Platforma .NET – Wykład 10
Technologia ASP.NET

Osoba prowadząca wykład, laboratorium i projekt:
dr hab. inż. Marek Sawerwain, prof. UZ

Instytut Sterowania i Systemów Informatycznych
Uniwersytet Zielonogórski

e-mail : M.Sawerwain@issi.uz.zgora.pl
tel. (praca) : 68 328 2321,
pok. 328a A-2,
ul. Prof. Z.Szafrana 2,
65-246 Zielona Góra

Ostatnia kompilacja pliku: Tuesday 6th June, 2023, t: 23:25

Spis treści

Plan wykładu – spotkania tydzień po tygodniu

(1) Informacje o wykładzie, pojęcie platformy, podstawowe informacje o platformie .NET

(2) Składowe platformy .NET: CLR, CTS, języki programowania, biblioteki klas, pojęcie podzespołu (ang. assembly)

(3) Programowanie w C# – środowisko VS, MonoDevelop, syntaktyka C#, wyjątki, współpraca z DLL

(4) Programowanie w C# – model obiektowy, typy uogólnione, lambda wyrażenia

(5) Programowanie w C# – aplikacje „okienkowe", programowanie wielowątkowe

(6) Programowanie w F# – podstawy, przetwarzanie danych tekstowych,

(*) "Klasówka I", czyli egzamin cześć pierwsza

(7) Dostęp do baz danych

Plan wykładu – tydzień po tygodniu

(8) Język zapytań LINQ, Entity Framework
(9) Obsługa standardu XML
(10) Technologia ASP.NET 1/2
(11) Technologia ASP.NET 2/2
(12) Model widok i kontroler – Model View Controller
(13) Tworzenie usług sieciowych SOAP i WCF (komunikacja sieciowa)
(14) Wykład monograficzny .NET 1
(15) Wykład monograficzny .NET 2
(*) "Klasówka II", czyli egzamin cześć druga

Plan wykładu

1. HTTP i (X)HTML
   1. protokół HTTP
   2. format komunikatów HTTP
   3. język opisu HTML
2. ASP.NET
   1. czym jest ASP.NET?
   2. formularze ASP.NET
   3. cykl życia strony
   4. dyrektywy i klasa Page
3. kontrolki w aplikacjach ASP.NET
   1. rodzaje kontrolek ASP.NET
   2. hierarchia kontrolek
   3. błędy i zdarzenia aplikacji ASP.NET

Protokół HTTP – fundament cywilizacji?

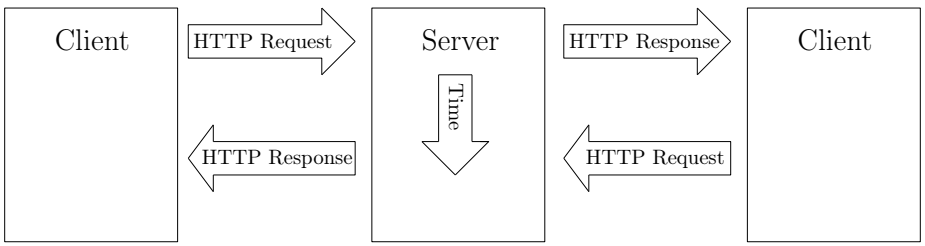
Protokół zaprojektowany jako bezstanowy (nie wyróżnia się grupowanie interakcji (czyli sesji)). Interakcja przeglądarki z serwerem WWW odbywa się według następującego schematu request – response:

- serwer nadsłuchuje żądania,
- klient otwiera połączenie – serwer odpowiada potwierdzeniem,
- żądanie HTTP jest wysyłane przez klienta,
- serwer przekazuje w odpowiedzi żądane zasoby lub informację o ich braku lub braku dostępu,
- połączenie zostaje zamknięte przez serwer.

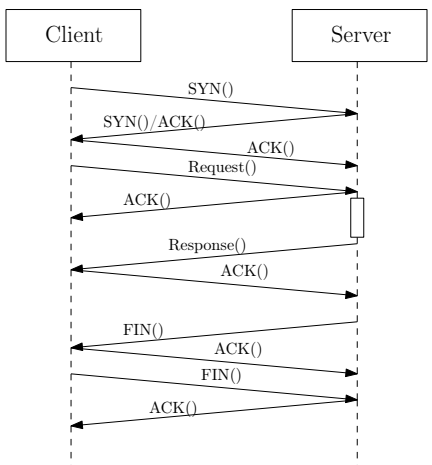
Protokół określa format komunikatu żądania oraz odpowiedzi, domyślny numer portu dla HTTP to 80.

Protokół HTTP – fundament cywilizacji?

Serwer HTTP i klienci:

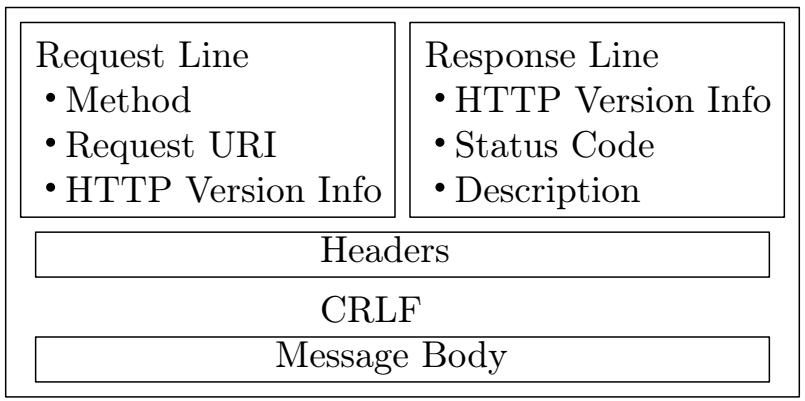


Przetwarzanie danych w protokole HTTP:



Protokół HTTP – fundament cywilizacji?

Opis linii żądania: nazwa metody, ścieżka do metody, wersja protokołu oraz linia odpowiedzi zawiera: wersję protokołu, kod rezultatu, opis w języku naturalnym.



Formaty żądania i odpowiedzi są podobne:

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

Nagłówki żądań dla protokołu HTTP v1.1

Nagłówek	Krótki opis
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 haśle 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	ciasteczka/cookies przesyłane 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, iż 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 przesłała żądanie

Komunikaty i nagłówki odpowiedzi HTTP

Kody statusu	
Kod	Krótki opis
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

Nagłówki odpowiedzi	
Nagłówek	Krótki opis
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 upłynięciu których przeglądarka powinna ponownie odświeżyć stronę. Nagłówek może także zawierać adres URL strony, którą przeglądarka ma pobrać
Set-Cookie	ciasteczko/cookie które przeglądarka powinna zapamiętać
WWW-Authenticate	typ oraz obszar autoryzacji jaki przeglądarka powinna podać w nagłówku **Authorization** przesłanym w kolejnym żądaniu
Server	rodzaj oprogramowania serwera (analogicznie jak **User-Agent**)

(X)HTML– fundamentem cywilizacji?

(X)HTML stanowi obecnie standard w tworzeniu stron WEB, nie ma liczącej się przeglądarki WWW, które „nie rozumie" (X)HTML:

Znacznik	Krótki opis	Przykład

`<html>`	określa początek i koniec strony	`<html>`
		....zawartość pliku HTML
		`</html>`

`<head>`	sekcja specjalna strony zawierająca podstawowe informacje taki jak: tytuł, referencje do źródeł zewnętrznych, i etc.	`<head>`
		....Content goes here
		`</head>`

`<title>`	określa tytuł strony który będzie prezentowany w przeglądarce	`<title>`
		Welcome to Planet Earth!
		`</title>`

`<body>`	określenie początku i końca tzw. ciała strony	`<body>`
		Zawartość strony ...
		`</body>`

`<a>`	odniesienie do innej strony	`<a href=''http://www.uz.zgora.pl''>`
		Odwiedź UZ!
		`</a>`

`<img>`	osadzenie obrazu na stronie	`<img src=''Logo.gif'' />`
`<b>`, `<i>` `<u>`	tekst pogrubiony, pochylony oraz podkreślony	Ten tekst `<b>`jest pogrubiony`</b>` podczas gdy `<i>`ten tekst jest pochylony`</i>`

(X)HTML– fundamentem cywilizacji?

(X)HTML stanowi obecnie standard w tworzeniu stron WEB, nie ma liczącej się przeglądarki WWW, które „nie rozumie" (X)HTML:

Znacznik	Krótki opis	Przykład
`<form> <input> <textarea> <select>`	stosowane do tworzenia formularzy poprzez które użytkownik przekazuje dane na serwer	`<input type=''text'' value=''Some Text'' />`

`<table>` `<tr>`, `<td>`	znaczniki odpowiedzialne za tworzenie tabeli. Tag `<table>` określa tabelę , znacznik `<tr>` oraz `<td>` określą wiersze i komórki tabeli	`<table>`
		`<tr>`
		`<td>`To jest komórka w kolumnie 1`</td>`
		`<td>`To jest komórka w kolumnie 2`</td>`
		`</tr>`
		`</table>`

`<ul> <ol>` `<li>`	Tworzenie list porządkowych oraz bez określonego porządku. Znaczniki `<ul>` oraz `<ol>` tworzą listę (zarówno nieuporządkowaną, prostą oraz uporządkowaną z numerami), znacznik `<li>` reprezentuje poszczególne wpisy na listę	`<ul>`
		`<li>`Pierwszy element (bullet)`</li>`
		`<li>`Drugi element (bullet)`</li>`
		`</ul>`
		`<ol>`
		`<li>`Pierwszy element (numer)`</li>`
		`<li>`Drugi element (numer)`</li>`
		`</ol>`

`<span>`	Pozwala na określenie bloku tekstu gdzie np.: będą obowiązywać inne zasady formatowania.	`<p>`To jest całkiem normalny tekst, podczas gdy `<span style=''color: red;''>`ten tekst będzie czerwony.`</span></p>`
`<div>`	Podobnie jak znacznik `<span>`, tag `<div>` tworzy kontener dla innych elementów. Jednakże, `<div>` funkcjonuje jako element blokowy, co powoduje wstawienie dodatkowego separatora po każdym elemencie `<div>`	`<div>` To jest tekst w jednej linii `</div>` `<div>` Ten tekst jest umieszczony dokładnie pod poprzednią linią.`</div>`

Różnice pomiędzy HTML i XHTML

Podstawowe różnice pomiędzy HTML i nową wersją XHTML:

- znaczniki muszą być zamknięte: `<p></p>`, `<img src=''Logo.gif''/>`, `<br/>`,
- nazwy znaczników i atrybutów powinny być pisane małymi literami,
- wartości atrybutów powinny być objęte cudzysłowami,
- znaczniki powinny być poprawnie zagnieżdżone,
- należy zawsze dodawać deklarację DOCTYPE:

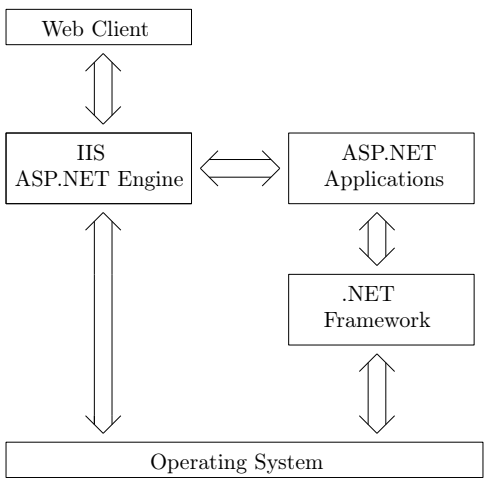
```
<!DOCTYPE html PUBLIC \-//W3C//DTD XHTML 1.0 Transitional//EN"
   \http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Czym jest ASP.NET?

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 interfejsu użytkownika (UI) - strony główne i strony z treścią, zarządzanie stanem, bezpieczeństwem.



Lokalizowanie zasobów — URI, Funkcjonowanie żądań i odpowiedzi – HTTP, Przedstawianie informacji i poruszanie się między zasobami – HTML.

Elementy ASP.NET

Aplikacja ASP.NET może składać się min. z następujących elementów:

- formularze Internetowe (Web Form) – pliki z rozszerzeniem *.aspx
- usługi Web (Web services) – pliki z rozszerzeniem *.asmx
- pliki logiki aplikacji – pliki z kodem źródłowym *.vb albo *.cs
- globalnej klasy aplikacji (.asax)
- plików konfiguracyjnych Web.config
- innych pliki typu: strony HTML, arkusze stylu CSS,

Model programowania obejmuje następujące elementy:

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

Rozwój technologii ASP.NET – 1.x

Pierwsza wersja ASP.NET (v1.x) pozwala na tworzenie aplikacji WEB wykorzystując podejście obiektowe oraz silną kontrolę typów. Do podstawowych właściwości oferowanych przez pierwszą odsłonę ASP.NET należą min.:

- ASP.NET dostarcza tzw. model „code-behind", co pozwala na oddzielenie warstwy prezentacji HTML od warstwy logiki biznesowej implementowanej jako kod (VB, C# code),
- strony ASP.NET są tworzone za pomocą języków platformy .NET (VB, C#), nie jest stosowany oddzielny język (o charakterze skryptowym) po stronie serwera, kod źródłowy jest kompilowany do podzespołów .NET,
- kontrolki WEB pozwalają na budowę interfejsu aplikacji ASP.NET w sposób podobny do aplikacji Windows Forms lub WPF,
- aplikacja ASP.NET samodzielnie zarządza stanem kontrolek podczas tzw. „postbacks" (wykorzystywany jest ukryty formularz __VIEWSTATE),
- aplikacje ASP.NET mogą wykorzystywać wszystkie dostępne podzespoły w ramach BCL i platformy .NET,
- aplikacje ASP.NET mogą być konfigurowane poprzez IIS oraz przez plik konfiguracyjny (Web.config).

Rozwój technologii ASP.NET – 2.0

ASP.NET 2.0 dodaje nowe dodatkowe elementy które wprowadzają dalsze ułatwienia w tworzeniu dynamicznych strony WWW. Lista najważniejszych cech jest następująca:

- dodanie ASP.NET Development Web Server (co oznacza iż nie ma potrzeby stosowania IIS na maszynie na której rozwijana jest aplikacja ASP.NET),
- duża liczna nowych kontrolek WEB min. kontrolki do nawigacji, kontrolki związane z bezpieczeństwem, nowe kontrolki do obsługi danych,
- wprowadzenie tzw. głównej strony (master page) co umożliwia podłączanie różnych wersji UI, do zbioru istniejących podstron,
- wsparcie dla tematów, co ułatwia na łatwe zmienianie wyglądu całej aplikacji sieciowej,
- wsparcie dla tzw. „Web Parts", które umożliwiają użytkownikom aplikacji konfigurację GUI oraz zapamiętywanie ustawień po stronie serwera,
- wprowadzenie opisu konfiguracji zapamiętywanej w plikach Web.config.

Rozwój technologii ASP.NET – 3.5

Nowością w ASP.NET 3.5 jest możliwość stosowania zapytań LINQ oraz następujące elementy związane z aplikacjami WEB:

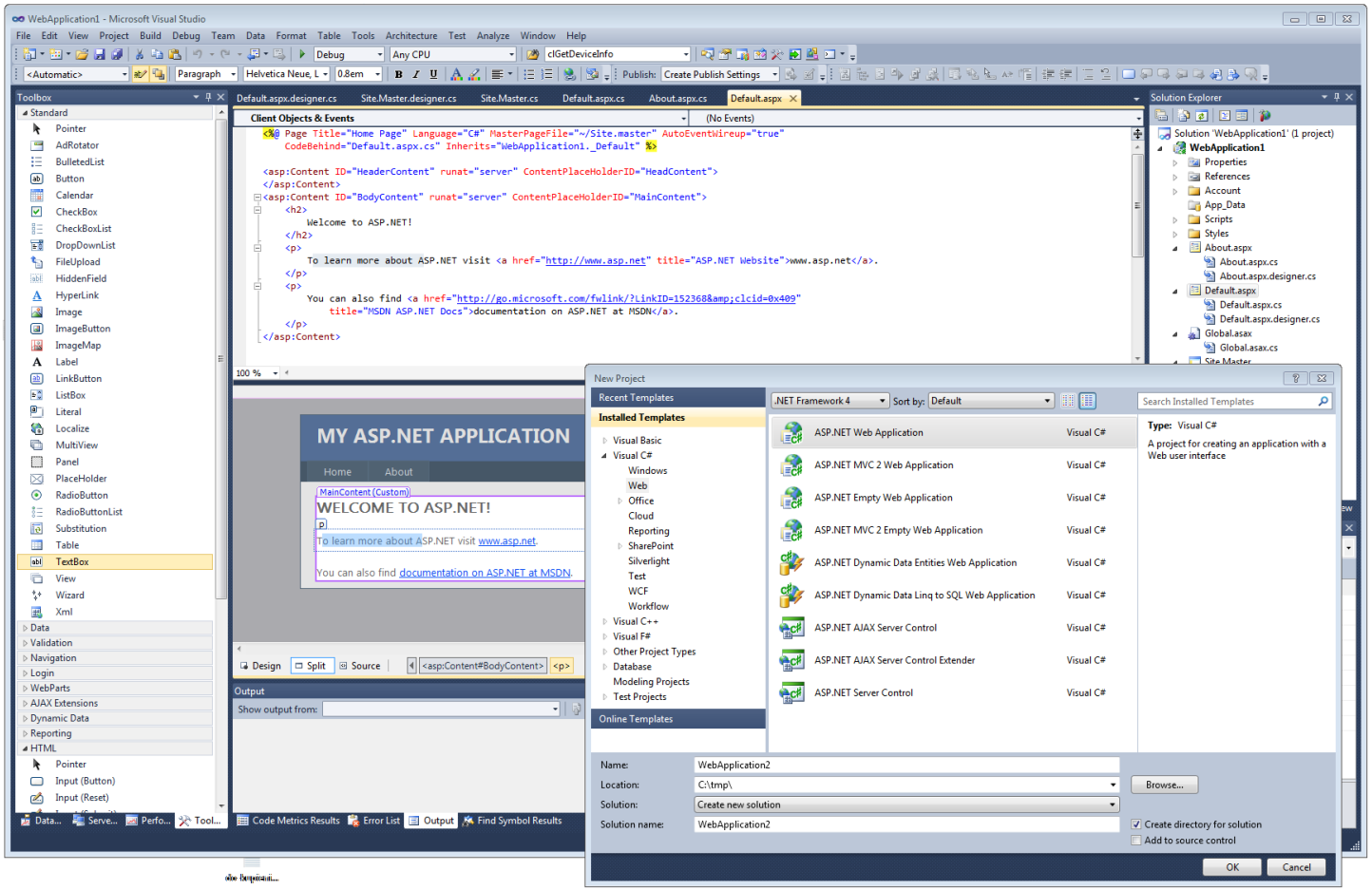
- nowe kontrolki do obsługi technologii Silverlight (aplikacje WPF wspierające zawartość multimedialną),
- wsparcie dla danych pozyskiwanych z klas „ADO.NET Entity",
- wspracie dla danych dynamicznych ASP.NET, jest to rozwiązanie podobne do „Ruby on Rails", dostęp do tabel jest zakodowany jako adresy URI i usługi ASP.NET samodzielnie przenoszą dane na poziom HTML,
- zintegrowane wsparcie dla technologii Ajax, w skrócie dla technologii „micropostbacks" służącej do odświeżania tylko fragmentów strony WEB najszybciej jak to jest możliwe.

Rozwój technologii ASP.NET – 4.0

Najnowsza odmiana platformy .NET 4.0 także przynosi pewne nowe elementy związane z aplikacji WEB, są to min. następujące innowacje:

- możliwość kompresji stanu widoku „view state" za pomocą standardu GZIP,
- uaktualnione definicje przeglądarek, co pozwala na poprawę wizualizacji strony aplikacji na nowych przeglądarkach i urządzeniach (Google Chrome, Apple IPhone, urządzenia BlackBerry, etc.).
- możliwość zmiany ustawień kontrolek za pomocą kaskadowych arkuszy stylów (cascading style sheet – CSS),
- wsparcie dla kontrolek „ASP.NET Chart", co umożliwia budowę stron ASP.NET zawierające wykresy do np.: przedstawiania danych statystycznych oraz finansowych,
- oficjalne wsparcie dla „ASP.NET Model View Controller" (model kontroler-widok), co zmniejsza zależności pomiędzy poziomami aplikacji stosującej szablony modelu kontroler-widok.

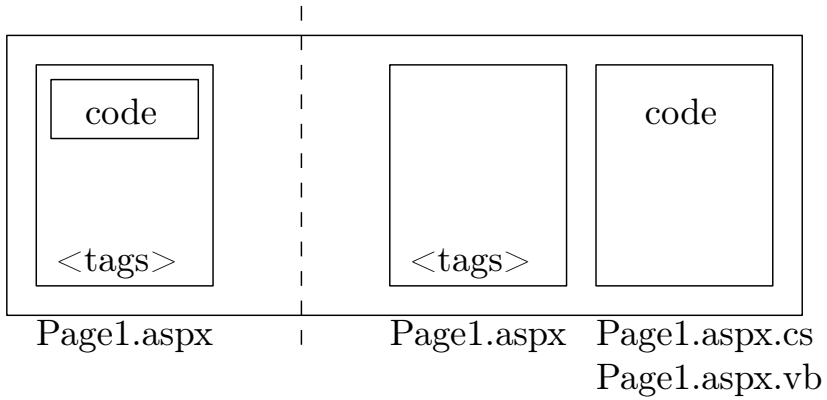
Visual Studio 2010



Najlepsze wsparcie oferuje naturalnie pakiet VS2010 ale do prostych zadań bądź bardzo wyspecjalizowanych zadań wystarczają także darmowy SharpDevelop oraz środowisko MONO (które wspiera ASP.NET do wersji 2.0).

Formularze ASP.NET

ASP.NET pozwala na rozdzielnie kodu logiki od interfejsu użytkownika:



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 kontrolek 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 kontrolek serwerowych, tekstu oraz znaczników HTML, który jest uszczegóławiany przez kod.

Model formularza ASP.NET

Strona ASP.NET jest reprezentowana jako drzewo obiektów:

- page object – obiekt strony
  `<%@Page Language="C#" Debug="False" ...="etc" %>`
  - HTML page
    ```
    <p>To jest jakiś tekst</p>
    <a href="oth1.aspx">link text</a>
    <img src="pic2.gif align="right" />
    ```
  - server control
    ```
    <input type="text" name="mylabel1"
                                    size="40" runat="server" />
    <input type="submit" name="startbtn"
                                value="Start" runat="server" />
    ```
  - other object
    ```
    <%@Import Namespace="System.Data" %>
    <%@Import Namespace="System.XML" %>
    ```

Model formularza ASP.NET

Komponenty formularzy:

- komponent wizualny
  - np.: pola tekstowe bądź kontrolki ekranowe strony (pliki *.aspx)
- logika interfejsu użytkownika
  - kod obsługujący zdarzenia zachodzące na stronie (pliki *.aspx.cs)

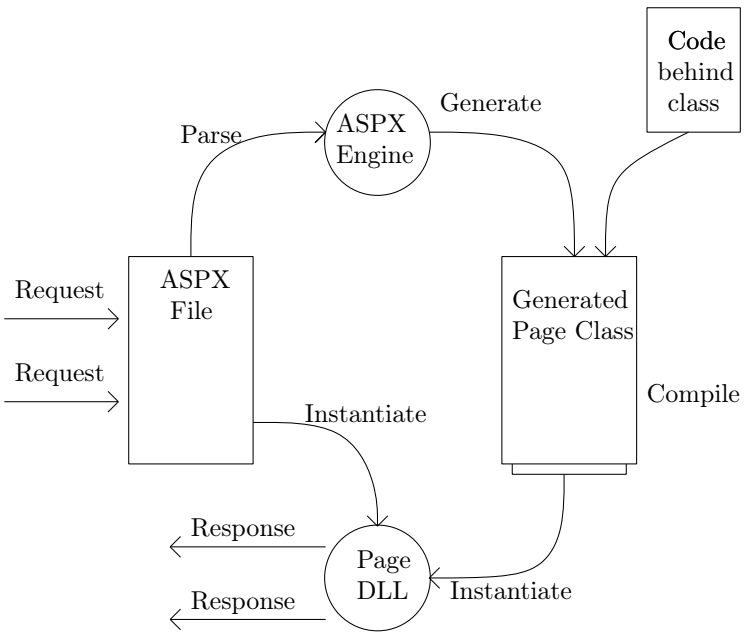
Strony

- Server controls
  - ukrycie procesu tworzenia UI, interakcja z użytkownikiem,
  - zgłaszanie zdarzeń zmieniających stan,
- Realizacja strony:
  - strona zgłasza zdarzenia związane z przetwarzaniem strony,
  - zdarzenia Init, Load, Render, Unload, etc.,
- obsługa zdarzeń
  - obsługa zdarzeń zgłaszanych przez kontrolki na stronie,
  - obsługa może być umieszczona „in-line", lub w oddzielnych stronach bądź bibliotekach DLL,

Przetwarzanie po stronie serwera

- obsługa zachowania kontrolek,
  - deklaratywna, znacznik runat="server"
- utworzenie strony HTML przesyłanej do klienta,
  - obsługa wielu różnych klientów,
  - XHTML, DHTML, HTML 3.2, WML, etc.
- przetwarzanie informacji od klienta
  - łączenie danych z formularza ze źródłem,
  - zgłaszanie zdarzeń „informujących".

Cykl życia strony



Strona jest obiektem klasy System.Web.UI.Page; toteż można korzystać z jej metod i właściwości

- Elementy GUI są obiektami System.Web.UI.WebControls; mamy dostęp do metod i właściwości kontrolek
- 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 pozbawione atrybutu `runat="server"` stanowią kod HTML i aplikacja ASP.NET w niezmienionej prześle je do klienta
- po załadowaniu wszystkich znaczników do pamięci ASP.NET wykonuje odpowiedni kod programu każdej z kontrolek serwerowych,
- po zakończeniu przetwarzania kodu wszystkich kontrolek serwerowych, ASP.Net wywołuje metodę Render każdej kontrolki,
- po utworzeniu strony ASP.NET zwalnia pamięć.

Dyrektywy na strony w formacie ASP.NET

Ogólnie składnia jest następująca:

```
<%@ dyrektywa atrybut=''wartość'' [, atrybut=wartość] %>
```

- `@Page` – definiuje atrybuty strony wykorzystywane przez kompilator stron. Umożliwia określenie parametrów protokołu HTTP, określenie przestrzeni nazw, definicję języka programowania.
- `@Control` – definiuje atrybuty kontrolki użytkownika (user control – UC)
- `@Register` – tworzy powiązanie pomiędzy nazwa pliku kontrolki użytkownika a nazwą odpowiadającego jej znacznika
- ...

Przykłady niektórych arybutów:

- 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,
- ErrorPage - definiuje docelowy URL dla przekierowania, jeśli wystąpi błąd,
- Inheriths - 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 - określenie głównej strony,
- SmartNavigation - odświeżanie tylko tych części formularza które się zmieniły
- Theme - określenie nazwy „tematu” używanego przez stronę.

Strona w formacie ASP.NET – aspx

Pusta strona ASP.NET ze środowiska Visual Studio 2010:

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.master"
    AutoEventWireup="true" CodeBehind="Default.aspx.cs"
    Inherits="WebApplication1._Default" %>

<asp:Content ID="HeaderContent" runat="server"
                                        ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server"
                                        ContentPlaceHolderID="MainContent">
    <h2>Welcome to ASP.NET!</h2>
    <p> To learn more about ASP.NET visit <a href="http://www.asp.net"
                                  title="ASP.NET Website">www.asp.net</a>.
    </p>
    <p> You can also find <a href="..."
          title="MSDN ASP.NET Docs">documentation on ASP.NET at MSDN</a>.
    </p>
</asp:Content>
```

Strona w formacie ASP.NET – aspx.cs

Zawartość plików **Default.aspx.cs** oraz **Default.aspx.designer.cs**:

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

namespace WebApplication1 {
    public partial class _Default : System.Web.UI.Page     {
        protected void Page_Load(object sender, EventArgs e)  {
        }
    }
}
. . . . . .
namespace WebApplication1 {
    public partial class _Default {
    }
}
```

Klasa Page

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

Application oraz **Session** – obiekty reprezentujące stan aplikacji i sesji. **Request** oraz **Response** obiekty reprezentujące obiekty HttpRequest/HttpResponse dla operacji żądania i odpowiedzi generowanych w przestrzeni aplikacji. **Controls** obiekt reprezentujący kontrolki. Własność IsPostBack jest prawdziwa, jeśli strona była już przesłana na serwer, w przypadku gdy żądanie dla strony pojawiło się po raz pierwszy wartością IsPostBack jest fałsz.

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

Metoda **RenderControl** jest odpowiedzialna za odebranie rezultatu serwera i przekazanie zawartości do obiektu writer, zapisywane są także dodatkowe informacje jest włączono śledzenie stanu serwera.

Klasa Page

```
// 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;
  //...

}
```

Własność **IsValid** jest prawdziwa jeśli żaden z obiektów sprawdzających poprawność (ang. validators) nie zgłosił błędów. **TemplateSourceDirectory** wskazuje na aktualny wirtualny katalog. Metoda **Validate()** uruchamia sprawdzanie poprawności strony. **Metoda MapPath(virtPath)** – pokazuje odwzorowania katalogu wirutalnego na katalog fizyczny.

Rodzaje kontrolek serwerowych ASP.NET

Klasyfikacja kontrolek ASP.NET jest następująca:

- Kontrolki serwerowe HTML (HTML server Control) – z atrybutem `runat="server"`
- Kontrolki serwerowe Web (Web Server Control) – `<asp:XY ... runat="server"</asp:XY>`
  - standardowe – etykiety, pola tekstowe, listy, i etc.,
  - danych – odczyt informacji ze źródeł danych
  - nawigacji – 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)

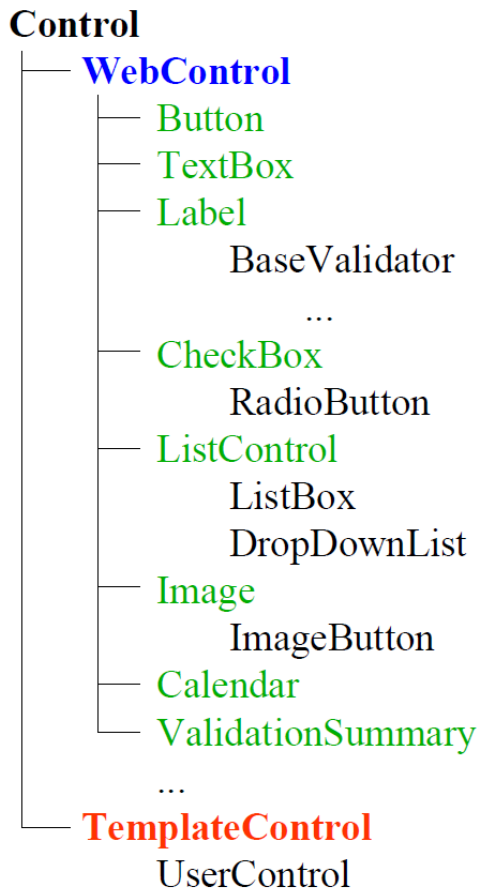
Kontrolki WEB

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

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

<asp:BulletedList BulletStyle="Numbered"
                              DisplayMode="LinkButton"
          ID="BulletedList1" OnClick="BulletedList1_Click"
                                          runat="server">
</asp:BulletedList>
```

- Runat=”server”
  - zdarzenia są obsługiwane przez serwer
  - zapis stanu widoku
- posiadają wbudowaną funkcjonalność
- wspólny model obiektowy
  - wszystkie kontrolki posiadają atrybuty ”Id” oraz ”Text” (odniesienia do tych kontrolek są dostępne po stronie serwera a nazwy obiektów są takie jakie same jak wartości ”Id”)
- tworzone są pliki HTML zgodne z określoną przeglądarką oraz przeprowadzany jest rendering kontrolek,
- dodawane kontrolki są reprezentowane jako zmienne składowe w klasie związane z stroną ASP.



Prosta strona aplikacji WEB

Fragment kodu strony ASP.NET:

```
<form id="form1" runat="server">
<div>
   <b>Środki na koncie:</b>
   <asp:Label ID="TotalLBL" Text="0" Runat="server"/> PLN <br />
   <asp:TextBox ID="AmountTBL" Runat="server"/>
   <asp:Button ID="SendMoneyBTN" Text="Wpłać środki" OnClick="ButtonClick"
                                                          Runat="server" />
   <br />
   Mini stronka testowa!
   <br />
</div>
</form>
```

Prosta strona aplikacji WEB

Kod w C# dla przykładowej strony:

```
namespace WebApplication2 {
    public partial class Site1 {

        protected global::System.Web.UI.WebControls.ContentPlaceHolder head;

        protected global::System.Web.UI.HtmlControls.HtmlForm form1;
        protected global::System.Web.UI.WebControls.Label TotalLBL;

        protected global::System.Web.UI.WebControls.TextBox AmountTBL;
        protected global::System.Web.UI.WebControls.Button SendMoneyBTN;
    }
}
. . . . . .
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace WebApplication2 {
    public partial class Site1 : System.Web.UI.MasterPage {
        protected void Page_Load(object sender, EventArgs e) {
        }
        protected void ButtonClick(object sender, EventArgs e) {
            int totalVal = Convert.ToInt32(TotalLBL.Text);
            int amountVal = Convert.ToInt32(AmountTBL.Text);
            TotalLBL.Text = (totalVal + amountVal).ToString();
        }
    }
}
```

Strona w formacie ASP.NET – aspx

Deklaracja listy numerowanej:

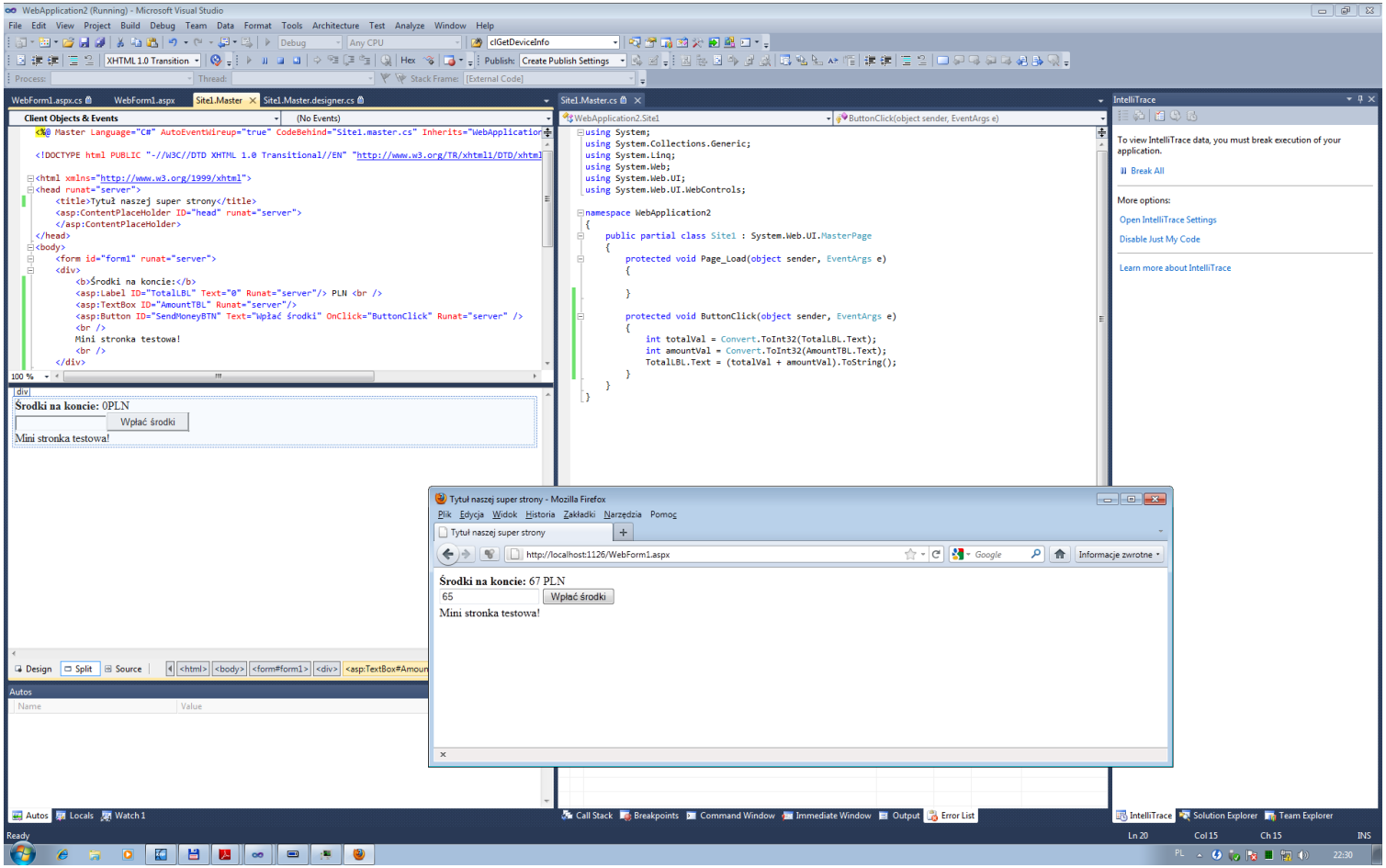
```
<form id="form1" runat="server">
    <div>
        Bullet styles:<br />
        <br />
        <asp:BulletedList BulletStyle="Numbered" DisplayMode="LinkButton"
             ID="BulletedList1" OnClick="BulletedList1_Click" runat="server">
        </asp:BulletedList>
    </div>
    </form>
```

Kod obsługujący powyższą listę, po kliknięciu na element, zmieniany jest styl listy:

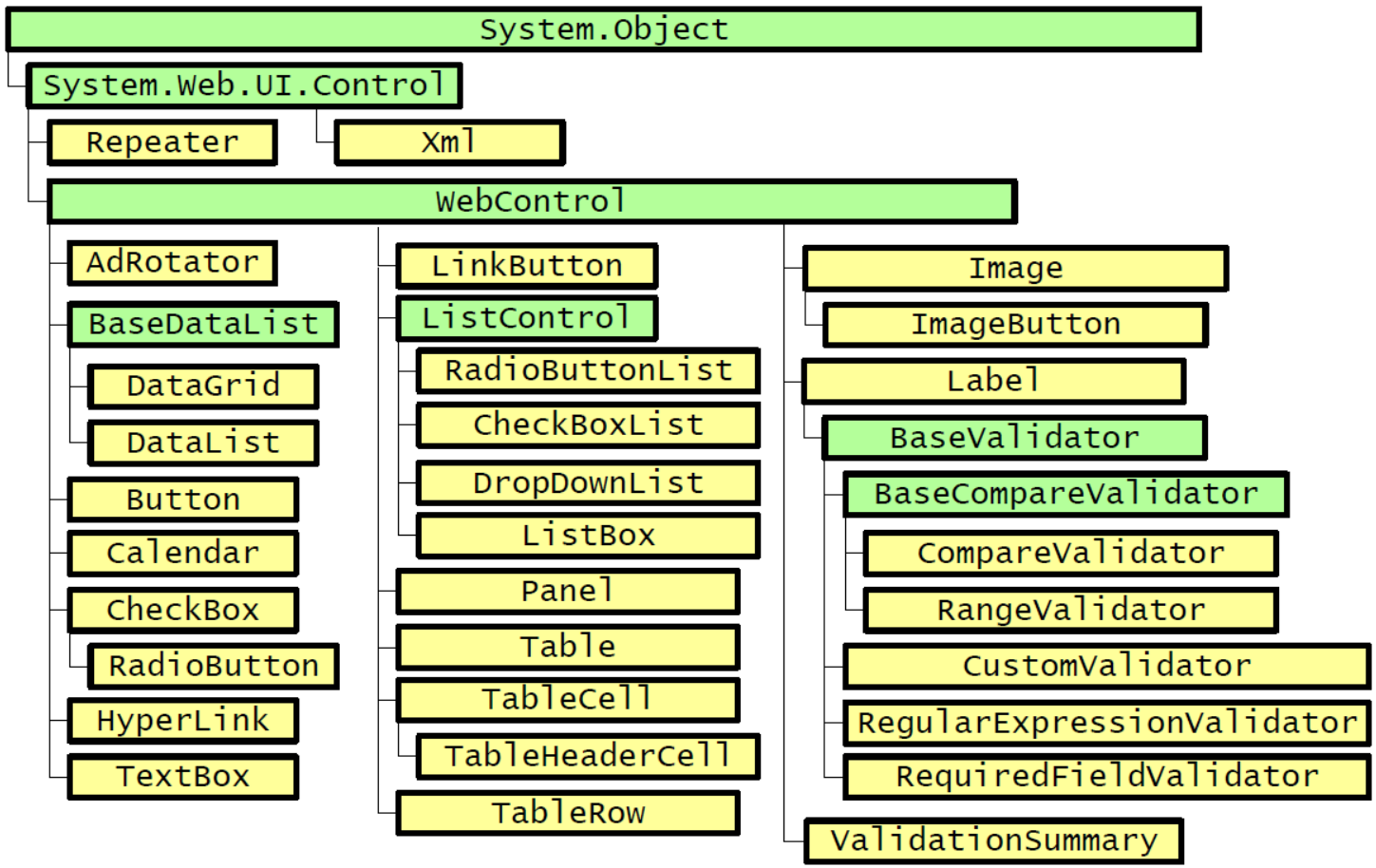
```
protected void Page_Load(object sender, EventArgs e) {
    if (!Page.IsPostBack) {
      foreach (string style in Enum.GetNames(typeof(BulletStyle))) {
        BulletedList1.Items.Add(style);
    }
  }
}

protected void BulletedList1_Click(object sender, BulletedListEventArgs e) {
    string styleName = BulletedList1.Items[e.Index].Text;
    BulletStyle style = (BulletStyle)Enum.Parse(typeof(BulletStyle), styleName);
    BulletedList1.BulletStyle = style;
}
```

Edycja w Visual Studio 2010



Hierarchia kontrolek WEB



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;
   ...
}
```

Własności:

- nazwa kontrolki,
- zagnieżdżone kontrolki,
- kontrolka rodzicielska,
- strona do której przynależy kontrolka,
- czy kontrolka powinna być widzialna,
- stan kontrolki,
- stan będzie stanem trwałem.

Metody:

- czy kontrolka posiada zagnieżdżone kontrolki,
- odszukanie kontrolki o podanym ID,
- wczytanie danych ze źródła danych,
- wczytanie stanu ze strumienia,
- zapis stanu do strumienia,
- narysowanie kontrolki.

Zdarzenia:

- wywołanie po utworzeniu kontrolki,
- po wczytaniu stanu,
- po wywołaniu DataBind,
- przed narysowaniem kontrolki,
- po tzw. zwolnieniu kontrolki.

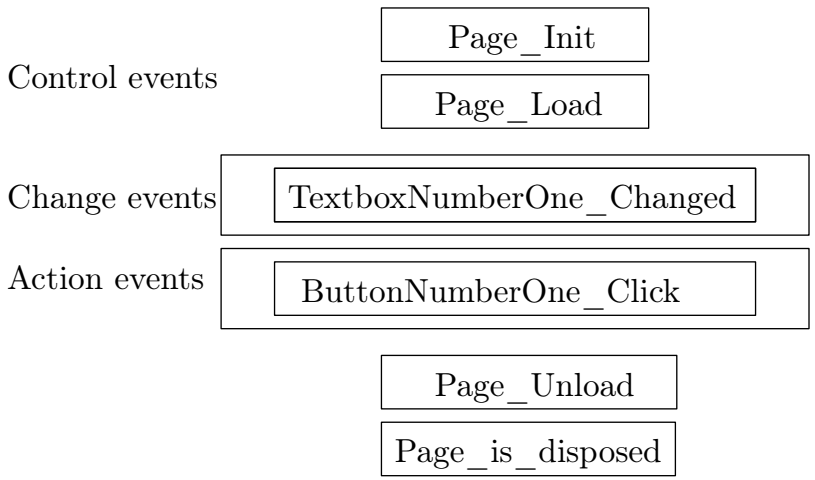
Błędy ASP.NET

Rodzaje błędów w 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

Zdarzenia stron i kontrolek oraz „cykl życia”

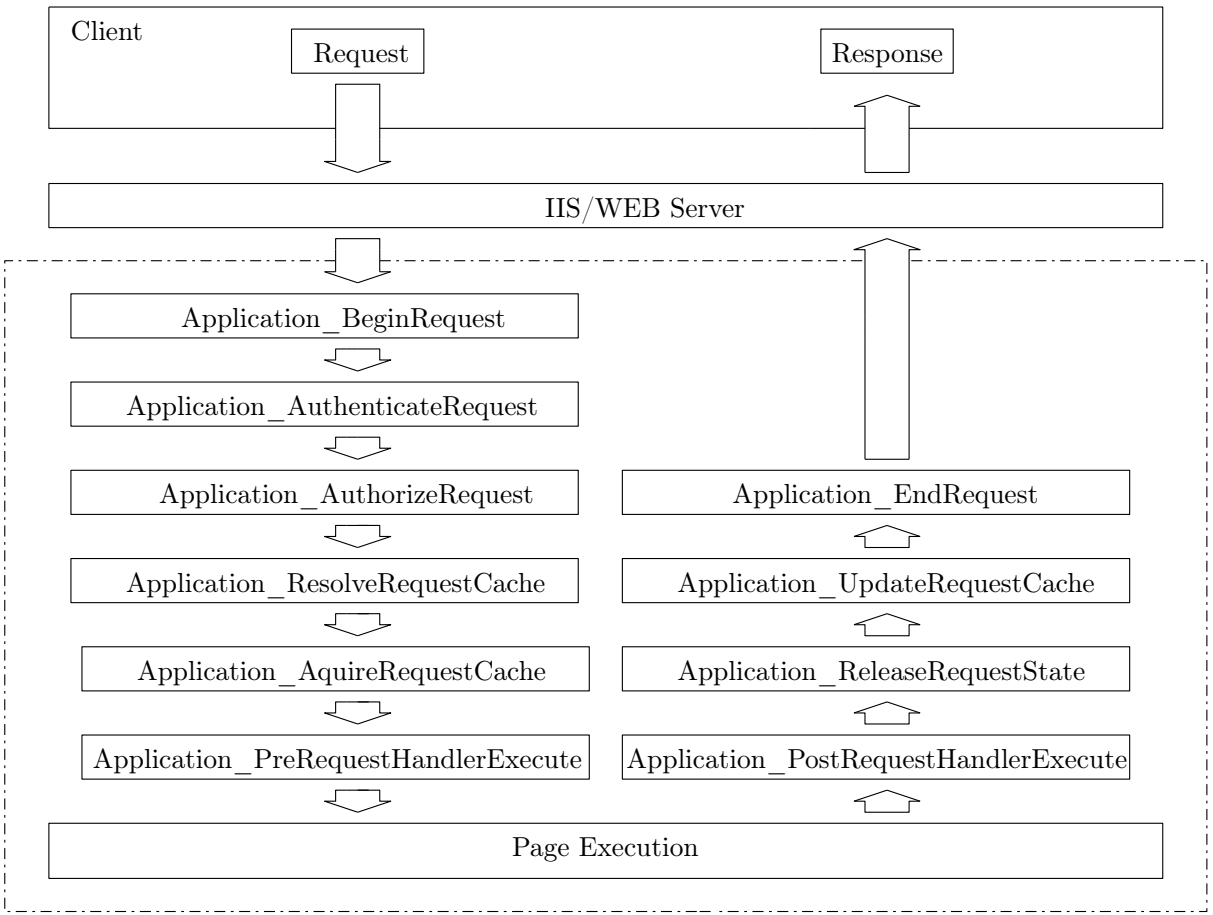
Przykład kolejności zdarzeń:



Przykłady zdarzeń:

Kontrolka	Zdarzenie	Opis
wszystkie	Init, Load, PreRender, Unload	• kiedy kontrolka jest tworzona • po tym jak kontrolka została załadowana do obiektu Page • przed wygenerowaniem kody HTML • przed usunięciem kontrolki z pamięci
Button	Click	jeśli przycisk został kliknięty
TextBox	TextChanged	jeśli zmieniony został tekst
CheckBox	CheckedChanged	jeśli stan CheckBox’a został zmieniony
ListBox	SelectedIndexChanged	jeśli element z listy został wskazany

Zdarzenia request i response



Zdarzenia globalne/warunkowe

- Application_Start
- Application_End
- Application_Error
- Session_OnStart
- Session_OnEnd

Zdarzenia związane z żądaniem

Zdarzenia aplikacji ASP.NET

Przykłady zdarzeń aplikacji ASP.NET:

- 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.

Plik Global obsługujący zdarzenia aplikacji

Plik Global.asax.cs

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

namespace MonoWebApp1 {
  public class Global : System.Web.HttpApplication {

  protected virtual void Application_Start (Object sender, EventArgs e) {  }
  protected virtual void Session_Start (Object sender, EventArgs e) { }
  protected virtual void Application_BeginRequest (Object sender, EventArgs e) { }
  protected virtual void Application_EndRequest (Object sender, EventArgs e) { }
  protected virtual void Application_AuthenticateRequest (Object sender, EventArgs e) { }
  protected virtual void Application_Error (Object sender, EventArgs e) { }
  protected virtual void Session_End (Object sender, EventArgs e) { }
  protected virtual void Application_End (Object sender, EventArgs e) { }
  }
}
```

W następnym tygodniu między innymi

1. nadal o ASP.NET, poprawność stron,
2. strona wzorcowa/master page,
3. zarządzanie stanem,
4. model ASP.NET Page Postback,
5. czym jest technologia MVC i jej korzenie,
6. ASP.NET Model View Controller.

Proponowane tematy prac pisemnych:

1. zalety ASP.NET względem starszych technologii budowania dynamicznych stron WWW,
2. tworzenie własnych kontrolek dla ASP.NET,
3. zagadnienie instalacji aplikacji ASP.NET.

Dziękuje za uwagę!!!