Projektowanie aplikacji ASP.NET

Wykład 03/15
ASP.NET – elementy wspólne

Wiktor Zychla 2024/2025

Spis treści

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.

Oswojenie się z podstawowymi mechanizmami tworzenia interfejsu użytkownika, utrwalania stanu, wykorzystania ciasteczek czy sesji, to podstawowy warsztat, na którym będzie można dalej budować mechanizmy bardziej zaawansowane.

1.1 Kontrola strumienia odpowiedzi

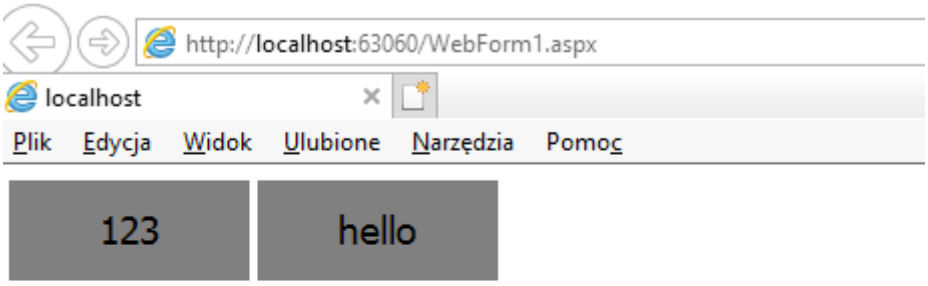
W poniższym, zaprezentowanym na wykładzie przykładzie używamy tej techniki do wygenerowania obrazka – licznika odwiedzin strony.

```
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>
        <img src="Image.aspx?counter=123" />
        <img src="Image.aspx?counter=hello" />
    </div>
    </form>
</body>
</html>
```



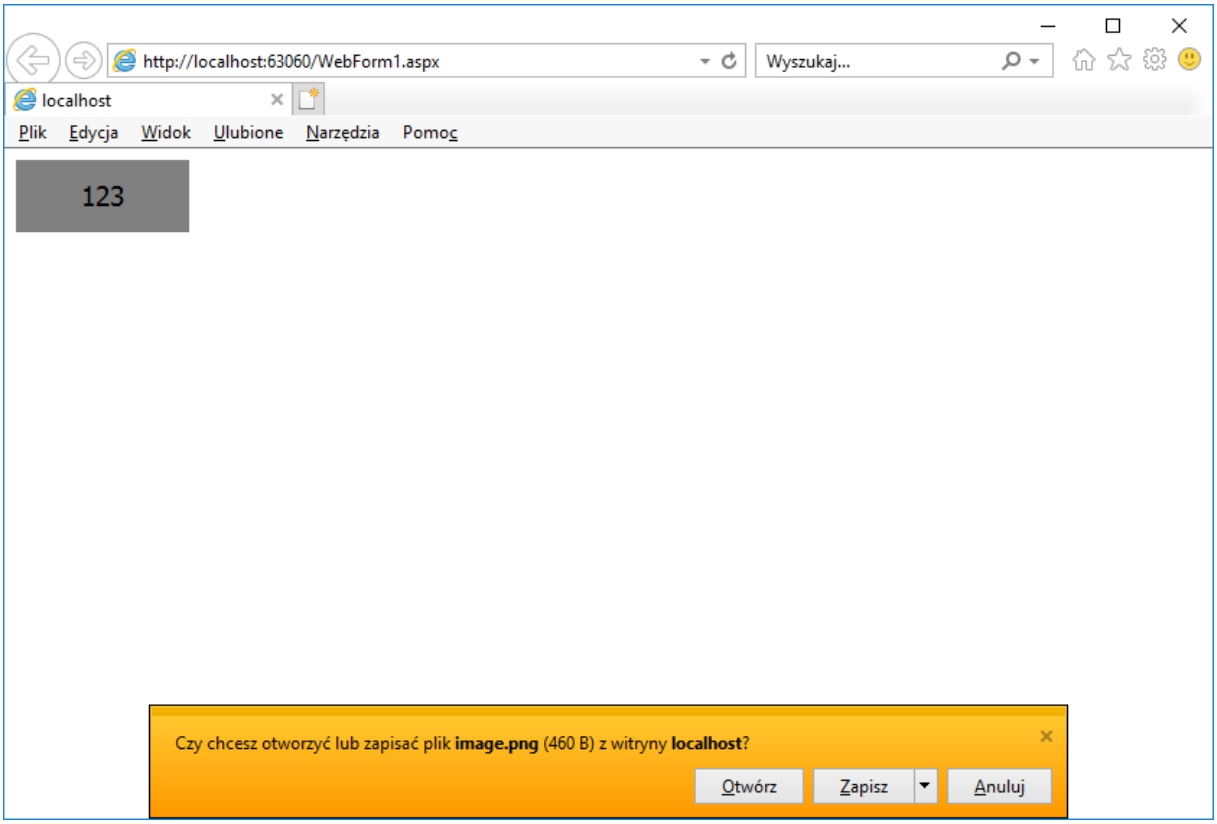
Przeglądarka interpretuje przesłaną przez serwer odpowiedź na podstawie nagłówka **Content-type**. Zmiana wartości tego nagłówka, w połączeniu z dodatkowym, opcjonalnym dodaniem wartości nagłówka **Content-disposition**, pozwala w pełni kontrolować strumień odpowiedzi.

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.

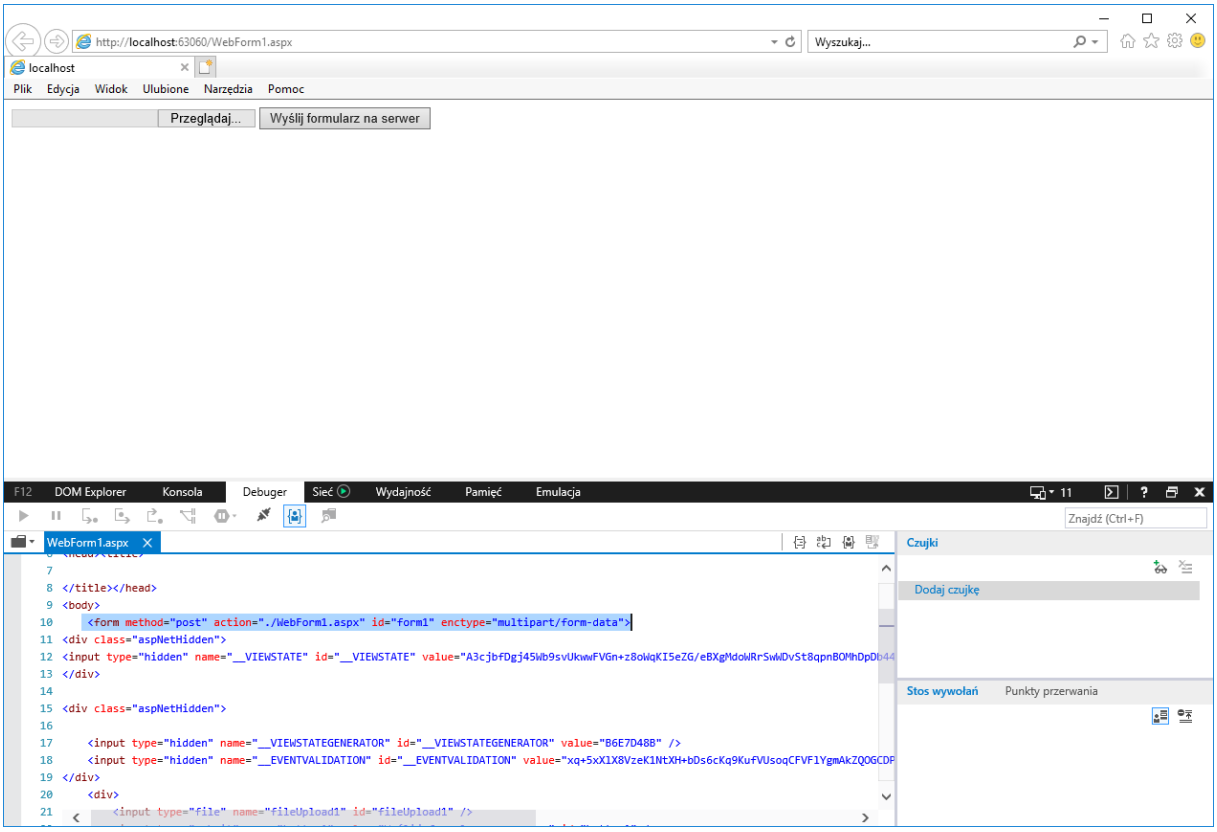
1.2 Przesyłanie danych na serwer

Przesyłanie danych na serwer przez użytkownika możliwe jest za pomocą formantu FileUpload

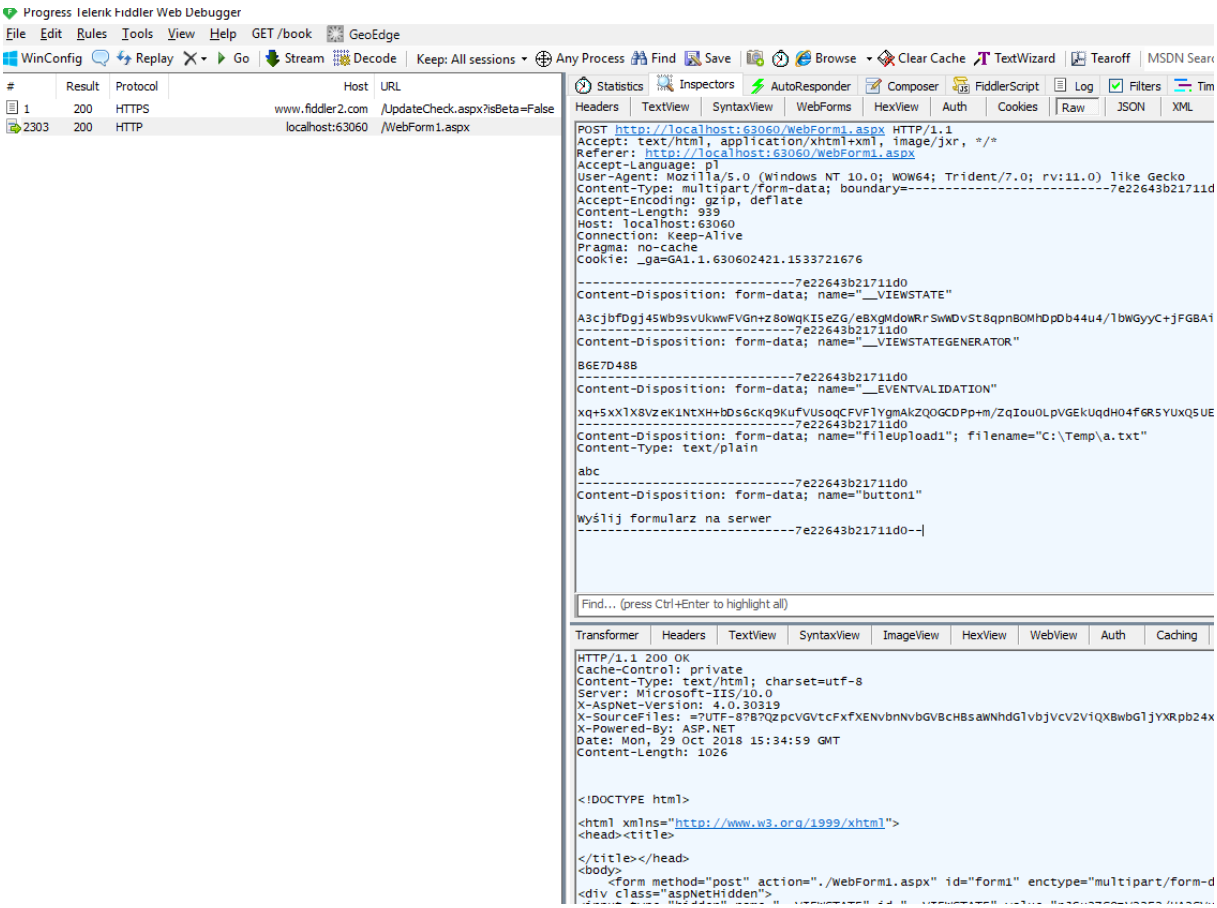
```
<!DOCTYPE html>

<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



1.3 Przekazywanie parametrów między żądaniami – ciastka/sesje/inne

Przekazywanie danych między żądaniami wymaga możliwości utrwalenia stanu i odtworzenia go przy ponownym przetwarzaniu na serwerze. Jest to możliwe za pomocą szeregu technik:

- Obiekt ViewState
- HTTP cookies
- Adres żądania
- Bezpośredni POST do innej strony
- Kontenery serwerowe: Application, Session, Items – użycie prawidłowego kontenera serwerowego jest krytyczne z uwagi na to że użycie niewłaściwego może prowadzić do poważnych nieprawidłowości

1.4 Architektura stosu aplikacyjnego

Podczas wykładu omówimy przykładowy stos aplikacyjny. Rozumiemy tu wykorzystanie zarówno elementów ASP.NET do programowania treści dostarczanych do klienta (przeglądarki) jak również dostępu do serwera bazodanowego.

Istnieje kilka możliwości organizacji kodu. W obu chodzi o udostępnienie obiektu usługowego – takiego, za pomocą którego następuje dostęp do bazy danych. Omówimy dwie techniki:

- Klasa bazowa dla klas obsługujących żądania – w klasie bazowej właściwość enkapsulująca dostęp do bazy danych. W tej konstrukcji czas życia obiektu usługowego jest związany z czasem życia klasy bazowej, a ta jest z kolei związana z czasem życia żądania.
- Konstrukcja tzw. pseudosingletonu, w którym warstwa dostępu do danych jest ukryta za fasadą używającą wybranego kontenera (Application, Session, Items) do zarządzania czasem życia obiektu usługowego

W obu podejściach należy zwrócić uwagę na prawidłową implementację **IDisposable** dla infrastruktury dostępu do danych (zwalnianie zasobów związanych z dostępem do bazy danych) i zweryfikować czy metoda **Dispose** prawidłowo wywołuje się na końcu przetwarzania żądania!