Projektowanie aplikacji ADO.NET + ASP.NET
Zestaw 6

ASP.NET MVC

2024-11-12

Liczba punktów do zdobycia: **6/48**
Zestaw ważny do: 2024-11-26

1. (**2p**) Powtórzyć przedstawiony na wykładzie przykład własnego routera dla ścieżek typu CMS w obu wariantach, .NET.Framework i .NET.Core (każdy za 1 punkt)

2. (**1p**) Pokazać jak działa wstrzykiwanie parametrów do konstruktora kontrolera w MVC Core. Konkretnie - przekazanie obiektu dostępu do danych (np. SqlConnection), do tej pory omawialiśmy m.in. możliwość umieszczenia kodu inicjującego połączenie we własnej klasie bazowej kontrolera. MVC Core daje jednak możliwość przekazywania parametru do konstruktora kontrolera.

   Dodatkowe, niepunktowane pytanie: czy takie przekazywanie parametrów do konstruktora kontrolera jest możliwe w MVC Framework?

3. (**1p**) Wzorując się na przykładzie z wykładu zaimplementować własne rozszerzenie `Login` z dwoma polami tekstowymi na potrzeby logowania użytkowników.

   Do napisania są dwa warianty, wiązanie nazw pól tekstowych wprost:

   ```
   @Html.Login( "UserName", "Password" )
   ```

   i przez składowe modelu:

   ```
   @Html.LoginFor( m => m.UserName, m => m.Password )
   ```

   W obu przypadkach efektem renderowania powinno być

   ```
   <input type="text" name="UserName" />
   <input type="text" name="Password" />
   ```

4. (**1p**) Zademonstrować działanie następujących mechanizmów MVC

   - wskazanie strony layoutowej, dynamiczne wskazanie strony layoutowej (na podstawie jakiegoś warunku w kodzie)
   - mechanizm widoków częściowych (partial views)
   - mechanizm sekcji (z renderowaniem warunkowym)

5. (**1p**) Przygotować własne atrybuty walidacyjne

- walidacja numeru PESEL
- walidator dopuszczający tylko litery alfaberu łacińskiego, polskie litery, cyfry i białe znaki

Dodatkowa (opcjonalna) część zadania - jak sprawić żeby walidator działał zarówno po stronie serwera (co jest niezbędne) jak i po stronie klienta (co jest opcjonalne). Jakie są zalety takiej dodatkowej walidacji po stronie klienta?

Wiktor Zychla