Projektowanie aplikacji ADO.NET + ASP.NET
Zestaw 5

ASP.NET MVC

2015-12-15

Liczba punktów do zdobycia: **12/52**
Zestaw ważny do: pierwsze zajęcia 2015 (5/12-01-2016)

1. (**2p**) Zadanie z paskiem zadań (1.6) powtórzyć w ASP.NET MVC.

2. (**1p**) Zadanie z przesyłaniem danych w obie strony (2.7) powtórzyć w ASP.NET MVC. Zarówno do odebrania strumienia danych w kontrolerze jak i do zwrócenia zawartości do użytkownika użyć elementów infrastruktury MVC.

3. (**2p**) Pokazać przykłady użycia typowych rozszerzeń:

   - `Html.ActionLink`
   - `Html.BeginForm`
   - `Html.CheckBox[For]`
   - `Html.DropDownList[For]`
   - `Html.Password[For]`
   - `Html.RadioButton[For]`
   - `Html.TextBox[For]`
   - `Html.TextArea[For]`

   Do czego służy `Html.Raw`?

4. (**1p**) Pokazać jak wykorzystuje się Forms Authentication do rozwiązania problemu autentykacji i autoryzacji w MVC.

   Formalnie - zaimplementować dostawcę (`MembershipProvider`), akcję logowania i trzy różne widoki: widok nie wymagający autentykacji, widok wymagający jakiejkolwiek autentykacji i widok wymagający autentykacji w konkretnej roli (np. Administrator).

5. (**2p**) 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" />
```

6. (**2p)** Zaimplementować własne wiązanie (binder) które listę pól tekstowych formularza

```
<input type="text" name="textBox1" />
<input type="text" name="textBox2" />
...
<input type="text" name="textBoxN" />
```

zwiąże automatycznie z listą napisów:

```
// wiązanie po nazwie -> textBox1...textBoxN do IEnumerable<string> textBox
[HttpPost]
public ActionResult FooBar( IEnumerable<string> textBox ) {
}
```

7. (**2p)** (**szkielet CMS**) Na wykładzie zademonstrowano przykład własnej reguły routowania i własnego handlera (przykład do pobrania ze strony wykładu). Zmodyfikować ten przykład tak, żeby obsługiwane były tylko dwa rodzaje adresów

   - adresy postaci `witryna/podwitryna/.../strona.html`
   - adresy postaci `witryna/podwitryna/.../podpodwitryna` (przyjąć że nazwa strony to `index.html`)

   (w stosunku do przykładu - nie obsługuje się nazwy klienta na pierwszym segmencie).

   Zmodyfikować regułę routowania tak, żeby zamiast własnego handlera, uruchamiał się wbudowany handler MVC (`MVCRouteHandler`) który spowoduje wykonanie metody `Render` w kontrolerze `PageController`.

   Zawartości stron przechowywać w bazie danych, a w akcji kontrolera - odczytywać i zwracać jako odpowiedź do użytkownika.

   Jak odwzorować strukturę witryn i stron CMS w relacyjnej bazie danych?

Wiktor Zychla