Projektowanie aplikacji ADO.NET + ASP.NET
Zestaw 6

SOAP, REST, ClickOnce

12-01-2015

Liczba punktów do zdobycia: **10/62**
Zestaw ważny do: 27-01-2015

1. (**1p**) Przygotować własną usługę aplikacyjną typu SOAP (ASP.NET WebService lub WCF) i klienta do niej korzystającego bezpośrednio z protokołu HTTP przez `TcpClient` lub `Socket`.

   Wskazówka. `http://netpl.blogspot.com/2007/05/klient-webservices.html`.

2. (**1+1p**) Przygotować usługę ASP.NET WebService (`*.asmx`) o określonym kontrakcie oraz usługę WCF (`*.svc`) o tym samym kontrakcie i wiązaniu `BasicHttpBinding` zwracającą listę osób z bazy danych. Zachować właściwą higienę konstrukcji stosu aplikacyjnego (jak na wykładzie) - między bazą danych a warstwą aplikacyjną klasy dziedzinowe, między warstwą aplikacyjną a klientem klasy DTO (Data Transfer Object).

   Pokazać, że oba typy klas proxy do usług aplikacyjnych (*stare* i *nowe*, czyli te dziedziczące z `SoapHttpClientProtocol` i te dziedziczące z `ClientBase`), mogą odwoływać się zamiennie do obu serwisów. Inaczej - skonstruować oba typy klas proxy i połączyć się każdą z nich do obu serwisów i odczytywać dane.

   Wskazówka. Usługi typu ASMX i WCF używają innej domyślnej konwencji przekazywania nazwy metody do wywołania. ASMX domyślnie oczekuje `http://service.namespace/action`, WCF oczekuje `http://service.namespace/interfacename/action`.

   Ujednolicić można to po obu stronach, na przykład tak:

```
[ServiceContract(Namespace="http://www.foo.bar.qux/customservice")]
public interface ICustomService
{
    [OperationContract( Action = "http://www.foo.bar.qux/customservice/HelloWorld" )]
    string HelloWorld( string Message );
}
```

3. (**1p**) Do usługi z poprzedniego zadania wygenerować proxy za pomocą dowolnej innej technologii wytwarzania oprogramowania (Java, PHP, ...?) i pokazać że WSDL/SOAP rzeczywiście określa dobrą platformę interoperacyjności - składanie usług wykonanych w różnych technologiach jest oczywiste.

4. (**1+1p**) Przygotować własną usługę aplikacyjną typu REST (ASP.NET WebAPI) zwracającą listę osób z bazy danych (jak w jednym z poprzednich zadań) w formacie JSON.

   Przygotować klienta w C# napisanego jakkolwiek (np. `HttpClient`).

   Przygotować klienta w Javascript napisanego za pomocą **jQuery**.

5. (**1p**) Przygotować solution złożone z projektu aplikacji Web i aplikacji ClickOnce. Deployować aplikację ClickOnce na serwer aplikacji do witryny aplikacji Web i pozwolić użytkownikowi uruchomić aplikację ClickOnce z poziomu aplikacji Web przez odpowiednio spreparowany link do zasobu `*.application`.

6. (**1p**) Rozszerzyć poprzednie zadanie o usługę aplikacyjną (Web Service) przy pomocy którego aplikacja ClickOnce będzie odwoływać się do logiki na serwerze aplikacji. Wygenerować klasę proxy po stronie aplikacji ClickOnce i pokazać w jaki sposób dynamicznie konfigurować adres zwrotny usługi aplikacyjnej tak, żeby nie trzeba było rekonfigurować aplikacji ClickOnce ręcznie tylko żeby adres usługi był konstruowany dynamicznie na postawie lokalizacji z której pobrana została aplikacja ClickOnce.

7. (**2p**) Rozszerzyć poprzednie zadanie o współdzielenie tożsamości aplikacji Web i usługi aplikacyjnej - obie powinny być zabezpieczone tym samym ciastkiem Forms Authentication.

   Wskazówka! Pokazany na wykładzie sposób przekazania wartości ciastka do aplikacji opisany jest tu: `http://netpl.blogspot.com/2008/02/clickonce-webservice-and-shared--forms.html`

   Pytanie dodatkowe: jak radzić sobie z wygasaniem ciastka Forms Authentication w przypadku bezczynności użytkownika? Zwykła aplikacja Web przekieruje sterowanie do strony logowania, a co powinna zrobić aplikacja ClickOnce?

Wiktor Zychla