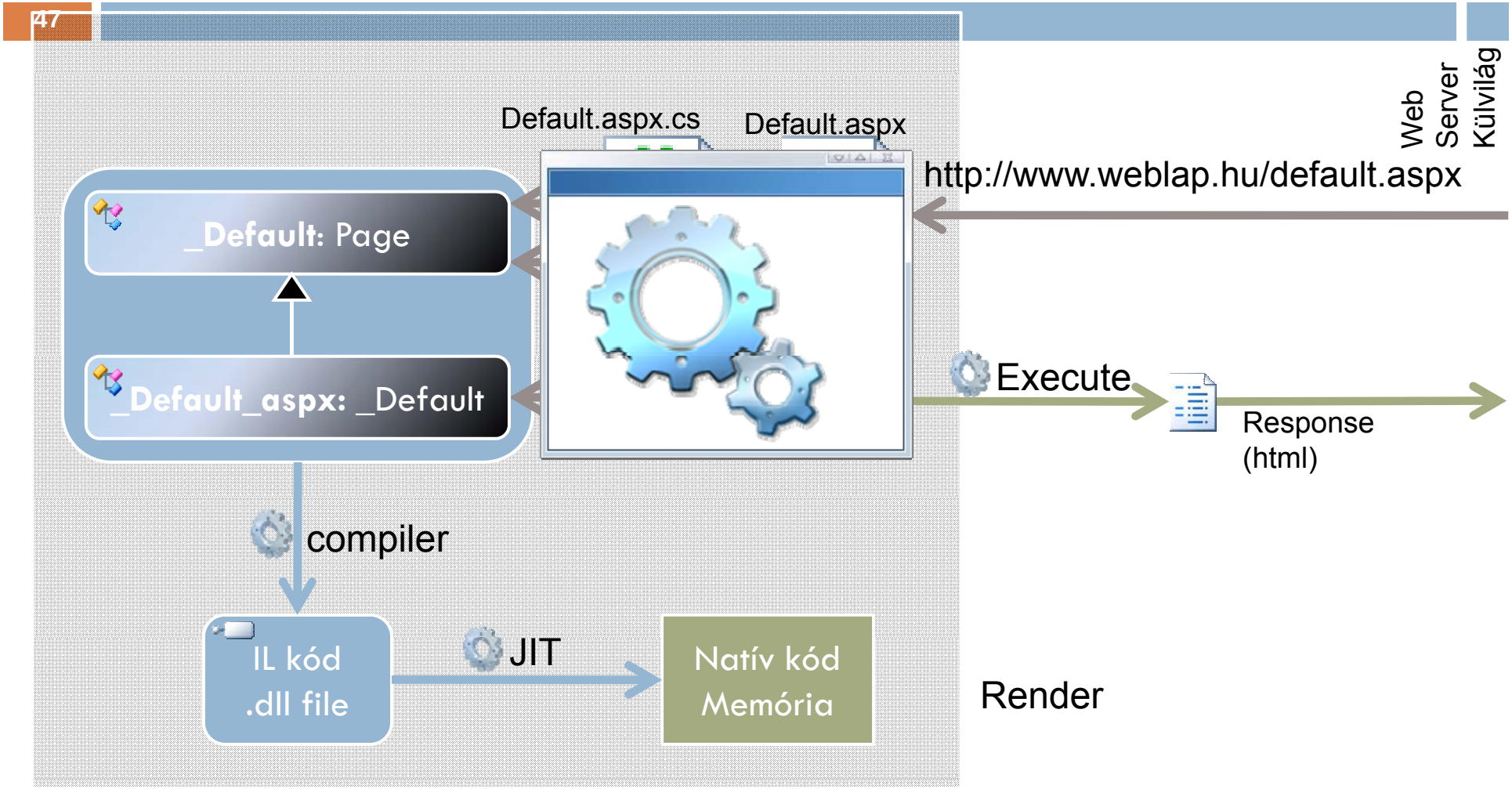
WEB Server vs. File Server



Dinamikus webalkalmazások



Az ASP.Net feldolgozási modellje



Az ASP.Net sablonja (ASPX)

aspx (markup)

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Defaul
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <input id="chk1" type="checkbox" />
        <br />
        <asp:Literal ID="ltr1" runat="server" Text="Literal"/>
        <br />
        <asp:Label ID="lbl1" runat="server" Text="Label"/>
        <br />
        <asp:TextBox ID="txt1" runat="server" Text="TextBox"/>
        <br />
        <asp:Button ID="cmd1" runat="server" Text="Button" />
    </div>
    </form>
</body>
</html>
```

response (html)

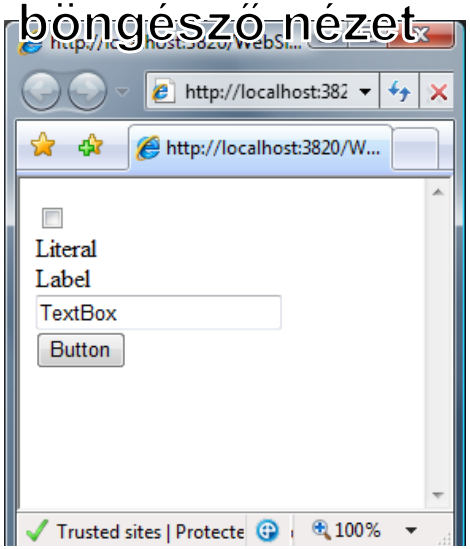
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
</head>
<body>
    <form name="form1" method="post" action="Default.aspx" id="form1">
    <div>
        <input id="chk1" type="checkbox" />
        <br />
        Literal
        <br />
        <span id="lbl1">Label</span>
        <br />
        <input name="txt1" type="text" value="TextBox" id="txt1" />
        <br />
        <input type="submit" name="cmd1" value="Button" id="cmd1" />
    </div>
    </form>
</body>
</html>
```

cs (codebehind)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        cmd1.Text = "Button";
        txt1.Text = "TextBox";
        lbl1.Text = "Label";
        ltr1.Text = "Literal";
    }
}
```

böngésző nézet



Állapotmentes környezet

Szia, hogy hívnak? Szia, hogy hívnak?

- Egy oldal lekérdezése egy egység
- A visszaküldendő html legenerálása után a szerver elfelejt mindent
  - legközelebb nem tudja, hogy a kérdező nem először jár ott, pláne azt hogy előtte mit adott oda
- A probléma megoldása: állapotkezelés (savestate)

Egy oldal életciklusa

Fontosabb mérföldkövek a példányosítástól

- Példányosítás
- Init
- Load
- Események
  - állapokezelési trükk
- Prerender
- Render

PreInit
Init
InitComplete
PreLoad
LoadState
Load
ProcessPostDat
LoadComplete
a
PreRender
PreRenderComplete
SaveState
ProcessPostData Second
SaveStateComplete
Try
Render
Raise ChangedEvents
Raise PostBackEvent

ASP.NET vs. Winforms

- Winforms
  - Futtatáshoz .NET keretrendszer szükséges
  - registry módosítás nélkül települ
  - Alacsony válaszidő
  - Sok előre megírt control
- ASP.NET
  - Futtatáshoz csak böngésző szükséges
  - Nem kell semmit a kliens gépre letölteni
  - Frissítés szükséges minden UI módosításhoz
  - Lassabb (de Rich UI segít)
  - Role-based security

Visual Studio Web site típusok

- File rendszer
  - Bárhol elhelyezhető
  - ASP.NET developement Server
- Helyi IIS
  - IIS root alatt
  - IIS-en fut ezért ha kész a rendszer akkor nem kell külön telepíteni
- Távoli IIS
  - Egy távoli gépen fut
  - Később ez se igényel extra telepítési lépéseket
- FTP
  - Egy távoli írható FTP-re kerülnek a file-ok
  - Általában ugyanezen a gépen lévő IIS futtat

Eseménykezelés

- ASP.NET objektumok eseményeket generálnak
  - Általában van egy default esemény
    - Pl Button click
- Minden esemény két paramétert küld.
  - Sender Object
  - EventArgs
- Egy eseménykezelő képes kiszolgálni több eseményt
  - Sender alapján switch
- AutoEventWireup
  - Pl page_load

Konfiguráció

- XML alapú leíró file-ok
  - Server beállítások – machine.config
  - Root Web beállítások – web.config
  - Web site Beállítások (opc) – web.config
  - ASP.NET webalkalmazás root (opc) – web.config
  - ASP.NET webalkalmazás alkönyvtár (opc) – web.config
- Az effektív jogosultságok ezek egymásra vetítéséből állnak össze
- Szerkeszthető tetszőleges szövegszerkesztővel vagy a Web-site administration tool-al

Konfiguráció - 2.

1. machine.config lekérése
2. Alapértelmezett web.config lekérése
   tulajdonságok hozzávétele az 1 –es halmazhoz.
   Ha van már meglévő tulajdonság és az felülírható akkor megtörténik a felülírás és a web.config jut érvényre.
3. Web site web.config lekérése
4. Webalkalmazás root web.config lekérése
5. Webalkalmazás alkönytár web.config lekérése

Hibakezelés

- Struktúrált hibakezelés
  - Try-catch
- Page-level
  - Váratlan hibák elkapására
  - Page_error event , GetLastError
- Application-level
  - Olyan hibákra amit nem dolgoztunk fel sehol máshol
  - Átirányíthatunk egy default errorpage-re
- Trace
  - elrejtett hibák detektálására
  - erőforrás használat monitorozására

Demo

- Hello World ASP.NET alkalmazás készítése
- Mi történik a háttérben? HTTP header, Firebug

Irodalomjegyzék

- Árvai Zoltán: A Visual Studio 2008 és az ASP.NET 3.5 újdonságai
- Frigó József: Többrétegű architektúrák
- Lippé Szabolcs: Webalkalmazások készítése
- Train4Business 12 órás Webfejlesztő tanfolyam: az ASP.NET programozási modell
- Turóczy Attila: .NET bevezetés