

Projektowanie aplikacji ADO.NET + ASP.NET

Zestaw 2

Podstawy ASP.NET

12-10-2010

Liczba punktów do zdobycia: **10/20**

- (1p)** Nauczyć się dodawać, odczytywać i usuwać ciastka (`HttpCookie`) w kodzie po stronie serwera. Jak wiarygodnie sprawdzić czy przeglądarka obsługuje ciastka?
- (1p)** Postudiować interfejsy obiektów `Request`, `Server` i `Response`. Do czego przydaje się statyczna właściwość `HttpContext.Current`?
Jak z `HttpContext.Current` odzyskać referencję do aktualnie przetwarzanej w potoku strony? *Wskazówka: przejrzeć wszystkie składowe obiektu, poszukiwana referencja jest wprost zapisana w jednej z nich.*
- (1p)** Nauczyć się różnic między kontenerami serwerowymi `Application`, `Session` i `Items`. Zademonstrować ich użycie w kodzie po stronie serwera za pomocą opakowań w *pseudosingletony*. Dostęp do którego kontenera musi być dodatkowo chroniony (`lock`) w akcesorze dostępu (`get`) i dlaczego?
- (1p)** Bez względu na sposób dostępu do danych, z kontekstu dostępu do danych zwykle korzystamy przy pomocy jakiegoś dedykowanego obiektu (`SqlConnection`, LINQowy `DataContext`, Hibernate'owy `ISession` itp).
Który kontener serwerowy jest najwłaściwszy do przechowywania takiego kontekstu dostępu do danych (`Application`, `Session` czy `Items`)? Jakie skutki uboczne miałyby przechowywanie kontekstu dostępu do danych nie w tym **właściwym**, ale w którymś z pozostałych kontenerów (co oczywiście technicznie jest możliwe)?
Zademonstrować kod, który opakowuje kontekst dostępu do danych w pseudosingleton przy wykorzystaniu tego właściwego kontenera.
W którym miejscu potoku przetwarzania poprawnie zwalniać zasoby kontekstu dostępu do danych (co jest istotne zwłaszcza wtedy jeśli obiekt dostępowy implementuje interfejs `IDisposable`)? Zademonstrować to zwalnianie na przykładzie.
- (1p)** Do czego służy plik `app_offline.htm`?
- (2p)** Przećwiczyć w praktyce przesyłanie danych binarnych w obie strony. Ściślej - wykonać aplikację, która pozwoli użytkownikowi wskazać plik lokalny na dysku i przesłać go na serwer (formant `<input type="file" ... />`), a po stronie serwera zostanie wyprodukowany i odesłany plik XML:

```
<opis>  
  <nazwa>nazwaprzesłanego pliku</nazwa>
```

```
<rozmiar>rozmiarprzesłanego pliku</rozmiar>  
<sygnatura>suma bajtów pliku modulo 0xFFFF</sygnatura>  
</opis>
```

Plik XML powinien być budowany dynamicznie i odesłany do klienta **bez** zapisywania jego zawartości na dysku serwera. W przeglądarce użytkownika nadesłana odpowiedź powinna spowodować podniesienie się domyślnego okna Otwórz/Zapisz/Anuluj.

7. (3) Wykorzystać obiekt sesji do następującego **naiwnego** rozwiązania problemu autentykacji: w każdej stronie aplikacji w wypadku stwierdzenia w kodzie zdarzenia `Page_Load` braku informacji o użytkowniku w sesji, kontekst przetwarzania przekierowywany jest do strony `Login.aspx`, na której po poprawnym potwierdzeniu tożsamości (login i hasło) informacja o tożsamości zapamiętywana jest w kontenerze sesji.

Dodatkowo punkt wejścia do aplikacji powinien być poprawnie przywrócony po autentykacji (użytkownik kieruje żądanie do którejkolwiek strony aplikacji, jest przekierowany do strony logowania, a następnie aplikacja sama powraca do strony, od której użytkownik chciał rozpocząć nawigację).

*Uwaga! O tym jak **poprawnie** implementować mechanizm autentykacji i autoryzacji będziemy rozmawiać na kolejnych wykładach. To zadanie ma wyłącznie pokazać, że referencyjne rozwiązanie, które poznamy w przyszłości, nie jest jedynym możliwym. W praktyce nie powinno się autentykacji opierać na sesji serwera (dlaczego?).*

Wiktor Zychła