Projektowanie aplikacji ASP.NET
Zestaw 1

Podstawy ASP.NET

09-10-2018

Liczba punktów do zdobycia: **10/10**
Zestaw ważny do: 30-10-2018

1. (**2p**) Przygotować przykładową aplikację z jedną stronę `*.aspx`, a w niej (formalnie: w jej kodzie serwerowym C#) spróbować otworzyć do odczytu wskazany plik tekstowy z dysku serwera (**StreamReader sr = ...**). Zawartość pliku wypisać na stronie. Jako serwer aplikacji wskazać IIS Express.

   Następnie 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 pul aplikacji utworzyć i przypisać dwa **różne** konta w systemie operacyjnym.

   Pokazać jak przeprowadzić deployment aplikacji rozwijanej lokalnie, w Visual Studio przy pomocy IIS Express, na pełny serwer IIS.

   Pokazać, że w przypadku IIS próba otwarcia pliku w kodzie serwerowym wymaga nadania jawnie w systemie operacyjnym uprawnień do pliku (lub foldera zawierającego plik) dla konta określającego tożsamość puli aplikacji (dlaczego tak się dzieje?).

   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) do podglądania ruchu klient-serwer na poziomie protokołów transportowych.

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

4. (**1p**) Nauczyć się korzystać z debuggera warstwy HTTP, Fiddlera (`http://www.telerik.com/fiddler`).

   Pokazać jak przechwytywać ruch z przeglądarki do internetu - jak podglądać zawartości żądań i odpowiedzi i jak stawiać pułapki i za ich pomocą **modyfikować** żądania i/lub odpowiedzi (zakładka Inspectors).

   Pokazać jak za pomocą Fiddlera wysyłać dowolne 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; zakładka Composer!).

   Czym różni się wysyłanie żądania GET od POST po stronie Fiddlera? Jak odróżnić takie żądania po stronie kodu aplikacji ASP.NET? Pokazać jak w kodzie aplikacji ASP.NET po stronie serwera odczytać parametry przesłane przez GET a jak te przesłane przez POST.

5. (**1p**) Nauczyć się korzystać z debuggera warstwy HTTP, Burpa (`https://portswigger.net/burp/`).

   Pokazać jak przechwytywać ruch z przeglądarki do internetu - jak podglądać zawartości żądań i odpowiedzi i jak stawiać pułapki i za ich pomocą **modyfikować** żądania i/lub odpowiedzi (zakładka Proxy/Options, Proxy/Intercept).

   Pokazać jak za pomocą Burpa wysyłać dowolne żądania typu GET i żądania typu POST do własnej strony ASP.NET (czyli jak wyklikać w Burp żądanie, które do naszej aplikacji wyśle Burp, a nie przeglądarka; zakładka Repeater!).

6. (**1p**) Wyjaśnić różnicę różnymi sposobami przekierowywania żądań:

   (a) hyperlink po stronie przeglądarki

   (b) POST między stronami

   (c) `Response.Redirect` na serwerze

   (d) `Server.Transfer` na serwerze

   Wskazówka: Redirecting Users to Another Web Forms Page.

7. (**3p**) 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.

   Pokazać jak użyć **różnych** metod przekazywania danych między stronami

   (a) `Response.Redirect` z argumentami (strona wyjściowa) oraz `Request.QueryString` (strona docelowa)

   (b) `PostBackUrl` na przycisku (strona wyjściowa) oraz `PreviousPage` (strona docelowa)

   (c) kontener serwerowy `Session`

Jeden punkt za każdą zaprezentowaną metodę.

Wskazówka: How to: Pass Values Between ASP.NET Web Forms Pages.

Wiktor Zychla