

2. Platforma Microsoft .NET

Maciej Piechówka

macpi@eti.pg.gda.pl

2008

Spis treści:

2.1 Microsoft .NET Framework: CLR, podzespoły, biblioteka klas

2.2 Dostęp do danych za pomocą ADO.NET

- Użycie języka XML w połączeniu z ASP.NET, LINQ

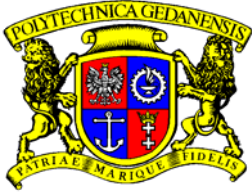
2.3 Wytwarzanie aplikacji .NET: podstawy ASP.NET

- HTTP - przypomnienie
- ASP.NET od wewnątrz
- Model strony, WebForms
- Model delegacyjny zdarzeń
- Kontrolki serwerowe, użytkownika, sprawdzające
- Wiązanie danych
- Zarządzanie stanem

2.4 Usługi Web

Literatura:

- Materiały firmy Microsoft, zasoby Internetu...
- Platt D. *Podstawy Microsoft .Net*, RM, 2001 ;
- S.C. Perry, *Core C# i NET*, Helion 2006
- Esposito D. *Tworzenia aplikacji za pomocą ASP.NET oraz ADO.NET*, RM 2002
- Connolly R. *ASP.NET 2.0. Projektowanie aplikacji internetowych*, Helion 2008

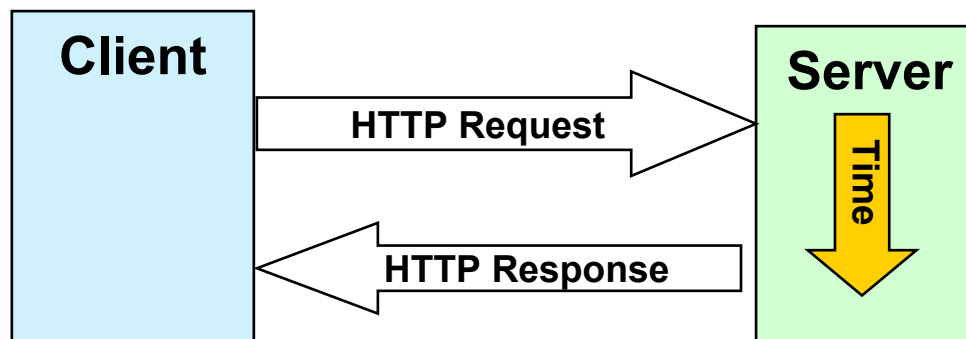
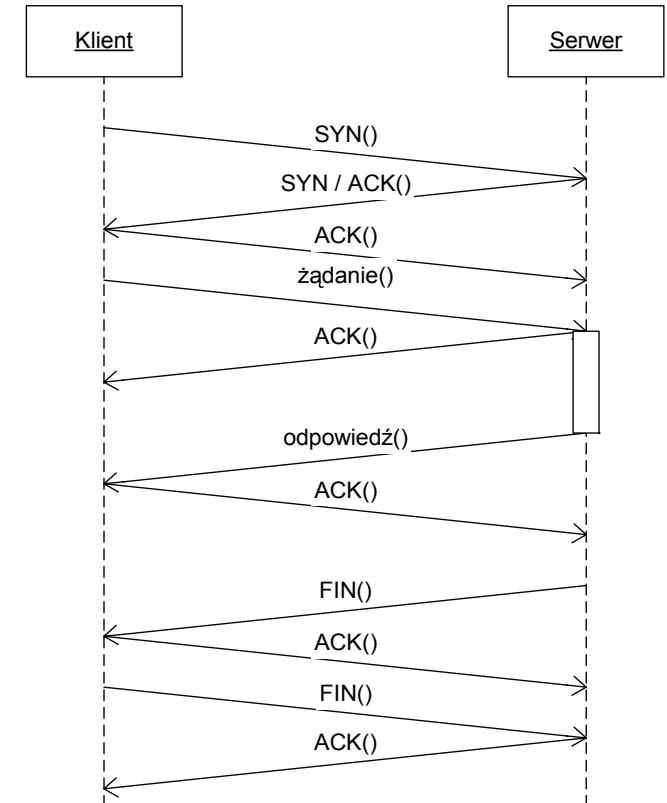


2.3.0 Protokół HTTP - przypomnienie

- Protokół zaprojektowany jako **bezstanowy** (brak pojęcia sesji grupującej interakcje). Interakcja przeglądarki z serwerem WWW odbywa się według schematu **żądanie - odpowiedź**:
 - Serwer nadsluchuje żądania,
 - Klient otwiera połączenie – serwer odpowiada potwierdzeniem,
 - **Żądanie** HTTP jest wysyłane przez klienta,
 - Serwer przekazuje w **odpowiedzi** żądane zasoby lub informację o ich niedostępności,
 - Połączenie zostaje zamknięte przez serwer.
- Protokół określa format komunikatu żądania oraz odpowiedzi.
- Domyślny numer portu: 80.

Wersje

- HTTP/1.0
- HTTP/1.1 (RFC 2068,2616)





Struktura komunikatu HTTP

Wiersz początkowy **żądania**:

Wiersz początkowy **odpowiedzi**:

- nazwa metody
- ścieżka do zasobu
- używana wersja protokołu

Request Line

- Method
- Request URI
- HTTP Version Info

Response Line

- HTTP Version Info
- Status Code
- Description

- wersja protokołu
- kod rezultatu (liczba)
- opis w języku naturalnym

Headers

CRLF

Message Body

Treść żądania

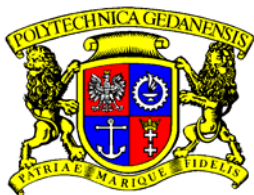
Formaty żądania i odpowiedzi są podobne:

- wiersz początkowy (zależny od typu komunikatu),
- dowolna liczba wierszy nagłówek,
 - typy nagłówek: General, Request, Response, Entity
- pusta linia (CRLF) – dla zaznaczenia końca sekcji nagłówkowej,
- opcjonalne ciało komunikatu,



Nagłówki żądań protokołu HTTP 1.1

Accept	typy MIME, które przeglądarka jest w stanie obsługiwać
Accept-Encoding	rodzaje kodowania (np.: gzip lub compress) jakie przeglądarka jest w stanie obsługiwać
Authorization	identyfikacja użytkownika wykorzystywana przez zasoby, do których dostęp jest chroniony hasłem. Zazwyczaj stosowana metoda przesyłania informacji o nazwie użytkownika i hasła polega nie na wykorzystaniu mechanizmów protokołu HTTP lecz zwykłych formularzy HTML
Connection	w przypadku protokołu HTTP 1.0 wartość keep-alive tego nagłówka oznacza, że przeglądarka jest w stanie obsługiwać trwałe połączenia. W protokole HTTP 1.1 trwałe połączenia są wykorzystywane domyślnie
Cookie	cookies przesyłane wcześniej z serwera do klienta
Host	nazwa komputera podana w oryginalnym adresie URL. W protokole HTTP 1.1 nagłówek ten jest wymagany
If-Modified-Since	określa, że klient chce pobrać stronę wyłącznie jeśli została ona zmodyfikowana po określonej dacie
Referer	adres URL strony, która była wyświetlona w przeglądarce w chwili, gdy wysyłano żądanie
User-Agent	łańcuch znaków identyfikujący przeglądarkę, która przesyłała żądanie



Elementy komunikatu odpowiedzi HTTP

• Kody statusu HTTP

1xx	kody informacyjne, klient powinien odpowiedzieć na nie wykonując jakąś czynność,
2xx	żądanie zostało poprawnie obsłużone
3xx	plik został przeniesiony; w takim przypadku odpowiedź zazwyczaj zawiera nagłówek Location określający nowe położenie pliku
4xx	błąd klienta (np. 400 - nieprawidłowe zapytanie, 403 - dostęp do zasobu zabroniony, 404 - zasób nie znaleziony)
5xx	błąd serwera

HTTP/1.1 200 OK

Content-Type: text/plain

...

Last-Modified: Sun, 24 Dec 2004 22:38:36 GMT

...

• Nagłówki odpowiedzi protokołu HTTP

Content-Encoding	określa sposób kodowania dokumentu
Content-Length	ilość bajtów przesyłanych w odpowiedzi
Content-Type	typ MIME zwracanego dokumentu
Expires	czas, po którym dokument należy uznać za nieaktualny i usunąć z pamięci podręcznej przeglądarki
Last-Modified	czas ostatniej modyfikacji dokumentu
Location	adres URL pod który przeglądarka powinna przesłać kolejne żądanie
Refresh	ilość sekund, po upływie których przeglądarka powinna ponownie odświeżyć stronę. Nagłówek może także zawierać adres URL strony, którą przeglądarka ma pobrać
Set-Cookie	cookie, które przeglądarka powinna zapamiętać
WWW-Authenticate	typ oraz obszar autoryzacji jaki przeglądarka powinna podać w nagłówku Authorization przesyłanym w kolejnym żądaniu
Server	rodzaj oprogramowania serwera (analogicznie jak User-Agent)

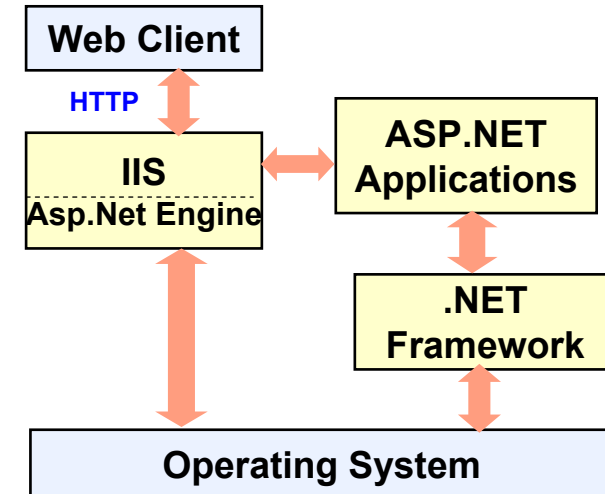


2.3 ASP.NET Web Applications

- **Obiektowe środowisko** projektowania dynamicznych aplikacji WWW
- **Rozdzielenie** projektowania prezentacji strony od kodu logiki biznesowej (logika strony)
 - Użycie komponentów dostarczanych przez .NET Framework - kontrolki Web, HTML działające po stronie serwera.
 - Kod strony jest kompilowany przy pierwszym żądaniu strony ASP.NET (.aspx)
 - Możliwość użycia języków programowania VB.NET, C#
- **Organizacja UI** - strony główne i strony z treścią
- **Zarządzanie stanem, bezpieczeństwem**

Aplikacja ASP.NET może składać się elementów:

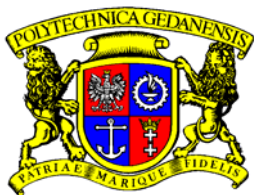
- Formularze Internetowe (Web Form) – pliki z rozszerzeniem (.aspx)
- Usługi Web (Web services) – pliki z rozszerzeniem (.asmx)
- Pliki logiki aplikacji – pliki z rozszerzeniem (.vb lub .cs)
- Globalnej klasy aplikacji (.asax)
- Pliku konfiguracyjnego Web.config
- innych pliki typu: strona HTML, arkusz CSS,



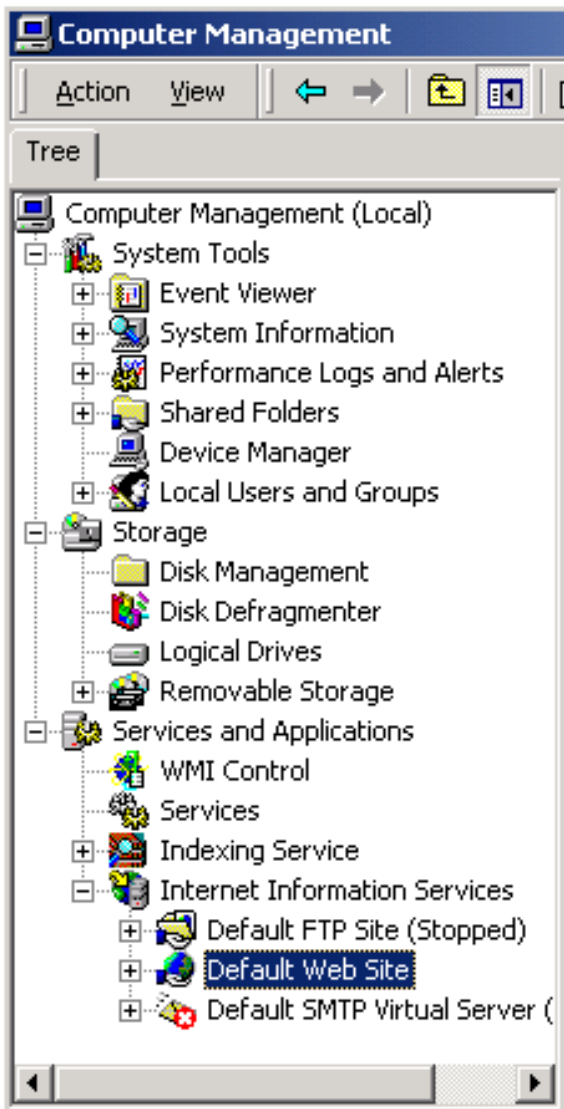
- Lokalizowanie zasobów – **URI**
- Funkcjonowanie żądań i odpowiedzi - **HTTP**
- Przedstawianie informacji i poruszanie się między zasobami - **HTML**

ASP.NET – model programowania

- **Web Forms**
- **Web Controls**
- **Event Handling**
- **Validators**
- **User Controls**
- **State Management**
- **Configuration of ASP.NET**

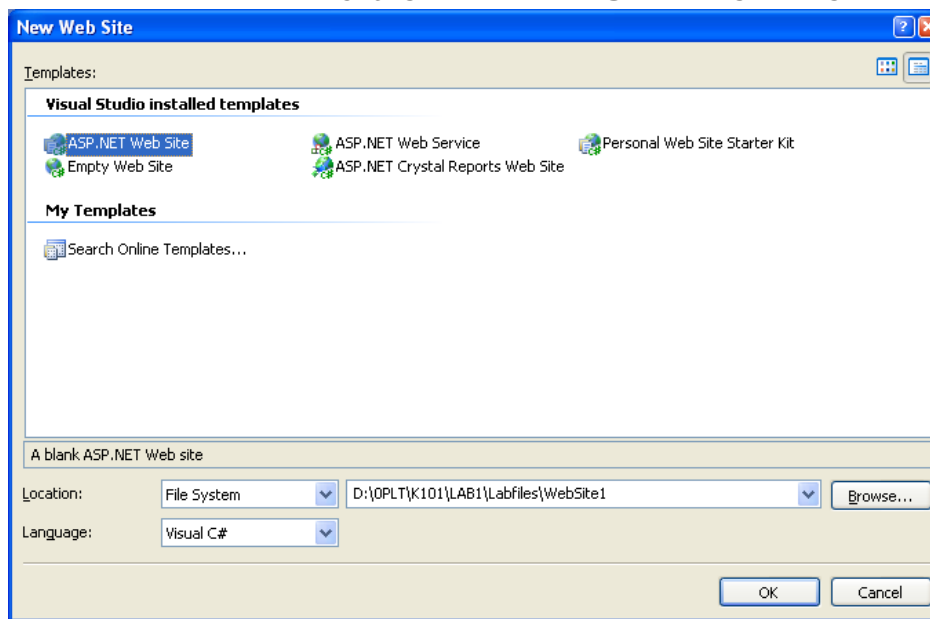


Tworzenie aplikacji ASP.NET



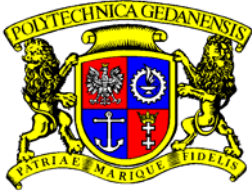
Visual Studio .NET – tworzenie nowej witryny

- Utworzenie i skonfigurowania katalogu wirtualnego (*virtual directory*) na serwerze IIS
- Utworzenie plików źródłowych i projektowych w odpowiadającym katalogu „fizycznym”



- W systemie plików
- Na lokalnym serwerze IIS
- Na zdalnym serwerze IS
- Na serwerze IIS z systemem plików dostępnym przez FTP

Aplikacja ASP.NET to: suma wszystkich plików, stron, procedur obsługi zdarzeń, modułów, kodu wykonywalnego (programów i bibliotek), wykonywanego lub uruchamianego w obrębie danego **katalogu wirtualnego** (i jego podkatalogów) na serwerze WWW



2.3.1 Formularz WEB (Web Form)

- **Rozdzielnie kodu od zawartości interfejsu użytkownika**

Single file

Separate files



Form1.aspx

Form1.aspx Form1.aspx.vb
or Form1.aspx.cs

Kod schowany strony (Code-Behind Pages)
– VS 2003
Code-Separation Model in Visual Studio
.NET 2005

Sekcja dyrektyw strony

- Konfigurują środowisko, w którym będzie pracowała strona.
- Określają sposób przetwarzania strony przez moduł wykonawczy HTTP.
- Umożliwiają importowanie przestrzeni nazw, ładowanie podzespołów, których nie ma w danym momencie w GAC, rejestrowanie nowych kontrolki z niestandardowymi nazwami tagów i prefiksami przestrzeni nazw.

Sekcja kodu

- Opatrywana tagiem `<script>` zawiera kod związany z daną stroną. Zawiera zwykle procedury obsługi zdarzeń i funkcje pomocnicze. Kod aplikacji może zostać umieszczony bezpośrednio w pliku .aspx tzw. *Code Inline* lub w dodatkowym pliku tzw. *Code Behind*.

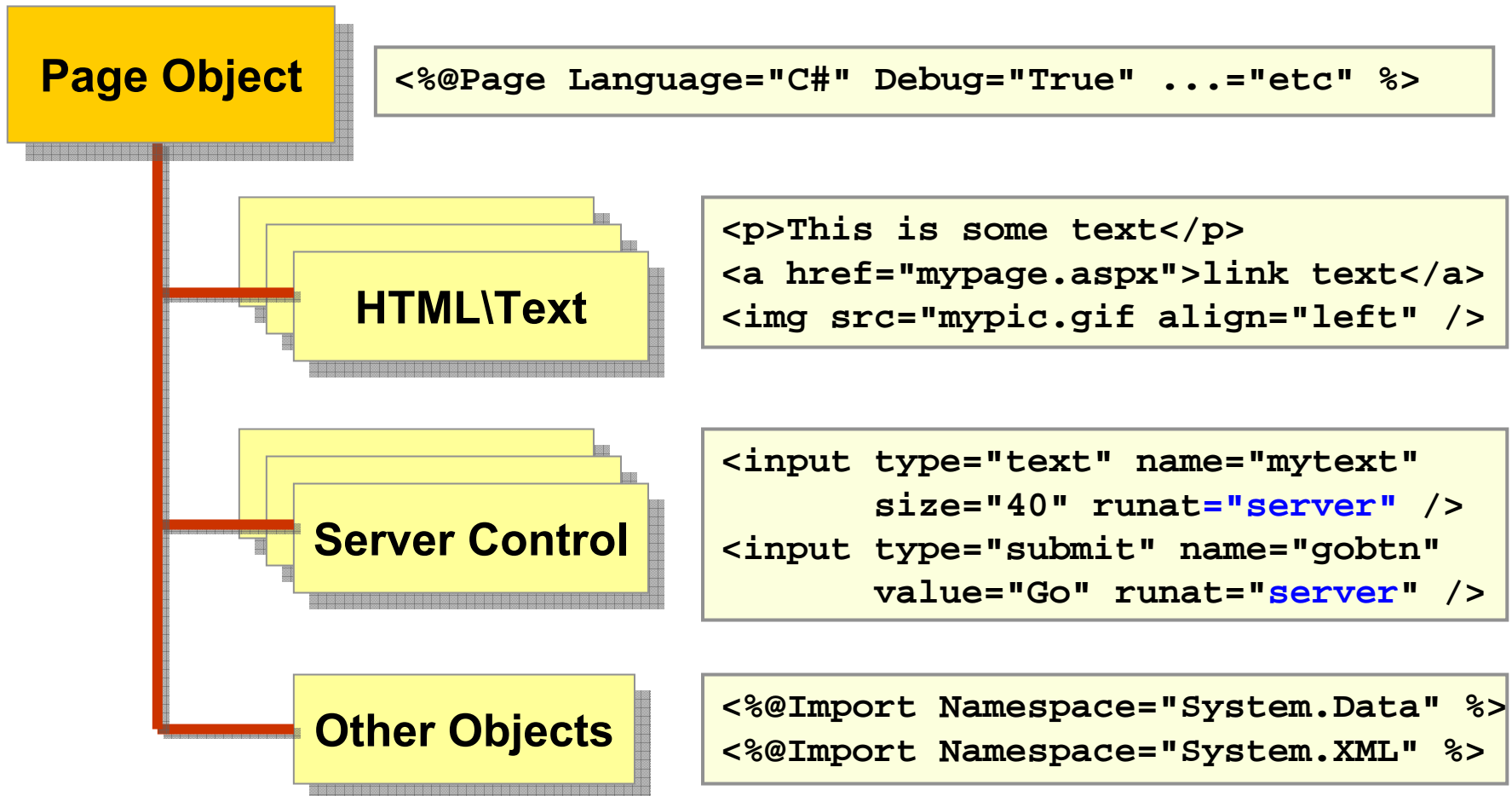
Sekcja układu strony (*page layout*)

- Zawiera reprezentację widoku strony w postaci zbioru kontrolki serwerowych, tekstu oraz znaczników HTML, który jest uszczegóławiany przez kod.



Model formularza/strony ASP.NET

- Strona ASP.NET jest drzewem obiektów





Model programowania – formularz Web Form

Components of Web Forms

- **Visual Component**
 - Works as a container for text and controls on a page *.aspx
- **User Interface Logic**
 - Consists of code that interacts with the form *.aspx.cs

Pages

- **Server controls**
 - Encapsulate UI generation, user interaction
 - Fire events for state changes

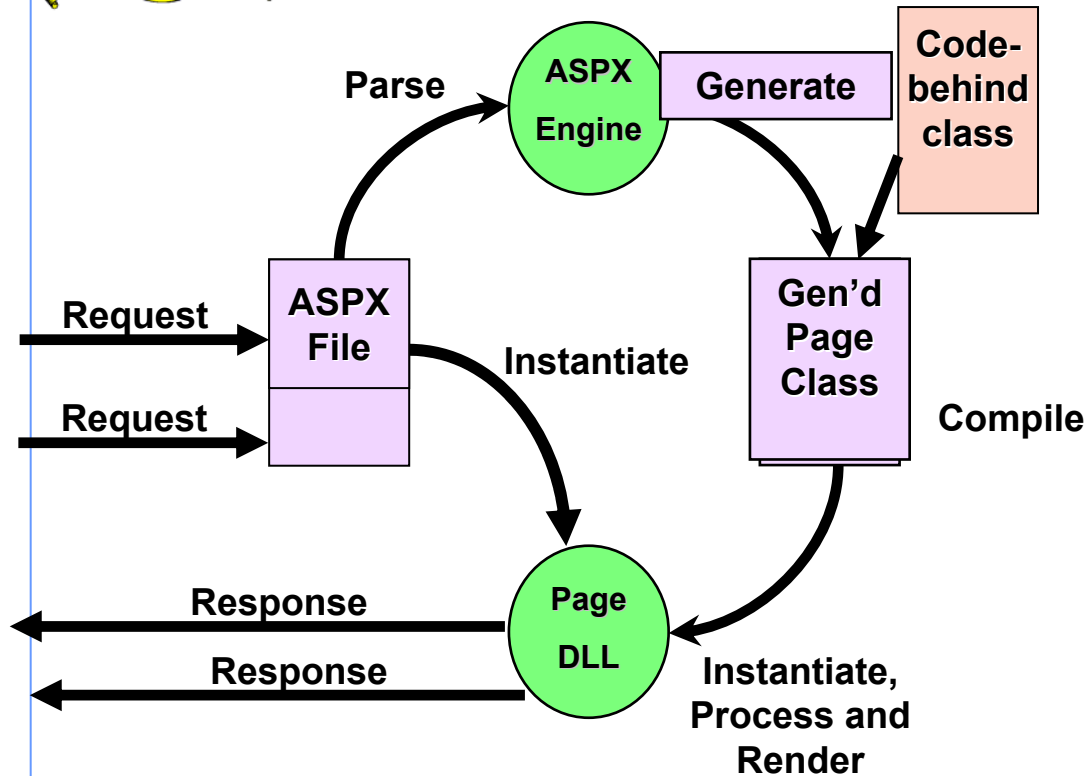
- **Page execution:**
 - Page fires events for phases of page processing
 - Init, Load, Render, Unload, etc
- **Event handler code**
 - Handles events raised by controls, page
 - Can be located in-line, or in separate file or DLL

Server Controls

- **Server controls encapsulate behavior**
 - Declarative, tag with `runat="server"`
- **Generate HTML** that is sent to the client
 - Can support multiple client types
 - DHTML, HTML 3.2, WML, etc.
- **Process input sent from client**
 - **Bind** to data in Forms collection
 - **Fire events** for notifications

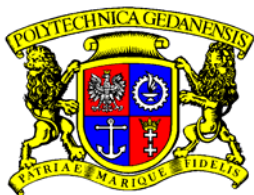


Cykl życia strony (2)



- Strona jest **obiektem** klasy ***System.Web.UI.Page***; mamy dostęp do jej metod i właściwości
- Elementy GUI są obiektami ***System.Web.UI.WebControls***; mamy dostęp do metod i właściwości kontrolki
- Strona Web ma dostęp do wszystkich klas .NET library

- Przeglądarka użytkownika odwołuje się do pliku o rozszerzeniu .aspx
- ASP.NET odczytuje plik z systemu plików serwera
- ASP.NET przegląda wszystkie znaczniki w pliku i ładuje je do pamięci
 - jeśli znacznik zawiera atrybut *runat=„server”*, ASP.NET ładuje odpowiednią kontrolkę serwerową. Typ kontrolki jest określony przez nazwę znacznika.
 - Znaczniki niezawierające atrybutu *runat=„server”* stanowią zwykły kod HTML. ASP.NET w niezmienionej postaci przekaże je do odbiorcy
- Po załadowaniu wszystkich znaczników do pamięci ASP.NET wykonuje odpowiedni kod programu każdej z kontrolki serwerowych.
- Po zakończeniu przetwarzania kodu **wszystkich kontrolki serwerowych**, ASP.Net wywołuje metodę ***Render*** każdej kontrolki
- Po wygenerowaniu strony ASP.NET **uwalnia pamięć**



Dyrektywy strony ASP.NET (1)

Dyrektywy strony:

Składnia: <%@ dyrektywa atrybut="wartość" [, atrybut=wartość] %>

@ Page - definiuje atrybuty strony wykorzystywane przez kompilator stron. Umożliwia określenie parametrów protokołu HTTP, określenie przestrzeni nazw, definicję języka programowania.

Atrybuty (przykłady):

Buffer - definiuje czy buforować odpowiedzi HTTP. Jeśli true - buforowanie ma być dostępne.

EnableViewState - wskazuje, czy informacja o właściwościach strony ma być przechowywana pomiędzy żądaniami strony

ErrorMessage - definiuje docelowy URL dla przekierowania, jeśli wystąpi błąd

Inherits - zewnętrzna klasa (nazwa klasy kodu schowanego), po której strona dziedziczy

Language - język stosowany do kompilacji wszystkich bloków wewnątrz strony

Src - adres URL pliku źródłowego definiującego zewnętrzną klasę

Trace - wskazuje, czy śledzenie jest włączone

MasterPageFile - Specifies the path of the master to use for building the current page

SmartNavigation - odświeżanie tylko tych części formularza które się zmieniły

Theme - Specifies the name of the theme to use for the page

...

@ Control - definiuje atrybuty kontrolki użytkownika (*user control*)

@ Register - tworzy powiązanie pomiędzy nazwą pliku kontrolki użytkownika a nazwą odpowiadającego jej znacznika.



Formularz WEB (Web Form) - strona prezentacyjna ASP.NET: widok HTML

- **.aspx** extension
- **Page** attributes
 - @ Page directive
- **Body** attributes
- **Form** attributes

WebForm1.aspx

```
<%@ Page Language="c#" Codebehind="WebForm1.aspx.cs"
    SmartNavigation="true"%>
<html>
  <body ms_positioning="GridLayout">
    <form id="Form1" method="post" runat="server">
      ...
    </form>
  </body>
</html>
```



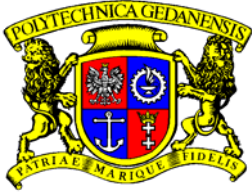
Formularz WEB (Web Form)

Logika strony - plik kodu schowanego (*code behind*)

[WebForm1.aspx.cs](#)

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace WebApplication1
{
    /// <summary>
    /// Summary description for WebForm1.
    /// </summary>
    partial class WebForm1 : System.Web.UI.Page
    {
        private void Page_Load(object sender, System.EventArgs e)
        {
            // Put user code to initialize the page here
        }
        #region Web Form Designer generated code
    }
}
```



Strona ASP.NET – dziedziczy po klasie Page

System.Web.UI.Page

```
class Page : TemplateControl, IHttpHandler
{
    // State management
    public HttpSessionState Application {get;}
    public HttpSessionState Session {virtual get;}
    public Cache Cache {get;}
    // Intrinsic
    public HttpRequest Request {get;}
    public HttpResponse Response {get;}
    public HttpServerUtility Server {get;}
    // Client information
    public string ClientTarget {get; set;}
    public IPPrincipal User {get;}
    //...
    public virtual ControlCollection Controls {get;}
    public bool IsPostBack {get;}
    //...
```

Application and **Session** - application state and session state

Request and **Response** – gets the **HttpRequest/HttpResponse** object for the requested page

Controls – object that represents the child controls

IsPostBack - true, if the page was sent to the server in a round trip. If the page was requested for the first time `IsPostBack == false`

```
public UserControl LoadControl (string virtualPath);
public override string ID { get; set; }
protected virtual void RenderControl (HtmlTextWriter writer);
```

RenderControl - outputs server control content to a provided `HtmlTextWriter` object and stores tracing information about the control if tracing is enabled.



Klasa Page (2)

```
// properties
public ValidatorCollection Validators {get;}
public bool IsValid { get; }
public virtual string TemplateSourceDirectory {get;}
...
// methods
public virtual void Validate();
public string MapPath(string virtualPath);

// Events
public event EventHandler Init;
public event EventHandler Load;
public event EventHandler PreRender;
public event EventHandler Unload;
//...
}
```

IsValid – true, if none of the validators on the page reported an error

TemplateSourceDirectory -
current virtual directory

Validate()
starts all validators on the page

MapPath(virtPath) – maps the
virtual directory to the physical one

Zdarzenia strony



2.3.2 Elementy formularza WebForm: Klasyfikacja kontrolek serwerowych ASP.NET

- **Kontrolki serwerowe HTML** (*HTML server Control*) - z atrybutem `runat=„server”`
- **Kontrolki serwerowe Web** (*Web Server Control*) –
`<asp:xx ...runat=„server”</asp:xx>`
 - Standardowe – etykiety, pola tekstowe, listy, ...
 - Data – odczyt informacji z pojemników danych
 - Navigation – kontrolki wyświetlające elementy nawigacyjne, takie jak ścieżki, menu różnych typów
 - Login – udostępniające funkcje sterowania dostępem, rejestracji użytkownika
 - Walidacji danych
 - WebParts – umożliwiające wydzielanie części strony Web jako obszaru dynamicznego, który autoryzowani użytkownicy mogą dostosować do swoich preferencji
- **Kontrolki użytkownika** (*Web User Controls*)
- **Wbudowane kontrolki Web** (*WebCustom Controls*)



Elementy WebForm: Kontrolki serwerowe: Server Control

```
<asp:Button id="Button1" runat="server" Text="Submit"/>
```

Namespace
reference

Class to
create

ID of
instance

Kontrolki Web są zdefiniowane w przestrzeni nazw
System.Web.UI.WebControls

- **Runat="server"**
 - Events happen on the server
 - View state saved
- **Have built-in functionality**
- **Common object model**
 - All have **Id** and **Text** attributes (can be referenced within server-side code using designated Id)
- **Create browser-specific HTML and provide rendering to the client as HTML**
- **Implicitly added as member variables to the generated Page-derived class definition**

Control

WebControl

Button

TextBox

Label

BaseValidator

...

CheckBox

RadioButton

ListControl

ListBox

DropDownList

Image

ImageButton

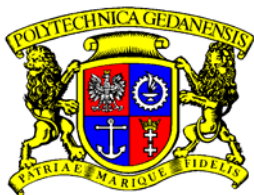
Calendar

ValidationSummary

...

TemplateControl

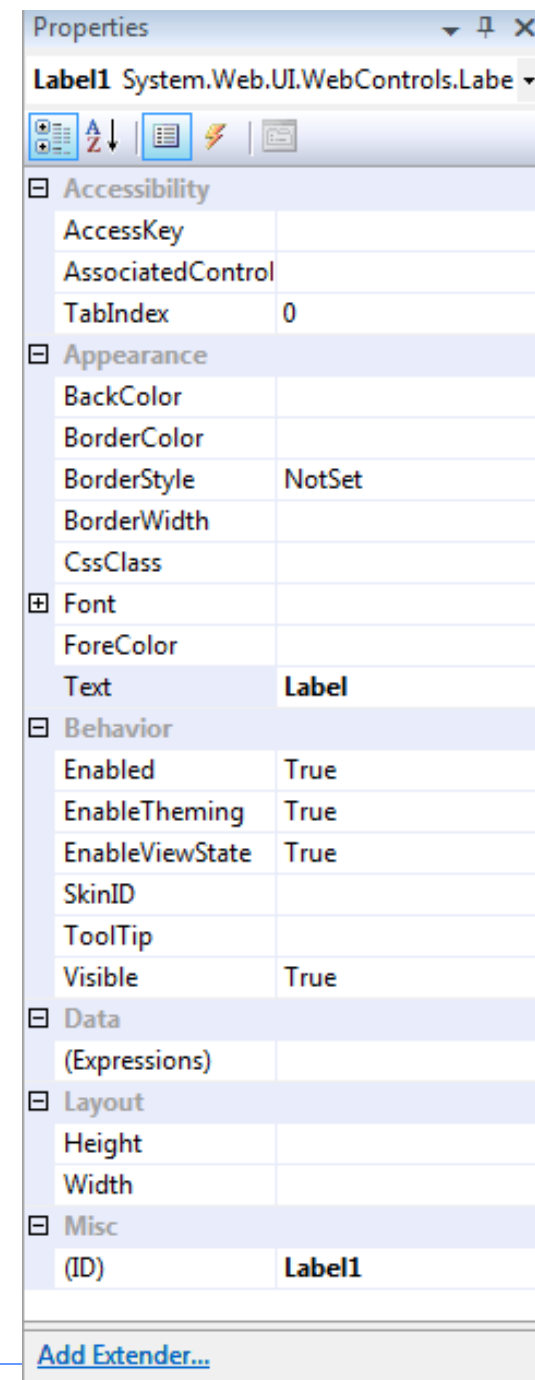
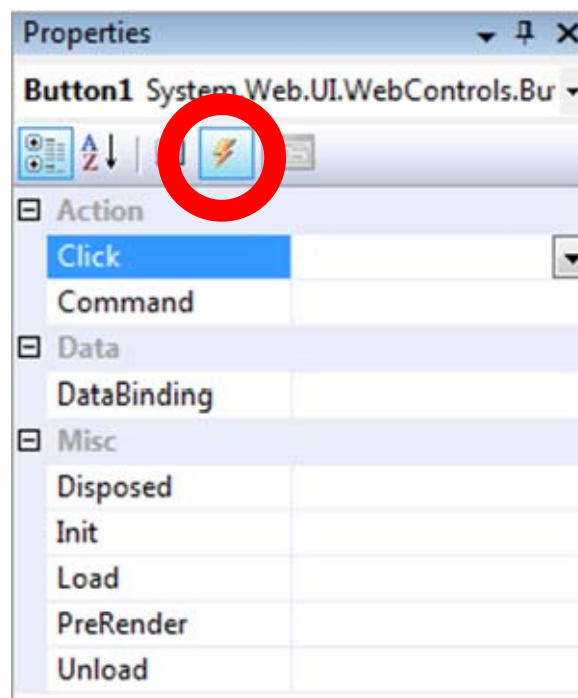
UserControl



Kontrolki na stronie

Każda kontrolka posiada zbiór właściwości, które możemy określić w oknie Properties

Interakcje użytkownika z kontrolkami generują zdarzenia, obsługiwane przez powiązane z nimi metody ich obsługi (event handler)





Przykład Web Forms (1) - trzy reprezentacje

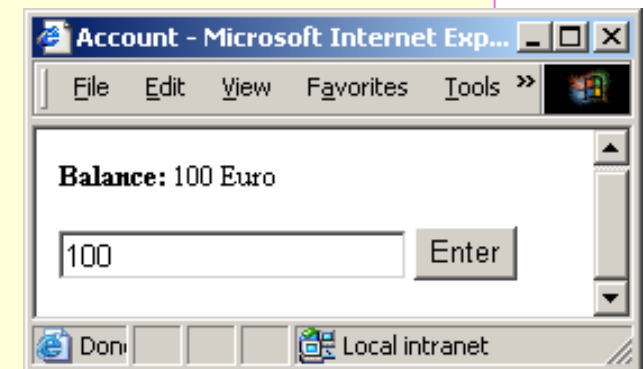
```
<%@ Page Language="C#" Inherits="Adder" Src="Adder.aspx.cs"%>
<html>
  <head><title>Account</title></head>
  <body>
    <form method="post" Runat="server">
      <b>Balance:</b>
      <asp:Label ID="total" Text="0" Runat="server"/> Euro<br><br>
      <asp:TextBox ID="amount" Runat="server"/>
      <asp:Button ID="ok" Text="Enter" OnClick="ButtonClick" Runat="server" />
    </form>
  </body>
</html>
```

Adder.aspx

```
using System; using System.Web.UI; using System.Web.UI.WebControls;
public class Adder : Page
{
  protected Label total;
  protected TextBox amount;
  protected Button ok;

  public void ButtonClick (object sender, EventArgs e) {
    int totalVal = Convert.ToInt32(total.Text);
    int amountVal = Convert.ToInt32(amount.Text);
    total.Text = (totalVal + amountVal).ToString();
  }
}
```

Adder.aspx.cs



Każdej
kontrolce
odpowiada
pole klasy:
wartość
znacznika ID
odpowiada
nazwie pola w
klasie kodu
ukrytego



Klasa Control

```
public class Control: ... {  
    public virtual string ID { get; set; }  
    public virtual ControlCollection Controls { get; }  
    public virtual Control Parent { get; }  
    public virtual Page Page { get; set; }  
    public virtual bool Visible { get; set; }  
    protected virtual StateBag ViewState { get; }  
    public virtual bool EnableViewState { get; set; }  
    ...  
  
    public virtual bool HasControls();  
    public virtual Control FindControl (string id);  
    public virtual void DataBind();  
    protected virtual void LoadViewState (object state);  
    protected virtual object SaveViewState();  
    protected virtual Render (HtmlTextWriter w);  
    ...  
  
    public event EventHandler Init;  
    public event EventHandler Load;  
    public event EventHandler DataBinding;  
    public event EventHandler PreRender;  
    public event EventHandler Unload;  
    ...  
}
```

Properties

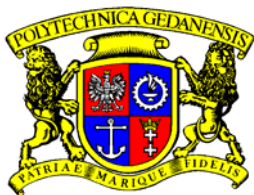
- name of the control
- nested controls
- enclosing control
- page to which the control belongs
- should the control be visible?
- state of this control
- should the state be persistent?

Methods

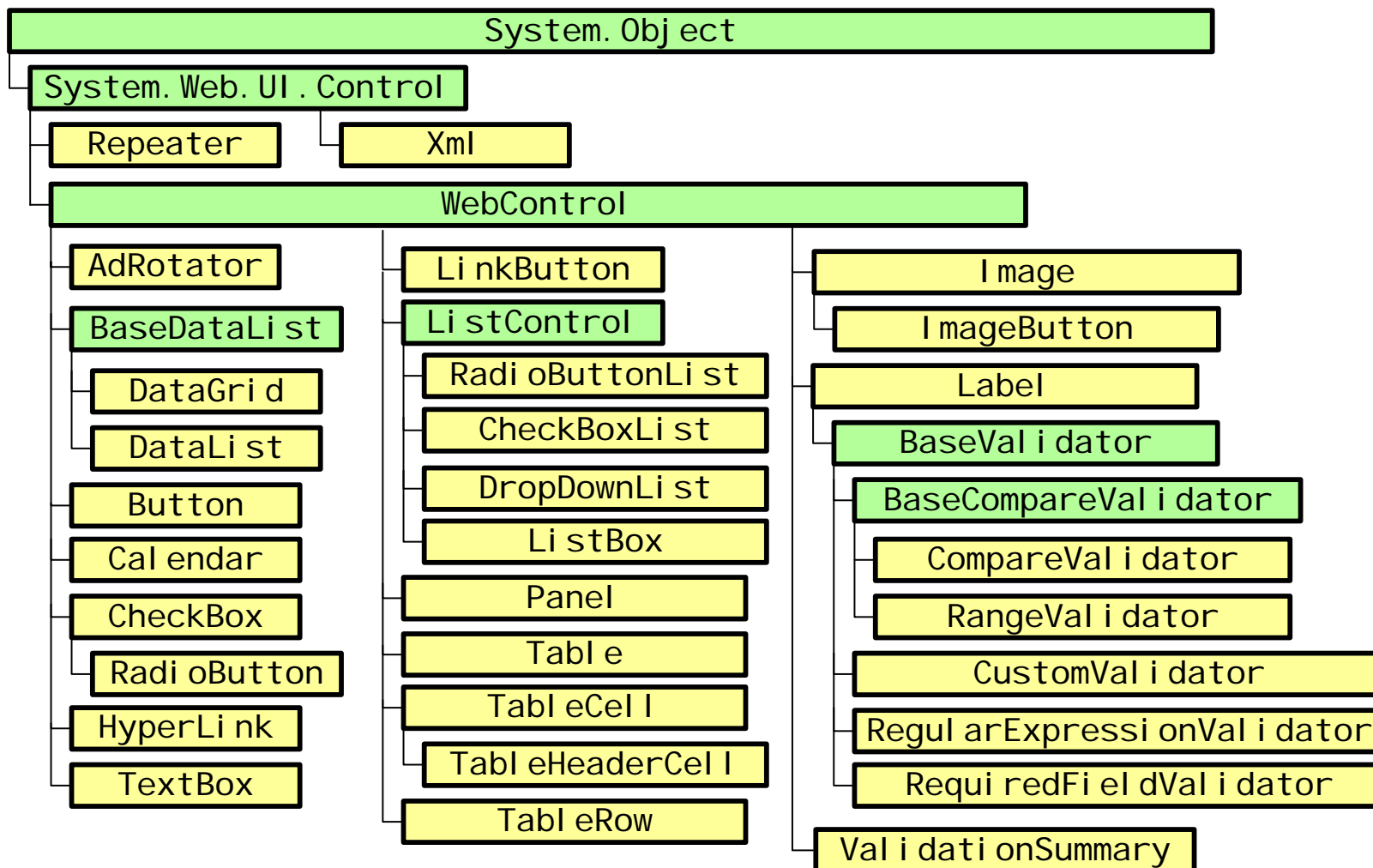
- does the control have nested controls?
- searches for a nested control with the name id
- loads data from a data source
- loads the state from the request stream
- saves the state to the response stream
- renders the control to HTML

Events

- after the control was created
- after the state was loaded from the request
- after DataBind was called
- before the control is rendered to HTML
- before the control is released


















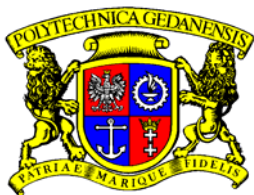
Hierarchia klas WebControls - przykłady









Web Controls - przykłady

 Label	Etykieta tekstowa
 TextBox	Pole tekstowe (element formularza do wprowadzania tekstu) działające w trybie jednowierszowym, wielowierszowym i trybie wprowadzania hasła
 Button	Przycisk
 LinkButton	Przycisk tekstowy
 ImageButton	Przycisk graficzny
 HyperLink	Hiperłącze ('link')
 DropDownList	Lista rozwijana (element formularza)
 ListBox	Pole listy
 DataGrid	DataGrid - tabela danych. Rozbudowana kontrolka pozwalająca na wyświetlanie i edycję dowolnych danych możliwych do przedstawienia w formie tabeli. Umożliwia m. in. sortowanie i stronicowanie danych.
 DataList	DataList - lista, która może być wypełniona dowolnymi danymi.
 Repeater	Repeater - najprostsza kontrolka pozwalająca na pracę z danymi tabelarycznymi. Umożliwia powielanie przygotowanego szablonu dla wszystkich elementów w zbiorze danych
 CheckBox	Pole wyboru (element formularza).
 CheckBoxList	Umożliwia tworzenie dynamicznych list pól wyboru .
 RadioButton	Pole opcji (element formularza).
 RadioButtonList	Umożliwia tworzenie dynamicznych list pól opcji



HTML Controls – przykłady (2)

 Image	Obrazek - tag HTML .
 Listbox	Pole listy - tag HTML <select size="n">...</select>, gdzie n-liczba wierszy.
 Dropdown	Pole rozwijane - tag HTML <select>...</select>.
 Horizontal Rule	Pozioma linia oddzielająca - tag HTML <hr>.

Widok kontrolek Web Controls - przykłady

abc

Click

Radio

Check

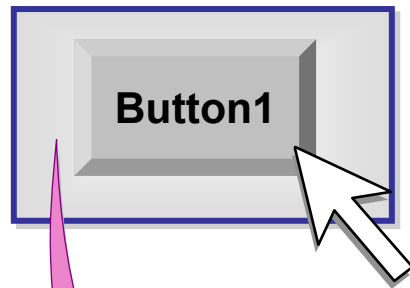
apples
pears
bananas

≤ Februar 2003 ≥						
Mo	Di	Mi	Do	Fr	Sa	So
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	1	2
3	4	5	6	7	8	9

EmployeeID	FirstName	LastName
1	Nancy	Davolio
2	Andrew	Fuller
3	Janet	Leverling
4	Margaret	Peacock
5	Steven	Buchanan
6	Michael	Suyama
7	Robert	King
8	Laura	Callahan
9	Anne	Dodsworth



2.3.3 Event Model in the .NET Framework



Invokes the delegate

```
this.button1.Click += new  
System.EventHandler(this.button1_Click);
```

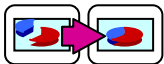
```
private void button1_Click(object sender,  
System.EventArgs e)  
{  
...  
}
```

Delegate calls the associated procedure

Application-level Event Handling
– Web Forms

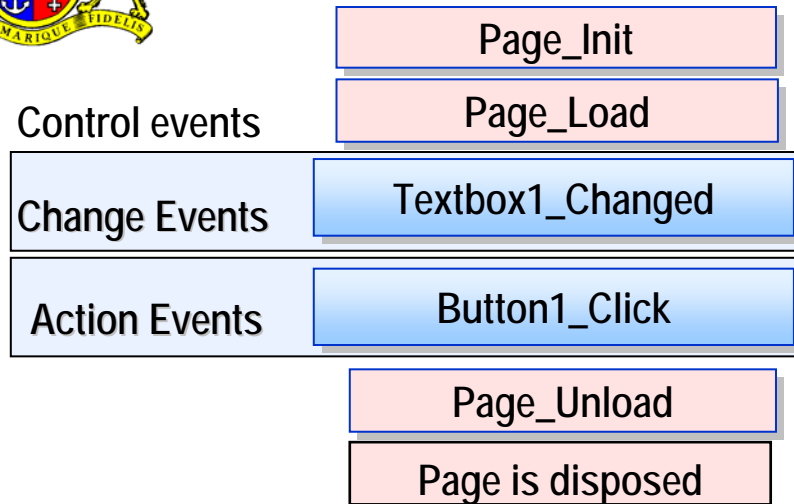
Delegate

- Delegate Model
 - Connect event sender and event receiver
 - Single and multicast delegates
- Event Delegates Are Multicast
- Event Wiring
 - Register event handler with event sender





Page and Controls Events Life Cycle



Control	Event	When does the event occur?
all	Init Load PreRender Unload	<ul style="list-style-type: none"> • when the control is created • after the data that were sent by the browser have been loaded into the control • before HTML code for this control is generated • before the control is removed from memory
Button	Click	when the button was clicked
TextBox	TextChanged	when the contents of the TextBox changed
CheckBox	CheckedChanged	when the state of the CheckBox changed
ListBox	SelectedIndexChanged	when a new item from the list has been selected



2.3.4 Błędy i ich obsługa w aplikacjach ASP.NET

- **Przekierowanie użytkownika na stronę błędu**

- Konfiguracja na poziomie strony

- atrybut **errorPage** w dyrektywie Page

- własność Page.ErrorPage

- Konfiguracja na poziomie aplikacji

- sekcja **customErrors** w pliku Web.config

- **Przechwytywanie i obsługa wyjątków**

- Obsługa wyjątków na poziomie lokalnym

Konstrukcja: try – catch – finally.
Response.Write(tekst).

- Obsługa wyjątków na poziomie strony

Zdarzenie Page.Error; obsługa metoda Page_Error ()

- Obsługa wyjątków na poziomie aplikacji

Zdarzenie HttpApplication; obsługa Application_Error zdefiniowana w pliku Global.asax

- **Śledzenie wykonywania aplikacji – tracing**

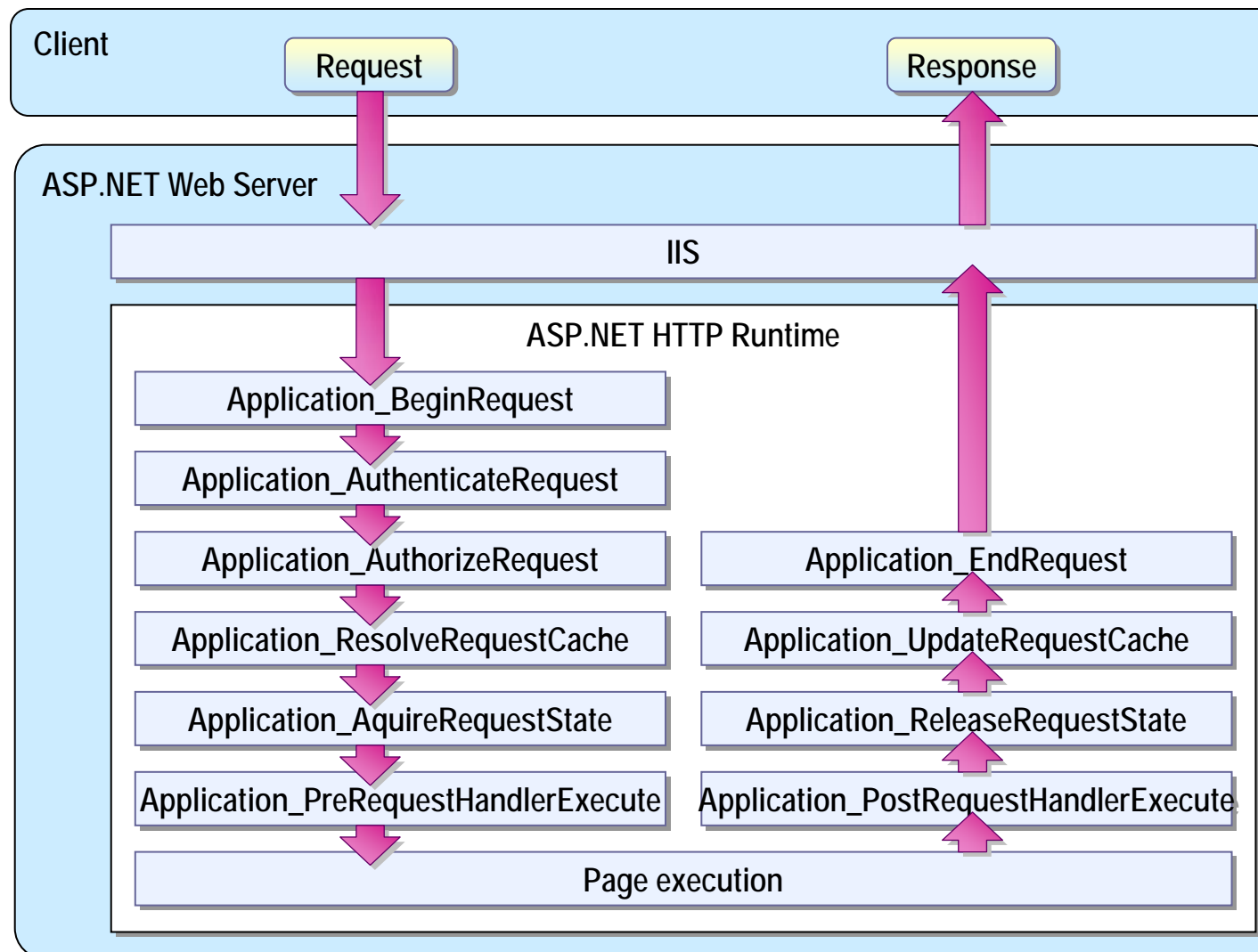
- Śledzenie wykonywania na poziomie strony

Trace.Write, Trace.Warn

- Śledzenie wykonywania na poziomie aplikacji



2.3.5 Kontrola zdarzeń aplikacji ASP.NET: plik aplikacji Global.asax

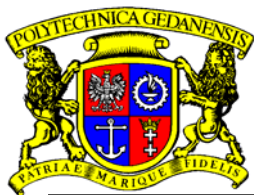


Zdarzenia globalne/warunkowe

- Application_Start
- Application_End
- Application_Error
- Session_OnStart
- Session_OnEnd

Zdarzenia związane z żądaniem

Aplikacja ASP.NET to: suma wszystkich plików, stron, procedur obsługi zdarzeń, modułów, kodu wykonywalnego (programów i bibliotek), wykonywanego lub uruchamianego w obrębie danego **katalogu wirtualnego** (i jego podkatalogów) na serwerze WWW



Zdarzenia aplikacji

Zdarzenie	Opis
BeginRequest	Zgłaszane w momencie rozpoczynania obsługi żądania
AuthenticateRequest	Zgłaszane gdy żądanie HTTP gotowe jest do uwierzytelnienia
AuthorizeRequest	Zgłaszane gdy żądanie HTTP gotowe jest do autoryzacji
ResolveRequestCache	Używane przez moduł pamięci podręcznej w celu obsługi danego żądania jeśli jest już przechowywane w pamięci podręcznej
AcquireRequestState	Zgłaszane gdy aplikacja uzyska informacje o stanie (np. sesji) związanym z danym żądaniem
PreRequestHandlerExecute	Zgłaszane bezpośrednio przed rozpoczęciem realizacji procedury obsługi żądań przez HTTP handler
PostRequestHandlerExecute	Zgłaszane bezpośrednio po zakończeniu realizacji procedury obsługi żądań przez HTTP handler
ReleaseRequestState	Zgłaszane w celu zapamiętania danych o stanie sesji dla danego żądania
UpdateRequestCache	Zgłaszane gdy aplikacja uaktualnia pamięć podręczną dla danego żądania
EndRequest	Zgłaszane w momencie zakończenia obsługi żądania
PreSendRequestContent	Zgłaszane bezpośrednio przed wysłaniem zawartości żądania HTTP
PreSendRequestHeaders	Zgłaszane bezpośrednio przed wysłaniem nagłówków żądania HTTP
Error	Zgłaszane w momencie wystąpienia jakiegokolwiek błędu



Przykład: plik aplikacji Global.asax.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Web;
using System.Web.SessionState;

namespace BenefitsCS
{
    /// <summary>
    /// Summary description for Global.
    /// </summary>
    public class Global :
        System.Web.HttpApplication
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer
        components = null;
        public Global()
        {
            InitializeComponent();
        }

        protected void Application_Start(Object
        sender, EventArgs e)
        {
        }
        ...
    }
}
```

```
protected void Session_Start(Object sender, EventArgs e)
{
}

protected void Application_BeginRequest(Object sender,
EventArgs e)
{
}

protected void Application_EndRequest (Object sender,
EventArgs e)
{
}

protected void Application_AuthenticateRequest (Object
sender, EventArgs e)
{
}

protected void Application_Error(Object sender,
EventArgs e)
{
}

protected void Session_End(Object sender, EventArgs e)
{
}

protected void Application_End(Object sender, EventArgs
e)
{
}
```



2.3.6 The ASP.NET 2.0 Page Postback Model

Komunikacja zwrotna, odsyłanie (*postback*) polega na powracaniu do określonej strony WWW w czasie trwania sesji z serwerem

- **ASP.NET Postback model** - a mechanism for sending control properties on Web pages from the Web browser to the Web server and for restoring those values when a response is sent back from the Web server to the Web browser
 - enables Web server controls to retain their values over multiple requests to the server, even though the underlying HTTP mechanisms are stateless
- **AutoPostBack** property - this property controls whether user interaction with the control should invoke a round-trip request to the server. Some Web server controls support the *AutoPostBack* property.
- **EnableViewState** property of a Web server control determines whether the control should retain its state for the duration of the postback.
- **Cross-Page Postbacks** – you can configure controls to post requests to a different page by setting their *PostBackUrl* properties

IsPostBack – właściwość strony umożliwiająca programowe rozróżnienie między żadaniami przesyłanymi zwrotnie a pierwszymi wywołaniami strony



Which Events Cause a Round Trip?

Round trip events (cause an immediate round trip)



Click

```
<asp:Button Text="click me" Runat="server"  
OnClick="DoClick" />
```

Delayed events (are handled at the next round trip)



TextChanged

```
<asp:TextBox Runat="server"  
OnTextChanged="DoTextChanged" />
```



SelectedIndexChanged

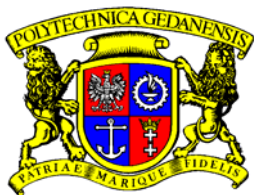
```
<asp:ListBox Rows="3" Runat="server"  
OnSelectedIndexChanged="DoSIChanged" />
```

AutoPostBack (causes a delayed event to lead to an immediate round trip)

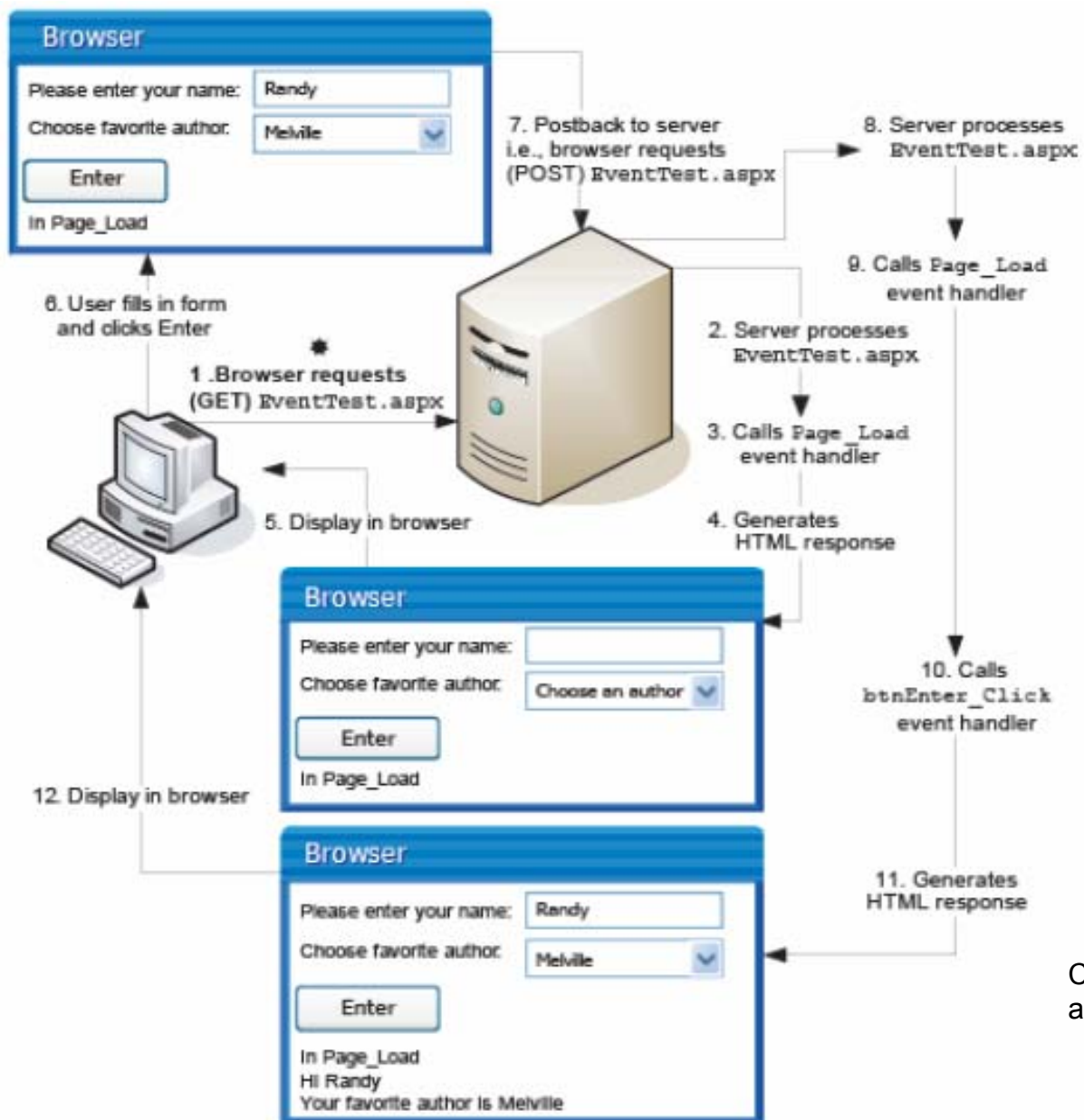


TextChanged

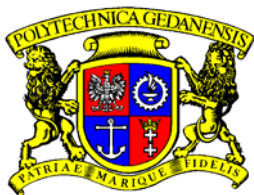
```
<asp:TextBox Runat="server"  
AutoPostBack="true"  
OnTextChanged="DoTextChanged" />
```



Postback



Connolly R. ASP.NET 2.0. Projektowanie aplikacji internetowych, Helion 2008



Mechanizm Cross-Page Postback

- Strony mogą wykonać post back do innych stron
- Wykorzystywane właściwości:
 - *control.PostBackUrl* – adres docelowy żądania zwrotnego
 - *Page.PreviousPage* – zwraca referencję do strony, która generowała żądanie zwrotne
 - *PreviousPage.IsCrossPagePostBack* – informacja, czy wystąpiło żądanie zwrotne z innej strony

```
<html >
  <body>
    <form runat="server">
      <asp:TextBox ID="Input" RunAt="server" />
      <asp:Button Text="Test" PostBackUrl="PageTwo.aspx" RunAt="server" />
    </form>
  </body>
</html >
```



2.3.7 Kontrolki: nawigacyjne, logowania

• Nowe kontrolki ułatwiają nawigację pomiędzy stronami WWW

- Plik **.sitemap** – opis struktury logicznej witryny (**mapa witryny**) to lista stron i adresów URL
- **Menu** - wyświetla rozwijane lub wyskakujące menu w oparciu o dane przekazane przez *SiteMapDataSource*
- **TreeView** – wyświetla dane jako hierarchiczny układ węzłów, które można rozwijać lub ukrywać;
- **SiteMapPath** - pokazuje ścieżkę dostępu do aktualnej strony poprzez pośredniczące strony
 - **SiteMapDataSource** – dostęp do hierarchicznej listy łączy, zapisanej w pliku XML o domyślnej nazwie **web.sitemap**

```
<siteMap>
<siteMapNode title="Home" url="default.aspx">
  <siteMapNode title="Tab1" url="subdir/default.aspx">
    <siteMapNode title="SubPage" url="subdir/foo.aspx" />
  </siteMapNode>
  <siteMapNode title="Tab2" url="Tab/default.aspx">
    <siteMapNode title="SubPage" url="Tab/foo.aspx"/>
  </siteMapNode>
</siteMapNode>
</siteMap>
```

• Login controls

- Kontrolki dla logowania, tworzenia użytkowników, odzyskiwania haseł, i inne...

Log In	
User Name:	<input type="text"/>
Password:	<input type="password"/>
<input type="checkbox"/>	Remember me next time.
<input type="button" value="Log In"/>	



Kontrolki sprawdzające - Validation Controls

Obiekty sprawdzające poprawność danych wejściowych wprowadzanych przez użytkownika

RequiredFieldValidator

sprawdzający czy w danym polu została wprowadzona jakakolwiek wartość

RangeValidator

sprawdzający czy dane znajdują się w określonym przedziale wartości

CompareValidator

porównujący wartości dwóch pól tekstowych

CustomValidator

sprawdzanie poprawności określone przez algorytm użytkownika

RegularExpressionValidator

sprawdzanie poprawności „na dopasowanie wzorca” określonego wyrażeniem regularnym



Dodawanie kontrolki sprawdzającej do Web Form (2)

1. Dodanie kontrolki sprawdzającej
2. Wybór kontrolki wejściowej do sprawdzania
3. Ustalenie właściwości kontroli

```
<asp:TextBox id="txtName" runat="server" />
```

```
<asp:Type_of_Validator  
  id="Validator_id"  
  runat="server"  
  ControlToValidate="txtName"  
  ErrorMessage="Message_for_error_summary"  
  Display="static|dynamic|none"  
  Text="Text_to_display_by_input_control">  
</asp:Type_of_Validator>
```

- RequiredFieldValidator
- CompareValidator
- RangeValidator
- RegularExpressionValidator
- CustomValidator

Wspólne atrybuty kontrolek:
Id kontrolki, której wartość będzie sprawdzana

Wiadomość wyświetlana, gdy dane są niepoprawne

Display - sposób wyświetlania kontrolki sprawdzania poprawności:

- Static - do strony zostanie dodane miejsce przeznaczone na wyświetlanie informacji
- Dynamic - miejsce na wyświetlanie informacji kontrolki zostanie dodane dynamicznie, jeśli dane okażą się niepoprawne
- None – wartość atrybutu ErrorMessage nigdy nie będzie wyświetlana



Zastosowanie kontrolki ValidationSummary (3)

- Wyświetla listę komunikatów o błędach, zgłoszonych przez znajdujące się na stronie kontrolki sprawdzania poprawności
- Może wyświetlać zawartość atrybutów *text* i *error messages*
- Używa `Text="*"` do wskazania lokalizacji błędu

```
<asp:ValidationSummary id="valSummary"
  runat="server"
  HeaderText="These errors were found:"
  ShowSummary="True"
  DisplayMode="List"/>
```

Atrybuty kontrolki podsumowania:

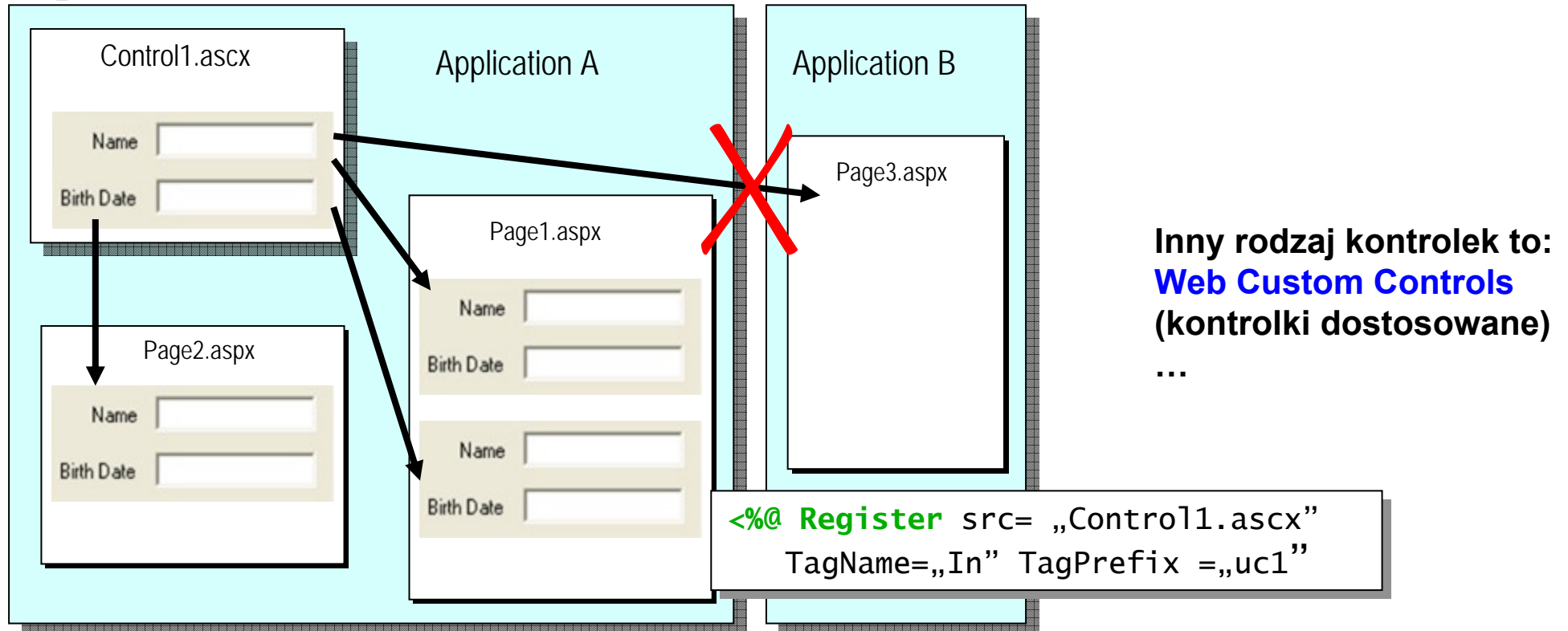
- `DisplayMode` – format wyświetlania podsumowania; `List` - jako listę w osobnych wierszach, `BulletList` - jako listę wypunktowaną, `SingleParagraph` - w jednym paragrafie
- `EnableClientScript` - włączenie/wyłączenie kodu generowanego po stronie klienta z podsumowania błędów sprawdzania poprawności
- `ShowSummary` – gdy `true`, podsumowanie zostanie wyświetlone na Web Form ...

Page.IsValid Property

```
private void cmdSubmit_Click(object s, System.EventArgs e)
{
    if (Page.IsValid)
    {
        Message.Text = "Page is Valid!";
        // Perform database updates or other logic here
    }
}
```



Kontrolki użytkownika - User Controls



- Kontrolki użytkownika umożliwiają wielokrotne wykorzystanie kodu i komponentów UI w aplikacji WEB
- Kontrolka użytkownika jest kontrolką serwerową o rozszerzeniu **.ascx**
- Zawiera HTML, oprócz znaczników **<HTML>**, **<BODY>**, or **<FORM>**
- Zawiera kod obsługi zdarzeń

```
<%@ Control Language="c#" %>
```



2.3.8 Wiązanie danych (Data-Binding)

- **Wiązanie danych** – proces uzyskiwania danych ze **źródła** i dynamicznego wiązania ich z pewną **właściwością** elementu wizualnego (kontrolki)
- Kontrolki serwera można powiązać logicznie ze źródłem danych, używając zbioru właściwości takich jak **Text, DataSource, DataTextField**
- Powiązanie kontrolki z danymi **uaktywniane** jest poprzez wywołanie metody **DataBind**.
- **Źródło danych** – obiekt implementujący interfejs **ICollection**
 - Dane reprezentowane przez klasy kolekcji .NET (tablice, słowniki, listy, sterty, kolejki)
 - Struktury danych definiowane przez użytkowników
 - Dane reprezentowane przez klasy bazodanowe (**DataTable, DataSet, DataReader**)
 - Kontrolki źródeł danych
 - Widoki reprezentowane przez klasę **DataView**



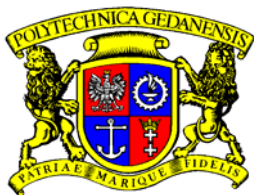
Kontrolki prezentacji i źródeł danych

Wyświetlanie informacji:

- **GridView:** Wyświetlanie rekordów w formie arkusza, umożliwia modyfikację i usuwanie danych
- **DetailsView:** wyświetla jeden rekord; wstawianie, modyfikacja, usuwanie
- **FormView:** wyświetla jeden rekord w sformatowanej postaci; wstawianie, modyfikacja, usuwanie
- ...

Kontrolki źródeł danych:

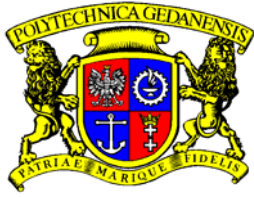
- **SqlDataSource:** umożliwia odczytywanie i zmianę informacji w bazie danych MS SQL
- **XMLDataSource:** umożliwia odczytywanie i zmianę informacji w plikach XML
- **ObjectDataSource:** umożliwia odczytywanie i zmianę danych zawartych w niestandardowych obiektach
- **SiteMapDataSource:** odczytywanie informacji z pliku mapy witryny



Kontrolki wyświetlające dane

Nazwa	Opis
GridView	Wyświetla wiele rekordów w formie arkusza (modyfikacja, usuwanie)
DetailsView	Wyświetla jeden rekord na raz, używając prostych instrukcji HTML (wstawianie, modyfikacja, usuwanie)
FormView	Wyświetla jeden rekord w sformatowanej postaci (wstawianie, modyfikacja, usuwanie)
DataList	Powtarza zdefiniowany przez projektanta szablon dla każdego rekordu z źródła danych
Repeater	Zbliżona do DataList, ale nie zawiera własnych elementów HTML.

- Powiązanie z danymi jest za pomocą określenia właściwości kontrolki (np. dla DropDownList)
 - **DataSourceId** – nazwa źródła danych (tabela lub zapytanie)
 - **DataTextField** – nazwa pola zawierającego wartości, które mają być wyświetlone
 - **DataValueField** – nazwa pola zawierającego wartości, które strona ma odesłać po wskazaniu danej opcji listy rozwijanej (pole klucza)

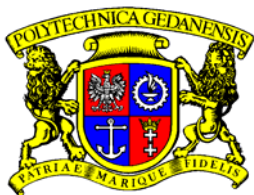


Kontrolka GridView

- **Zbliżona do kontrolki DataGrid**
 - Generuje zbiór danych w postaci tabel HTML
- **Wbudowane funkcje: sortowanie, stronicowanie, wybieranie, modyfikowanie i kasowanie**
- **Bogaty asortyment typów komórek: ImageFields, CheckBoxFields**
- **Wiele możliwości modyfikacji UI**

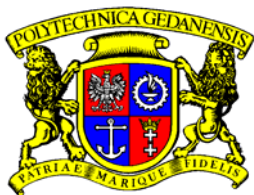
```
<asp:SqlDataSource ID="Emplooyees" RunAt="server"
  ConnectionString="server=local host; database=northwi nd; . . . "
  SelectCommand="sel ect l astname, fi rstname, ti tle from empl oyees" />

<asp:Gri dVi ew DataSourceID="Emplooyees" Wi dth="100%" RunAt="server" />
```



Typy komórek GridView

Nazwa	Opis
BoundField	Generuje kolumny z tekstem
ButtonField	Generuje kolumny z przyciskami (push button, image, or link)
CheckBoxField	Generuje kolumny z polem wyboru dla typu Boolean
CommandField	Generuje kontrolki dla wyboru i edycji danych w GridView
HyperLinkField	Generuje kolumny z łączami
ImageField	Generuje kolumny z obrazkami
TemplateField	Generuje kolumny z użyciem szablonów HTML



Kontrolki źródła danych

Zapełnienie kontrolek danymi poprzez wiązanie nie bezpośrednio z kolekcją danych, ale z kontrolką do obsługi źródła danych

Deklaratywne podpinanie danych

<i>Name</i>	<i>Description</i>
SqlDataSource	Łączy kontrolki z bazami SQL
AccessDataSource	Łączy kontrolki z bazami Access
XmlDataSource	Łączy kontrolki z danymi XML
ObjectDataSource	Łączy kontrolki z komponentami
SiteMapDataSource	Łączy kontrolki nawigacyjne z danymi nawigacyjnymi (mapą witryny)

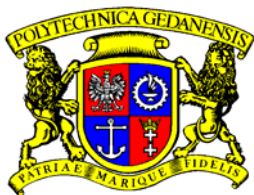


Kontrolka SqlDataReader

```
<asp:SqlDataSource ID="Titles" RunAt="server"
  ConnectionString="server=localhost;database=pubs;integrated security=true"
  SelectCommand="select title_id, title, price from titles" />

<asp:DataGrid DataSourceID="Titles" RunAt="server" />
```

- **Deklaratywne podpinanie danych do baz SQL**
- **Dwustronne podpinanie danych**
 - SelectCommand
 - InsertCommand, UpdateCommand i DeleteCommand
- **Możliwość umieszczania wyników zapytania w pamięci podręcznej Cache**
- **Parametryzowane operacje**



XmlDataSource

- Deklaratywne wiązanie do danych XML
- Wspiera umieszczanie danych w pamięci podręcznej cache i transformacje XSL
- Jednostronne wiązanie danych

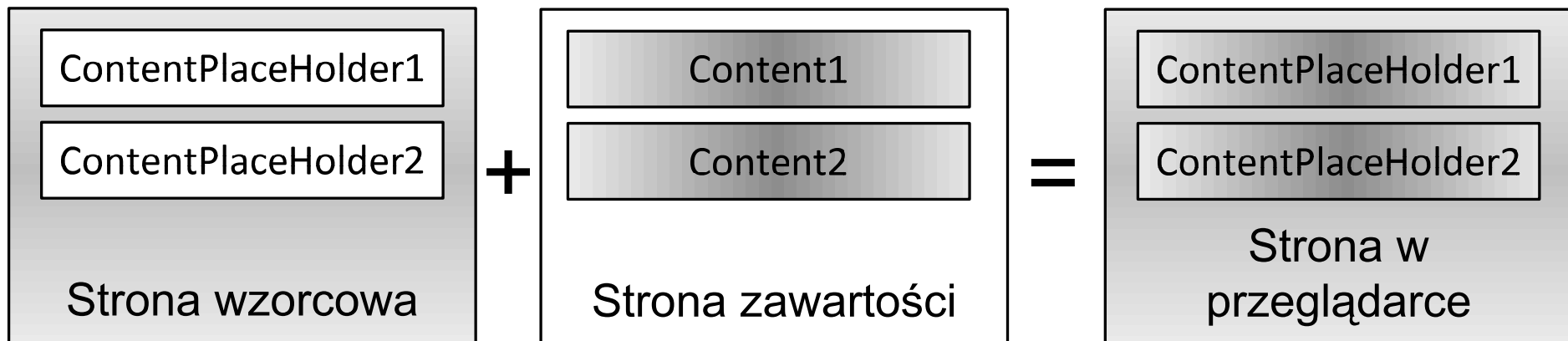
```
<asp:XmlDataSource ID="Rates" DataFile="Rates.xml" RunAt="server" />  
<asp:TreeView ID="MyTreeView" DataSourceID="Rates" RunAt="server" />
```

Nazwa	Opis
DataFile	Ścieżka do pliku XML
TransformFile	Ścieżka do pliku XSL
EnableCaching	Włączone/wyłączone korzystanie z pamięci podręcznej Cache
CacheDuration	Czas przetrzymywania danych w pamięci podręcznej (sekundy)
CacheExpirationPolicy	Typ CacheDuration
CacheKeyDependency	Zależność Cache na określonym kluczu
XPath	Wyrażenie XPath służące filtrowaniu danych



Tworzenie stron wzorcowych - Master Page

- Strona wzorcowa - umożliwia sterowanie wspólnym wyglądem dla całej witryny internetowej; zawiera tylko wspólne elementy
- Strona zawartości - miejsce w którym ma zostać umieszczona treść strony zawartości jest określone przez specjalną kontrolkę **ContentPlaceholder**





- Plik *.master* zawiera szablon wykorzystywany przez wszystkie strony aplikacji
- Inne strony zawierają się jako kontrolki w stronie „Master Page”




Tworzenie strony wzorcowej i stron zawartości (2)

- **Dodawanie strony wzorcowej do projektu**
 - za pomocą menu *Website\Add New Item*, wybierając szablon *Master Page*
- **Dodawanie strony zawartości**
 - Dodatkowo zaznaczenie opcji *Select master page*. Strona zawartości korzystająca z tej strony wzorcowej musi zawierać kontrolkę **Content**, która atrybut *ContentPlaceHolderID* jest równy wartości *ContentPlaceHolderID* strony wzorcowej.
- **Modyfikowanie istniejącej strony Web na stronę używającą strony wzorcowej**

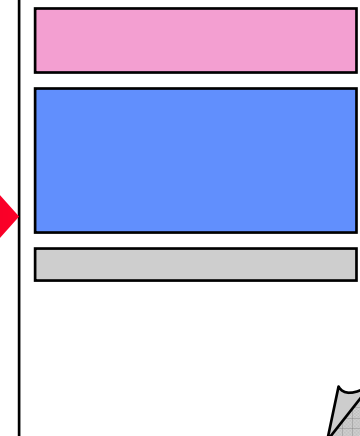
Site.master

```
<%@ Master %>  
  
<asp:ContentPlaceHolder  
ID="Main"  
RunAt="server" />  

```

default.aspx

```
<%@ Page MasterPage-  
File="Site.master" %>  
  
<asp:Content  
ContentPlaceHolderID=  
"Main" RunAt="server" />  
  
  
</asp:Content>
```

http://.../default.aspx





Definiowanie i korzystanie z szablonu

```
<%@ Master %>
<html >
  <body>
    <!-- Banner shown on all pages that use this master -->
    <table width="100%">
      <tr>
        <td bgcolor="darkblue" align="center">
          <span style="font-size: 36pt; color: white">ACME Inc. </span>
        </td>
      </tr>
    </table>

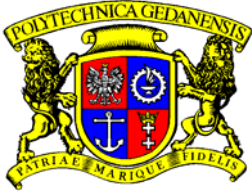
    <!-- Placeholder for content below banner -->
    <asp:ContentPlaceholder ID="Main" RunAt="server" />
  </body>
</html >
```

Definiowanie

Użycie

```
<%@ Page MasterPageFile="~/Site.master" %>

<asp:Content ContentPlaceholderID="Main" RunAt="server">
  This content fills the placeholder "Main" defined in the master page
</asp:Content>
```



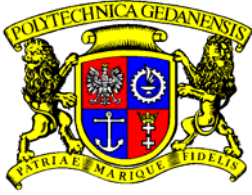
2.3.9 Zarządzanie stanem w ASP.NET

Zarządzanie stanem – zdolność do przechowywania i przekazywania informacji np. pomiędzy żądaniami pojedynczego użytkownika.

Kryteria mechanizmów przechowywania i zarządzania stanem: zasięg, czas życia, dopuszczalny rozmiar przechowywanych danych oraz po jakiej stronie są przechowywane.

- **Aplikacji (application state)** - do przechowywania danych i stosowania w obrębie całej aplikacji (dla każdego użytkownika tej aplikacji)
np. dane konfiguracyjne, liczba sesji, ...
- **Sesji (session state)** – do przechowywania danych i dostępu do danych w obrębie sesji (dla pojedynczego użytkownika)
np. stan karty zakupów, email klienta, ...
- **Strony (page state)** - do przechowywania informacji strony sieci WWW pomiędzy kolejnymi przetworzeniami strony
np. zawartość pól tekstowych TextBoxes, stan CheckBoxes, ...

Aplikacja ASP.NET to: suma wszystkich plików, stron, procedur obsługi zdarzeń, modułów, kodu wykonywalnego (programów i bibliotek), wykonywanego lub uruchamianego w obrębie danego katalogu wirtualnego (i jego podkatalogów) na serwerze WWW



Zapamiętywanie stanu kontrolki - Saving View State

- From the base Control class, all the controls inherit the *ViewState* property
- Two most common problems of the view state:
 - High load time: page must serialize and deserialize the view state for itself and all child controls
 - Increased page size: because the view state is placed in a hidden form called ***_VIEWSTATE***

Ukrytej kontrolka, zwana *_ViewState*, przechowuje w zaszyfrowanej postaci dane kontrolek formularza

- Problem wydajności - selektywne włączanie /wyłączanie mechanizmu zapamiętywania stanu

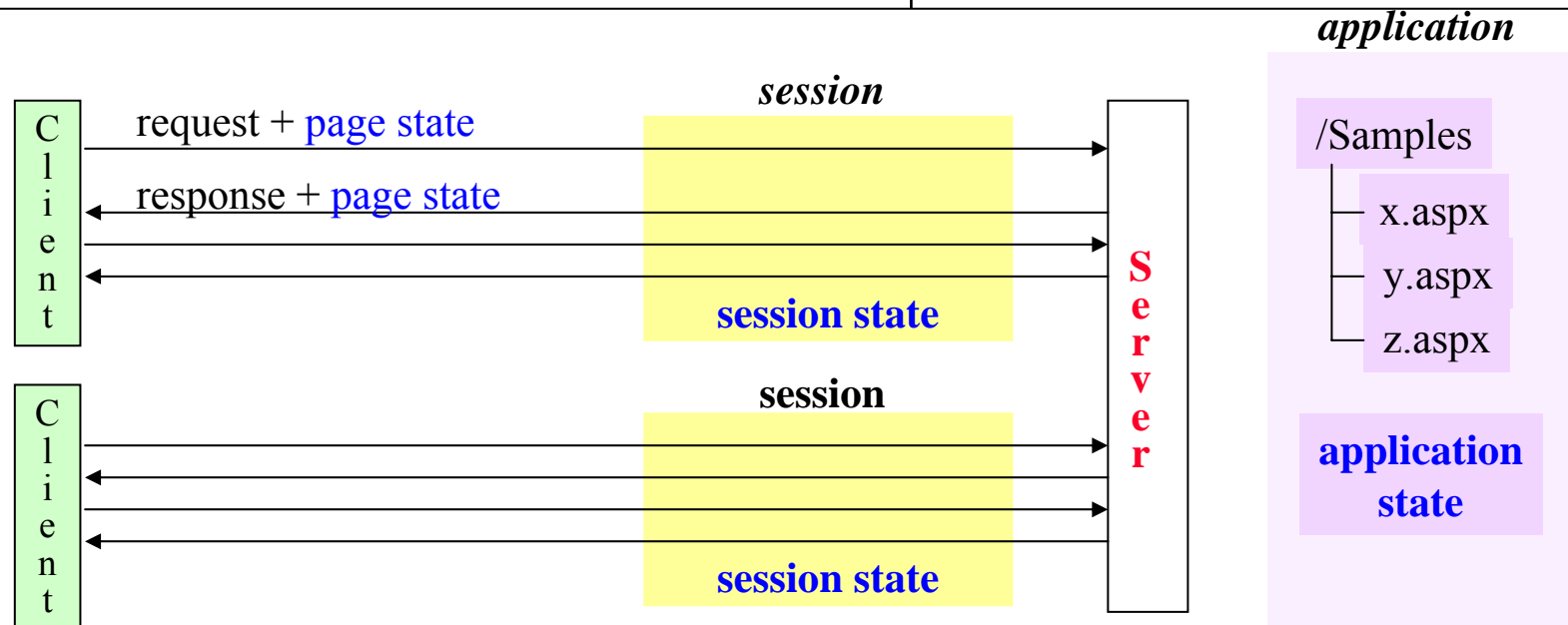
```
<input type="hidden" name=„_VIEWSTATE”  
value="dDwtMTA4MzE0MjEwNTs7Pg==" />
```

```
<%@ Page EnableViewState="False" %>  
<asp:ListBox id="ListName" EnableViewState="true"  
runat="server">  
</asp:ListBox>
```



Types of State Management (1)

Server-Side State Management	Client-Side State Management
<p>Application state</p> <ul style="list-style-type: none"> Information is available to all users of a Web app. 	<p>Cookies</p> <ul style="list-style-type: none"> Text file stores information to maintain state
<p>Session state</p> <ul style="list-style-type: none"> Information is available only to a user of a specific session 	<p>The ViewState property</p> <ul style="list-style-type: none"> Retains values between multiple requests for the same page
<p>Database</p> <ul style="list-style-type: none"> In some cases, use database support to maintain state on your Web site 	<p>Query strings</p> <ul style="list-style-type: none"> Information appended to the end of a URL





Obiekty zarządzania stanem w ASP.NET

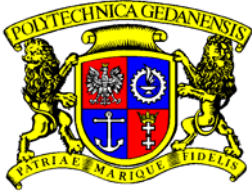
Dane o **stanie aplikacji** są przechowywane w obiekcie klasy `HttpApplicationState`, typu słownikowego zawierającego pary typu klucz-wartość.

- Zasoby – szybki dostęp (stan aplikacji jest przechowywany w pamięci); jednak przechowywanie dużych bloków danych w stanie aplikacji może wypełnić pamięć serwera.
- Ulotność – stan aplikacji jest przechowywany w pamięci, jest usuwany z niej w momencie zatrzymania lub restartu aplikacji lub w momencie awarii serwera.
- Skalowalność – stan aplikacji nie jest dzielony na serwery w farmie serwerów.
- Współbieżność – do stanu aplikacji może jednocześnie odwoływać się wiele wątków (zapewnienie mechanizmów bezpiecznej aktualizacji przechowywanych obiektów).

Dane o **stanie sesji** użytkownika są przechowywane w obiekcie klasy `HttpSessionState`, typu słownikowego zawierającego pary typu klucz-wartość.

- Zawartość tego obiektu jest dostępna poprzez właściwości `Session` klasy `Page` i `HttpContext`.
- Niepowtarzalny identyfikator sesji, służy do identyfikacji kolejnych żądań pochodzących od tego samego klienta.
- Właściwość `Session.Mode` wskazuje gdzie przechowywane są dane o sesji: `InProc`, `StateServer` lub `SqlServer`.

Plik `global.asax` umożliwia definiowanie **procedur obsługi zdarzeń** na poziomie sesji i aplikacji.



Server-Side State: Initializing and Using Application and Session Variables

- Variables are initialized in **Global.asax**
 - The **Application** object shares information among all users of a Web application
 - The **Session** object stores information for a particular user session

```
protected void Application_Start(Object sender, EventArgs e)
{
    Application["NumberOfVisitors"] = 0;
}
```

- **Set session and application variable**

```
Session["BackColor"] = "blue";
Application.Lock();
Application["NumberOfVisitors"] = (int)Application["NumberOfVisitors"] + 1;
Application.Unlock();
```

- **Read session and application variable**

```
strBgColor = (string)Session["BackColor"];
lblNbVisitor.Text = Application["NumberOfVisitors"].ToString();
```



Server-Side State: Dostęp do informacji o stanie, parametry zmiennych sesji, aplikacji

Dostęp do informacji:

Page state

writing: **ViewState**["counter"] = counterVal;

reading: int counterVal = (int) **ViewState**["counter"];

Session state

writing: **Session**["cart"] = shoppingCart;

reading: DataTable shoppingCart = (DataTable) **Session**["cart"];

Application state

writing: **Application**["database"] = databaseName;

reading: string databaseName = (string) **Application**["databaseName"];

Application and Session Variable Duration

- Session variables have a set duration after last access (default is 20 minutes)
- Session duration can be changed in **web.config**:
- Application variables persist until the **Application_End event** is fired

```
<configuration>
  <system.web>
    <sessionState timeout="10" />
  </system.web>
</configuration>
```



Przechowywanie stanu sesji

- **InProc**

- Przechowywane w przestrzeni adresowej *ASP.NET worker process*
- Szybkie działanie,
- Dane są tracone gdy proces jest restartowany

- **Out-of-Proc**

- **State Server**

- Usługa Windows *ASP.NET State Service*

- Instalowana domyślnie, ale nie uruchomiona

- Niezależne od IIS – proces *aspnet_state.exe*

- Konfiguracja IP i portu

`stateConnectionString=„,tcpip=127.0.0.1:42424”`

- **SQL Server**

- Dane sesyjne przechowywane w SQL Server

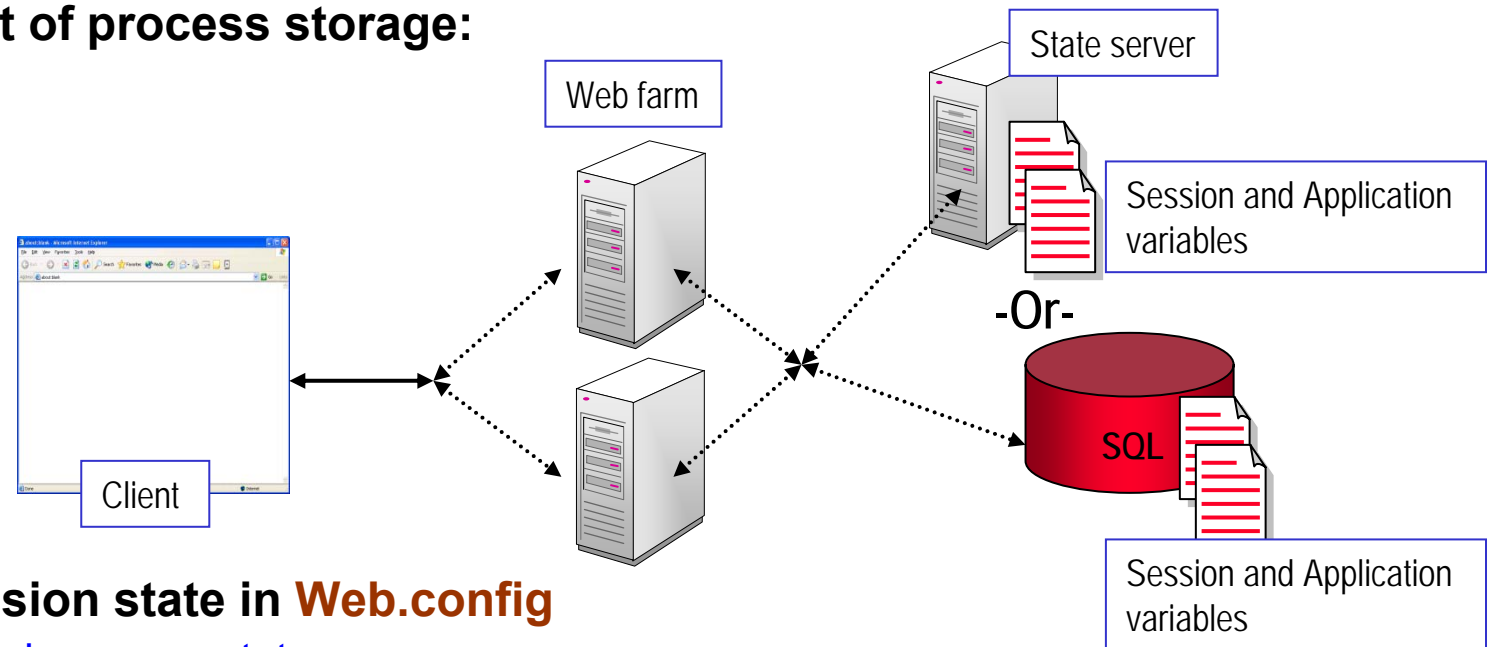
- Wymaga stworzenia baz danych *InstallSqlState.sql / InstallPersistantSqlState.sql*

- Największa niezawodność – możliwość klastrowania



Server-Side State: Scalable Storage of Application and Session Variables

- By default, the session state is managed in process
- Disadvantage of in process storage:
 - Not Scalable
- ASP.NET provides **out of process** storage of session state
 - State can be stored in a SQL Server database or a state server
- Advantages of out of process storage:
 - Scalable



- Configure the session state in **Web.config**
 - Mode is set to `sqlserver` or `stateserver`
- Then, configure the SQL server

```
<sessionState mode="SQLServer" sqlConnectionString="data source=SQLServerName; Integrated security=true" />
```



Client-Side State: Using Cookies to Store Session Data

- **Creating a cookie:**

```
HttpCookie objCookie = new HttpCookie("myCookie");  
DateTime now = DateTime.Now;
```

```
objCookie.Values.Add("Time", now.ToString());  
objCookie.Values.Add("ForeColor", "White");  
objCookie.Values.Add("BackColor", "Blue");
```

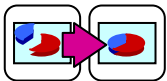
```
objCookie.Expires = now.AddHours(1);
```

```
Response.Cookies.Add(objCookie);
```

To create a persistent cookie, specify the expiration time

- **Cookie contains information about the domain name**

```
Set-Cookie: Username=John+Chen; path=/  
domain=microsoft.com;  
Expires=Tuesday, 01-Feb-05 00.00.01 GMT
```





Client-Side State: Retrieving Information from a Cookie

- **Read the cookie**

```
HttpCookie objCookie = Request.Cookies["myCookie"];
```

- **Retrieve values from the cookie**

```
lblTime.Text = objCookie.Values["Time"];  
lblTime.ForeColor = System.Drawing.Color.FromName  
                    (objCookie.Values["ForeColor"]);  
lblTime.BackColor = System.Drawing.Color.FromName  
                    (objCookie.Values["BackColor"]);
```



Using Cookieless Sessions

- Each active session is identified and tracked using session IDs
- Session IDs are communicated across client-server requests using an HTTP cookie or included in the URL
- Cookieless sessions
 - Session ID information is encoded into URLs

```
http://server/(h44a1e55c0breu552yrecob1)/page.aspx
```

- Cannot use absolute URLs
- Most browsers limit the URL size to 255 characters, which limits use of cookieless Session IDs

Setting Up Cookieless Sessions

- Session state is configured in the <SessionState> section of Web.config
- Set cookieless = true

```
<sessionState cookieless="true" />
```