

Projektowanie aplikacji ASP.NET

Wykład 03/15

ASP.NET – elementy wspólne

Wiktor Zychła 2019/2020

Spis treści

1	Podstawowe elementy programowania dynamicznego WWW w ASP.NET	2
1.1	Kontrola strumienia odpowiedzi	2
1.2	Przesyłanie danych na serwer	4
1.3	Przekazywanie parametrów między żądaniami	6
1.4	Architektura stosu aplikacyjnego	6

1 Podstawowe elementy programowania dynamicznego WWW w ASP.NET

Na bieżącym wykładzie zostaną omówione takie elementy ASP.NET które są wspólne dla wszystkich technologii przetwarzania, jakimi będziemy się dalej zajmować. Nie ważne więc czy to jest stare WebForms, współczesne MVC, WCF czy WebAPI, te elementy omówione niżej działają zawsze tak samo.

1.1 Kontrola strumienia odpowiedzi

W poniższym, zaprezentowanym na wykładzie przykładzie używamy tej techniki do wygenerowania obrazka – licznika odwiedzin strony.

```
public partial class Image : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string counterValue = this.Request.QueryString["counter"];
        int w = 120, h = 50;

        this.Response.Clear();

        System.Drawing.Image image = new Bitmap(w, h);
        using (Graphics g = Graphics.FromImage(image))
        {
            StringFormat sf = new StringFormat()
            {
                Alignment = StringAlignment.Center,
                LineAlignment = StringAlignment.Center
            };

            g.Clear(Color.Gray);

            g.DrawString(counterValue, new Font("Tahoma", 14),
                Brushes.Black, new Rectangle(0, 0, w, h), sf);

            image.Save(Response.OutputStream, ImageFormat.Png);
        }

        this.Response.End();
    }
}
```

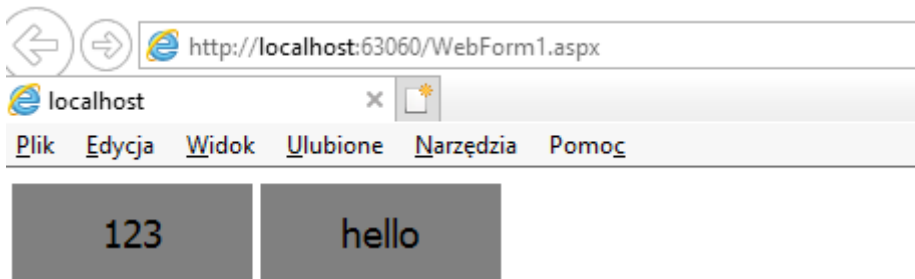
Parametr wartości licznika może być odczytany na serwerze jakkolwiek, w naszym przykładzie jest odczytywany z parametru przekazanego przez przeglądarkę w adresie.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
```

```

<div>
  
  
</div>
</form>
</body>
</html>

```



Przeglądarka interpretuje przesłaną przez serwer odpowiedź na podstawie nagłówka **Content-type**. Zmiana wartości tego nagłówka, w połączeniu z dodatkowym, opcjonalnym dodaniem wartości nagłówka **Content-disposition**, pozwala w pełni kontrolować strumień odpowiedzi.

W szczególności, dla Content-disposition równego **attachment**, zmienia się sposób interpretacji zasobu przez przeglądarkę:

```

public partial class Image : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string counterValue = this.Request.QueryString["counter"];
        int w = 120, h = 50;

        this.Response.Clear();

        this.Response.AppendHeader("Content-
disposition", "attachment; filename=image.png");

        System.Drawing.Image image = new Bitmap(w, h);
        using ( Graphics g = Graphics.FromImage(image))
        {
            StringFormat sf = new StringFormat()
            {
                Alignment = StringAlignment.Center,
                LineAlignment = StringAlignment.Center
            };

            g.Clear(Color.Gray);

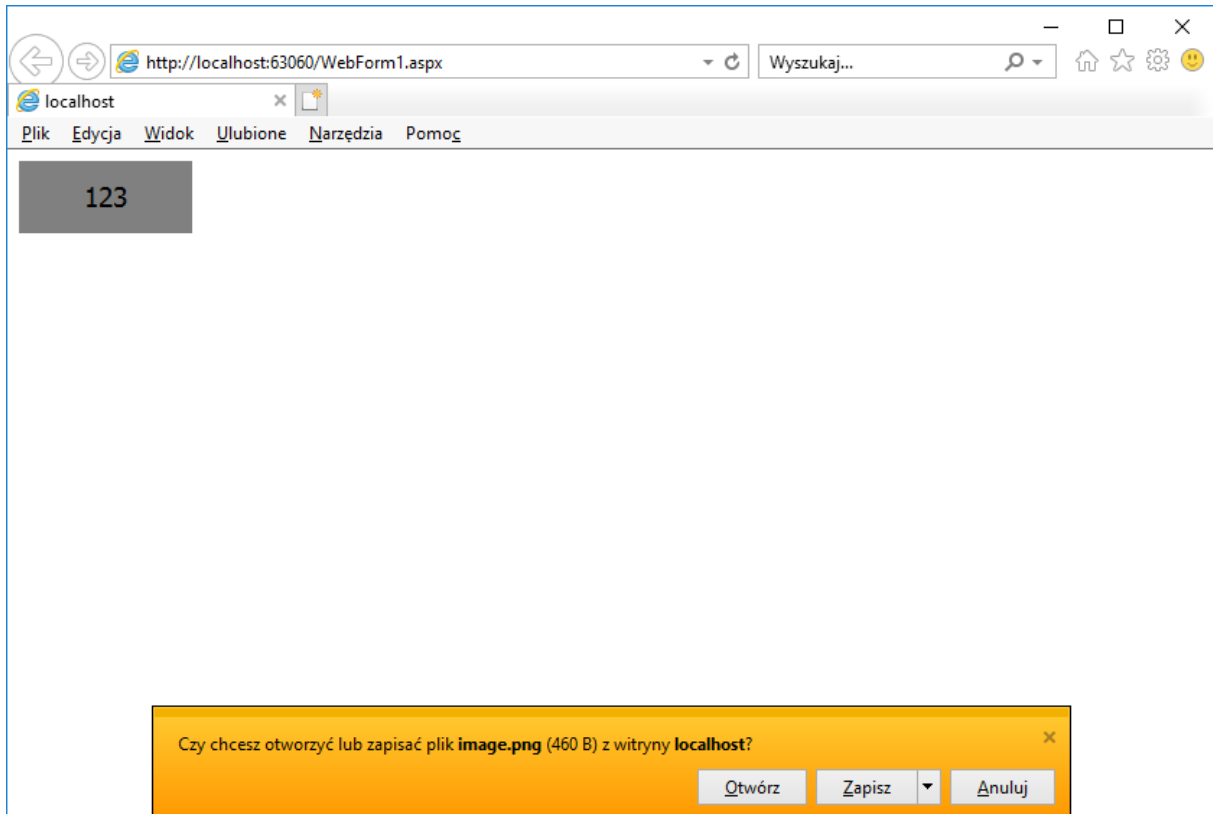
            g.DrawString(counterValue, new Font("Tahoma", 14),
                Brushes.Black, new Rectangle(0, 0, w, h), sf);
        }
    }
}

```

```
        image.Save(Response.OutputStream, ImageFormat.Png);
    }

    this.Response.End();
}
}
```

Przeglądarka zapyta użytkownika co należy zrobić ze wskazanym zasobem:



Tej techniki można używać do generowania z serwera dynamicznych dokumentów które użytkownik powinien móc „pobrać” do lokalnego środowiska, na przykład raportów, wydruków, faktur itd.

1.2 Przesyłanie danych na serwer

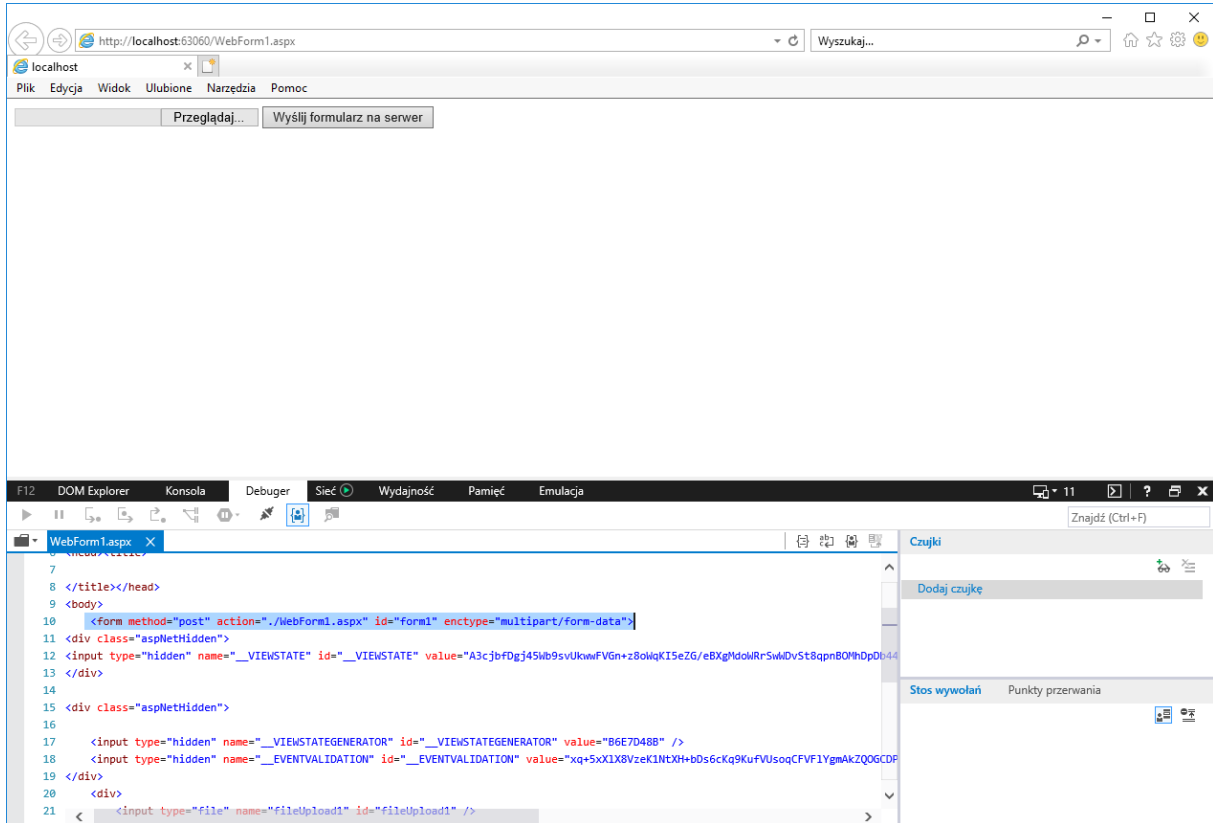
Przesyłanie danych na serwer przez użytkownika możliwe jest za pomocą formantu [FileUpload](#)

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:FileUpload ID="fileUpload1" runat="server" />
            <asp:Button ID="button1" runat="server" Text="Wyślij formularz na serwer" />
        </div>
    </form>
```

```
</body>
</html>
```

który zmienia sposób generowania formularza – formularz otrzymuje specjalny typ zawartości (**enctype**), **multipart/form-data**:



Taki typ zawartości formularza pozwala odesłać zawartość pliku, ponieważ do całego formularza nie jest stosowany standardowy sposób kodowania zawartości (klucz=wartość&klucz=wartość) tylko poszczególne elementy są oddzielone jawnym separatorem

The screenshot shows the Fiddler Web Debugger interface. The top menu includes File, Edit, Rules, Tools, View, and Help. The main window is divided into several panes:

- Request List:** Shows a single request at #2303, Result 200, Protocol HTTP, Host localhost:63060, and URL /WebForm1.aspx.
- Request Details:** Displays the raw request body, which is a multipart form-data containing:
 - Accept-Language: pl
 - User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko
 - Content-Type: multipart/form-data; boundary=-----7e22643b21711d0
 - Content-Length: 939
 - Host: localhost:63060
 - Connection: keep-alive
 - Pragma: no-cache
 - Cookie: _ga=GA1.1.630602421.1533721676
 - Form fields:
 - __VIEWSTATE
 - __VIEWSTATEGENERATOR
 - __EVENTVALIDATION
 - fileUpload1 (filename: "C:\Temp\1.txt")
 - button1 (value: "Wyślij formularz na serwer")
- Response Details:** Shows the raw response body, which is an HTML document:


```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/10.0
X-AspNet-Version: 4.0.30319
X-SourceFiles: #?UTF-8?QzpcvGVtcFxfXENVbnVbVbGCHBSawNhdG1vbJvc2ViqXBwbG1jYXRpb24x
X-Powered-By: ASP.NET
Date: Mon, 29 Oct 2018 15:34:59 GMT
Content-Length: 1026

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>
</title></head>
<body>
<form method="post" action="/WebForm1.aspx" id="form1" enctype="multipart/form-d
<div class="aspNetHidden">
```

1.3 Przekazywanie parametrów między zdaniami

Przekazywanie danych między zdaniami wymaga możliwości utrwalenia stanu i odtworzenia go przy ponownym przetwarzaniu na serwerze. Jest to możliwe za [pomocą szeregu technik](#):

- Obiekt [ViewState](#)
- [HTTP cookies](#)
- [Adres ządania](#)
- Bezpośredni [POST do innej strony](#)
- Kontenery serwerowe: Application, [Session](#), Items

1.4 Architektura stosu aplikacyjnego

Podczas wykładu omówimy przykładowy stos aplikacyjny wykorzystujący pojęcie tzw. [pseudosingletonu](#), w którym warstwa dostępu do danych jest ukryta za fasadą używającą wybranego kontenera (Application, Session, Items) do zarządzania czasem życia obiektu usługowego.