

ASP.NET

Áttekintés

- ▶ Bevezetés
- ▶ Alapok
- ▶ Szerver-oldali vezérlők (kontrollók)
- ▶ ASP.NET oldal életciklusa, eseménymodell

ASP vs. ASP.NET

ASP - Active Server Pages

- ▶ a Microsoft által kifejlesztett szerver-oldali script nyelv
- ▶ a "klasszikus" ASP (utolsó verziója ASP 3.0)
- ▶ egy ASP állomány értelmezett, kiterjesztése: .asp

ASP.NET

- ▶ az ASP.NET a klasszikus ASP továbbfejlesztett változata, de NEM (teljesen) kompatibilis vele
- ▶ A Microsoft .NET platformjának része
- ▶ web alkalmazások készítését lehetővé tevő keretrendszer
- ▶ egy ASP.NET állomány lefordított, kiterjesztése: .aspx

Mindkettő az IIS (Internet Information Services) szerveren fut

ASP.NET előnyök (ASP-hez képest)

- ▶ Több programozási nyelvet támogat
- ▶ Programozható kontrollók
- ▶ Eseményvezérelt programozás
- ▶ XML alapú komponensek
- ▶ Felhasználó azonosítás (felhasználói fiókok, szerepek)
- ▶ Jobb skálázhatóság
- ▶ Hatékonyság (lefordított kód)
- ▶ Egyszerűbb konfigurálás és telepítés

Fejlesztési-/futtatási környezet

- ▶ IIS szerver (Add/Remove Windows components)
- ▶ .NET keretrendszer
- ▶ .NET fejlesztői környezet (Visual Studio... –kereskedelmi vagy ingyenes verzió–, Visual Web Developer (integrált szerver: ASP.NET Development Server), WebMatrix)

hivatalos weboldal:

<http://www.asp.net>

dokumentáció:

- ▶ Dokumentáció a Microsoft oldalán
- ▶ Tutorialok

Webalkalmazás készítését illetően az alábbi háromféle megközelítést támogatja:

- ▶ Web Pages – gyors és egyszerű módja annak, hogy a szerver oldali kódot a HTML elemekkel vegyítve dinamikus tartalmat hozzunk létre
- ▶ **Web Forms** – eseményvezérelt weboldal-készítés
- ▶ ASP.NET MVC – komplexebb webalkalmazások fejlesztésére alkalmas keretrendszer, mely különböző tervezési minták betartására ösztönöz

ASP.NET oldalak jellemzői

Az ASP.NET oldalak (.aspx állomány) dinamikus tartalom létrehozását teszik lehetővé

ASP.NET web oldal feldolgozásának lépései:

- ▶ oldal (első) lekérése (GET metódus): az oldal először fut a szerveren
- ▶ a dinamikusan generált tartalom vissza lesz küldve és megjelenik a kliens böngészőjében
- ▶ a felhasználó pl. információt ír be/kiválaszt vmilyen opciókat... , majd megnyom egy gombot (ha egy másik oldalra mutató hivatkozásra kattint - az oldal feldolgozása ezzel véget ér)
- ▶ a form-adatok a szerverre lesznek "visszapostázva" (*postback*), POST metódussal -tipikusan- ugyanahhoz az oldalhoz
- ▶ a szerveren ismét ugyanaz az ASP.NET oldal fut le – a küldött információ hozzáférhető szerver oldalon
- ▶ a szerveren történt feldolgozást követően a generált tartalom visszamegy és meg lesz jelenítve (*rendering*) a böngészőben

ASP.NET web oldalak jellemzői (folyt.)

- ▶ Ez a ciklus ismétlődik mindaddig, amíg a felhasználó az adott oldallal dolgozik.
- ▶ Egy ilyen ciklus neve: *Round trip* ("körutazás")
- ▶ Lehetőség van arra is, hogy a kérés paramétereit egy másik oldallal dolgoztassuk fel (*cross-page posting*)

oldal élettartama

- ▶ Minden egyes kérés egy újabb oldal (egyúttal egy új Page instancía) létrehozását eredményezi (a cache-eléstől eltekintve)

ASP.NET oldal felépítése:

A felhasználói interfész tervezésére az ASP.NET kétféle modellt kínál:

- ▶ Single-File Page Model – vizuális elemek, illetve eseménykezelők vagy egyéb kód ugyanazon .aspx állományban
- ▶ Code-Behind Page Model – különválasztja a vizuális elemeket, és az ezeket kezelő logikát

Egy ASP.NET oldal az alábbiakat tartalmazhatja:

- ▶ statikus HTML/XHTML
- ▶ ASP.NET direktívák (pl. < % @ Page ... % >)
- ▶ egyetlen form elem (ha vannak szerver kontrolok)
- ▶ ASP.NET szerver-oldali vezérlők (server controls)
- ▶ szerver oldali megjegyzés < % -- megjegyzes -- % >

a Single-File Page Model használata esetén:

- ▶ *script blokk* – script elembe (runat = ‘server’) attribútummal)
- ▶ ágyazott kód – alprogram megadása, pl. eseménykezelő

Példa ASP.NET oldal

```
<%@ Page Language="C#" %>
<html>
<script runat="server">
void Button1_Click(object sender, System.EventArgs e) {
    Label1.Text = ("Üdvözöllek, " + TextBox1.Text);
}
</script>
<head><title>ASP.NET példa</title></head>
<body>
<form runat="server">
    <p>Neved:</p><p>
    <asp:TextBox id="TextBox1" runat="server" />
    <asp:Button id="Button1" runat="server"
        Text="Katt" OnClick="Button1_Click" /> </p><p>
    <asp:Label id="Label1" runat="server"></asp:Label>
    </p>
</form></body></html>
```

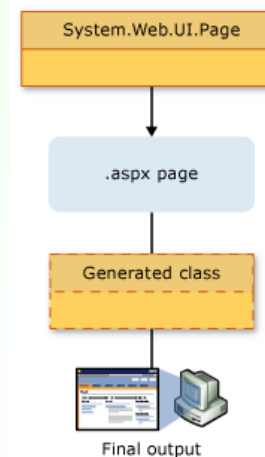
ASP.NET alapok - Code-Behind Page Model

két részre osztott felhasználói interfész (Code-Behind Page Model):

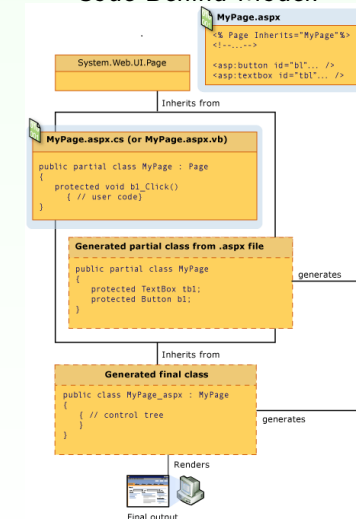
- ▶ vizuális elemek: ASP.NET dokumentum (*markup* file)
- ▶ logika: hozzá tartozó "háttérkód" (*code-behinde* file) – C#, VB.NET, stb.
- ▶ az ASP.NET dokumentum a használt .NET programozási nyelvnek megfelelő osztállyá lesz fordítva (a háttérkóddal együtt)
- ▶ a lefordított kód egy összeállításban (assembly) szerepel, mely a szerverre telepítve futtatható
- ▶ Az ASP.NET dokumentumból generált osztály (*document class*) a System.Web.UI.Page osztály leszármazottja (lásd diagram →)
- ▶ a Page osztálytól örökölt fontosabb adattagok:
 - ▶ Request, Response objektumok (lásd még: belső objektumok)
 - ▶ az oldalon található HTML-, illetve webkontroloknak megfelelő HTMLControls, WebControls osztályok
 - ▶ IsPostBack tulajdonság

Öröklődési modell

Single Page Model:



Code Behind Model:



ASP.NET dokumentum háttérkódjának megadása

ASP.NET web oldal projekt esetén:

```
<%@ Page Language="..." CodeFile="allomanynev.aspx.cs"
Inherits="allomanynev" %>
```

- ▶ telepíthetjük a web oldal forráskódját a szerverre – ezek automatikusan le lesznek fordítva az első (bármelyik oldalra vonatkozó) kérés érkezésekor
- ▶ nem kötelező, de végezhető előfordítás is (hogy a legelső hozzáférés se legyen lassú)

ASP.NET web alkalmazás projekt esetén:

```
<%@ Page Language="..." CodeBehind="allomanynev.aspx.cs"
Inherits="allomanynev" %>
```

- ▶ a web oldalakat explicit módon le kell fordítani telepítés előtt (így tesztelhető is telepítés előtt)
- ▶ egy vagy több összeállítást (assembly-t) telepíthetünk.

Direktívák

- ▶ A direktívák az oldal feldolgozására vonatkozó információt szolgáltatnak (nem lesznek megjelenítve)

a @ Page direktíva segítségével megadható:

- ▶ a programozási nyelv, melyen az oldalhoz tartozó kód íródott
- ▶ a háttérkód elérhetősége (Code-Behind Page Model esetén)
- ▶ debug opciók
- ▶ van-e az oldalhoz rendelt ún mester-oldal (Master Page)

más direktívák:

- ▶ **@ Import** – használni kívánt névterek importálása
- ▶ **@ OutputCache** – oldal cache-elésére vonatkozó információk
- ▶ **@ Implements** – valamilyen .NET interfész implementálása esetén
- ▶ **@ Register** – újabb vezérlők (pl. felhasználó által definiált vezérlők) regisztrálása az oldalon való használat érdekében

ASP.NET szerver-oldali kontrolok/vezérlők

Az ASP.NET az alábbi kategóriákba sorolt vezérlőket kínálja fel:

- ▶ **HTML kontrolok** (HTML controls) – szerver-oldalon programozhatóvá tett (X)HTML elemek
- ▶ **web kontrolok** (web controls) – a HTML vezérlőknél komplexebb funkcionalitású kontrolok
- ▶ **validációs/érvényesség-vizsgáló kontrolok** (validation controls) – lehetővé teszik különböző form-elemekbe bevitt értékek megadott kritériumok szerinti ellenőrzését
- ▶ **felhasználói vezérlők** (user controls) – a felhasználó által készített, újrafelhasználható elemek készítését teszik lehetővé

Az ASP.NET oldalon szereplő vezérlők az oldalnak megfelelő osztályon belül egy fastruktúrában helyezkednek el.

Egyszerű pl.:

- ▶ visszaszamlal_a.aspx, visszaszamlal_b.aspx
- ▶ háttérkóddal: visszaszamlal_c.aspx, visszaszamlal_c.aspx.cs

HTML szerver-oldali vezérlők

- ▶ az egyszerű (X)HTML elemeket az ASP.NET egyszerű szöveggént kezeli
- ▶ bármelyik HTML elem szerver oldalon hozzáférhetővé/programozhatóvá tehető a `runat = "server"` attribútum hozzáadásával
- ▶ a HTML vezérlők a megfelelő (X)HTML elemhez hasonló módon lesznek megjelenítve
- ▶ az `id` attribútumban megadott néven érhető el a megfelelő objektum
- ▶ a HTML vezérlő minden egyes attribútuma elérhető a szerver-oldali objektum tulajdonságaként

pl.

- ▶ `HTMLControlsPI.aspx`

HTML vezérlők – Referencia



17 / 30

Web kontrolok

- ▶ nem feltétlenül feleltethetőek meg egy az egyben egy (X)HTML elemnek
- ▶ komplexebb funkcionalitást biztosítanak a HTML vezérlőkhöz képest
- ▶ a megjelenítésük függhet a böngésző típusától vagy a tulajdonságok beállításától

szintaxis:

```
<asp:ctrlname attributes runat="server" id="UniqueID" />
```

- ▶ `<asp:TextBox runat="server" id="name" />`
- ▶ hivatkozás a beírt szövegre: `name.Text`

Pl.

- ▶ `WebControlsPI.aspx`

web kontrolok – Referencia



18 / 30

Néhány gyakran használt web kontrol:

web kontrol típusa	Neki megfeleltethető (X)HTML elem(ek)
AdRotator	<code></code> és <code><link></code>
Button	<code><input type = "button submit reset" /></code>
Calendar	nincs
Checkbox	<code><input type = "checkbox" /></code>
CheckBoxList	nincs
DropDownList	<code><select></code>
Image	<code></code>
ImageButton	nincs
ImageMap	nincs
Label	nincs
Panel	<code><div></code>
RadioButton	<code><input type = "radio" /></code>
RadioButtonList	nincs
Table	<code><table></code>
TableCell	<code><th></code> , <code><td></code>
TableRow	<code><tr></code>
TextBox	<code><input type = "text" /></code>



19 / 30

Megjegyzés:

- ▶ Egyes vezérlők (pl. `CheckBox`, `CheckBoxList`, `DropDownList`, `TextBox`) rendelkeznek egy `AutoPostBack` tulajdonsággal. Ha ez be van állítva, a vezérlő értékének módosítása automatikusan `postback`-et generál.

Pl.

- ▶ lásd (később) `SessionHandlingPI.aspx`



20 / 30

Vezérlők állapotának megőrzése - ViewState segítségével

View state:

- ▶ az ASP.NET implicit megoldást kínál a vezérlők állapotának (NEM a form elemek értékének) két "postback" közötti megőrzésére
- ▶ egyéb oldal-specifikus adatok ideiglenes lementésére is használhatjuk
- ▶ a dokumentum osztály rendelkezik egy ViewState nevű StateBag típusú adattaggal, melybe a vezérlők állapota le lesz mentve
- ▶ a válasz generálása során a ViewState szerializálva lesz, illetve el lesz küldve a kliens böngészőre egy __VIEWSTATE nevű hidden mezőként

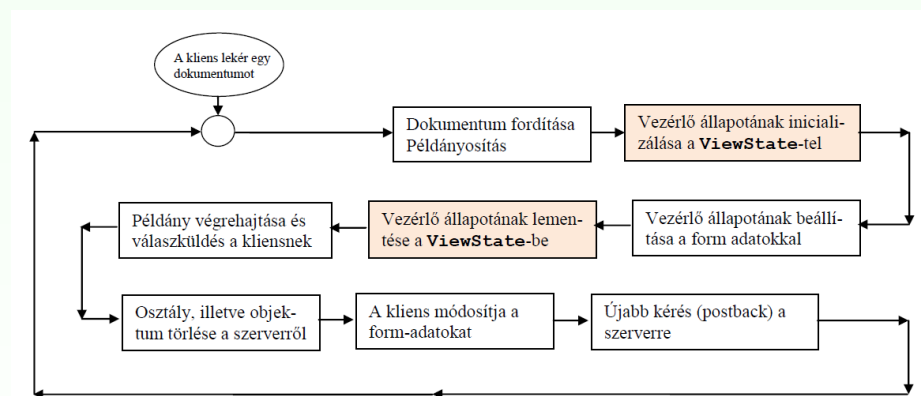
a view state használata kikapcsolható, ha szükséges

ViewStateMode =" Disabled" vagy EnableViewState=" false"

- ▶ oldal szinten (a @ Page direktívában)
- ▶ vezérlő szinten

view state

ASP.NET dokumentum életciklusa:



Pl.

- ▶ lásd: ViewStatePl.aspx

Érvényesség-vizsgáló kontrolok

- ▶ a felhasználó által bevitt adatok ellenőrzését teszik lehetővé
- ▶ hiba esetén a megfelelő hibaüzenet megjelenítése testreszabható
- ▶ HTML- illetve web kontrolokkal egyaránt használhatóak

Különböző típusú érvényesség-vizsgáló vezérlők:

- ▶ **RequiredFieldValidator** – biztosítja, hogy a kötelező bemenetet mindenképpen kitöltse a felhasználó
- ▶ **CompareValidator** – összehasonlítja végez egy adott értékkel
- ▶ **RangeValidator** – megvizsgálja, hogy az érték egy megadott intervallumban van
- ▶ **RegularExpressionValidator** – megvizsgálja, hogy a érték megfelel-e egy megadott reguláris kifejezésnek (pl. e-mailcím, telefonszám ellenőrzése)
- ▶ **CustomValidator** – felhasználó által definiált funkcionalitás az ellenőrzés végezhető kliens és/vagy szerver oldalon

- ▶ **ValidationSummary** – az érvényesség-vizsgálat során észlelt hibák összesítésére ad lehetőséget

Megjegyzés:

Az ASP.NET automatikusan ellenőrzi az esetlegesen "rosszindulatú" bemenetet (pl. HTML elemek, script kód), akkor is, ha nem használunk külön érvényesség-vizsgáló vezérlőket

Pl.

- ▶ ErvenyessegEllenorzesPl.aspx

Validation controls – Referencia

ASP.NET oldal életciklusa

Egy ASP.NET oldal életciklusa során különböző fázisokon megy keresztül:

- ▶ inicializálás, vezérlők példányosítása, állapot megőrzés/visszaállítás, eseménykezelők futtatása, kimenet generálása (rendering)

Fázisok:

- ▶ Kérés érkezik – Az ASP.NET eldönti, hogy szükség van-e az oldal feldolgozására vagy visszaküldhető a cache-ben tárolt változat
- ▶ Start – Request, Response, IsPostBack, illetve UICulture beállítása
- ▶ Inicializálás – az oldalon szereplő vezérlők elérhetővé válnak (de az állapotuk még nincs a view state alapján beállítva)
- ▶ Load – postback kérés esetén a vezérlők állapota inicializálva lesz a view state alapján
- ▶ folyt. ...

ASP.NET oldal életciklusa (folyt.)

Fázisok (folyt.)

- ▶ Postback esemény-kezelés – postback esetén meghívódnak a vezérlők megfelelő eseménykezelői, majd az érvényesség-vizsgáló vezérlők érvényességvizsgáló metódusai (megj. az érvényességvizsgálatot kiváltó esemény eseménykezelője csak ezt követően lesz meghívva)
- ▶ Kimenetgenerálás (rendering) – előtte az összes vezérlő esetén le lesz mentve a view state. Meghívódik rekurzívan a vezérlők Render metódusa
- ▶ Unload – miután a válasz el lett küldve a kliensnek, a már nem szükséges adatstruktúrák törlésére kerül sor

ASP.NET szerver-oldali vezérlők esemény modellje

esemény-modell:

- ▶ az ASP.NET egy esemény-alapú modellt kínál a dokumentumok szerver-oldali programozására
- ▶ az esemény kliens oldalon váltódik ki
- ▶ az esemény kezelése szerver-oldalon történik
- ▶ a különböző vezérlőkhöz kapcsolódó, szerver-oldalon kezelhető tipikus események:
 - ▶ kattintás – automatikus postback
 - ▶ módosítás (csak egyes vezérlőknél) – hatásuk a következő postback-ig késleltetve van/ vagy azonnali postback (lásd: AutoPostBack tulajdonság)
- ▶ a vezérlők és maga az oldal is generálhat ún. *életciklus-eseményeket*: Init, Load, PreRender

Eseménykezelés

Eseménykezelők

- ▶ két paraméter: az eseményt generáló objektum (**sender**), esemény objektum (az eseménnyel kapcsolatos információk – **e**)
- ▶ oldalhoz kapcsolódó események esetén automatikus esemény-eseménykezelő hozzárendelés (a @ Page direktíva AutoEventWireup attribútuma be kell legyen állítva) névkonvenció alapján: Page_ *esemény*
- ▶ vezérlőhöz kapcsolódó eseménykezelő megadása (pl.):
<asp:TextBox ID="textbox1" Runat="server"
eseménykezelő:
 OnTextChanged="NameChange" />
protected void NameChange(object **sender**, EventArgs **e**)
{ //...kód }

