Projektowanie aplikacji ADO.NET + ASP.NET
Zestaw 1

Podstawy ASP.NET

07-10-2014

Liczba punktów do zdobycia: **10/10**
Zestaw ważny do: 21-10-2014

1. (**2p**) Na serwerze IIS 7/8 założyć dwie różne witryny w różnych fizycznych lokalizacjach na dysku. Dla każdej z witryny utworzyć osobną pulę aplikacji. Jako tożsamości puli aplikacji utworzyć i przypisać dwa **różne** konta w systemie operacyjnym.

   Następnie w aplikacji utworzyć jedną stronę `*.aspx`, a w niej (formalnie: w jej kodzie serwerowym C#) spróbować otworzyć do odczytu wskazany plik na dysku (**StreamReader sr = ...**).

   Pokazać, że próba otwarcia pliku wymaga nadania jawnie w systemie operacyjnym uprawnień do pliku (lub foldera zawierającego plik) dla konta określającego tożsamość puli aplikacji.

   Pokazać co się dzieje w przypadku niewystarczających uprawnień (konto puli aplikacji nie ma uprawnień dostępu do pliku).

   Dlaczego nie jest dobrą praktyką przełączanie tożsamości puli aplikacji na LocalSystem?

   Którąś z utworzonych witryn udostępnić na dwóch różnych nagłówkach hosta (np. `ap1.myserver.com` i `ap2.myserver.com`). Pokazać, że witryna działa poprawnie adresowana na dwa różne sposoby. Nauczyć się korzystać z lokalnej mapy hostów (`Windows/System32/Drivers/etc/hosts`).

2. (**1p**) W konfiguracji witryny na serwerze aplikacji umieć wskazać miejsce definiowania filtrów kojarzących rozszerzenia zasobów z logiką ich przetwarzania. Co się stanie jeśli na serwerze aplikacji umieścimy zasób o rozszerzeniu, na które nie mapuje się żaden filtr, a następnie spróbujemy do serwera wysłać żądanie o wydanie tego zasobu?

   Zademonstrować to na przykładzie eksperymentu z zasobem o rozszerzeniu np. ***.foo**: utworzyć w aplikacji zasób o takim rozszerzeniu i spróbować go pobrać z poziomu przeglądarki.

3. (**1p**) Nauczyć się korzystać z debuggera warstwy TCP (np. Wireshark lub Microsoft Network Monitor) do podglądania ruchu klient-serwer na poziomie protokołów transportowych.

   Pokazać jak podglądać ramki TCP/IP i jak podglądać komunikację za pomocą protokołu HTTP.

4. (**2p**) Nauczyć się korzystać z debuggera warstwy HTTP, Fiddlera (`http://www.fiddler2.com/`). Pokazać jak za pomocą Fiddlera symulować żądania typu GET i żądania typu POST do własnej strony ASP.NET (czyli jak wyklikać w Fiddlerze żądanie, które do naszej aplikacji wyśle Fiddler, a nie przeglądarka!).

Czym po stronie kodu aplikacji różni się żądanie GET od POST?

Zademonstrować, że jedną z różnic jest sposób przekazywania parametrów z klienta na serwer. Pokazać jak w kodzie serwerowym odczytać parametry przesłane przez GET a jak te przesłane przez POST.

5. (**2p**) Po stronie serwera używamy formantów serwerowych (`<asp:...>`), po stronie klienta renderują się one jako formanty HTML.

   Pokazać, że zwykłe formanty HTML też mogą być przetwarzanie po stronie serwera (`runat="server"`). Jaki mógłby być z tego pożytek? Zademonstrować to na przykładzie formantu `input`, który zostanie przetworzony po stronie serwera.

   Pokazać, że formantom ASP.NET można w kodzie serwerowym dodawać explicite zdarzenia po stronie klienta (czyli mieć dużą kontrolę nad tym jak po stronie klienta renderuje się formant serwerowy). Do czego tym razem można wykorzystać tą możliwość? Zademonstrować to na przykładzie formantu `<asp:Button>` któremu zostanie przypisane zdarzenie kliknięcia po stronie klienta.

6. (**2p**) Przygotować aplikację ASP.NET WebForms, która pozwala wprowadzić i wydrukować standardowy "pasek zgłoszenia zadań".

   Aplikacja ma składać się z dwóch stron `*.aspx`: formularza zgłoszenia i formularza wydruku.

   Na formularzu zgłoszenia użytkownik aplikacji powinien mieć możliwość wpisania imienia i nazwiska, daty, nazwy zajęć i numeru zestawu oraz kompletu wyników kolejnych deklarowanych 10 zadań z odpowiednią liczbą punktów. Program powinien kontrolować poprawność wpisywanych danych.

   Po zaakceptowaniu formularza zgłoszenia, użytkownik powinien w przeglądarce zobaczyć formularz wydruku: pasek zgłoszenia w postaci możliwej do natychmiastowego wydrukowania.

   Użyć dowolnej, wybranej przez siebie metody przekazywania danych między stronami (GET/POST, ciastka, kontenery serwerowe).

Wiktor Zychla