Info

W celu udostępnienia danych przez moduł TelWinWS najlepiej udostępniać te dane poprzez IIS. IIS wspiera wszystkie wersje protokołu SOAP co daje nam pewność, że aplikacja kliencka może być napisana na dowolnej platformie. Druga zaleta to łatwość dopasowania odpowiedniego sposobu weryfikacji użytkownika.

Zapytania do web serwisu opartego o IIS są identyczne jak zapytania do web serwisu w wersji samodzielnej, bez opierania się o serwer IIS.

Poniższy przykład przedstawia sposób pobierania danych z serwisu dla przykładowych zmiennych. Dla tego przykładu użyto sposobu weryfikacji typu „forms". Sposób ten jest rozwinięciem weryfikacji http Basic, z punktu widzenia programowania jest on również trudniejszy.

Przykład pobierania danych meteo

Przykład przygotowano w Visual Studio.

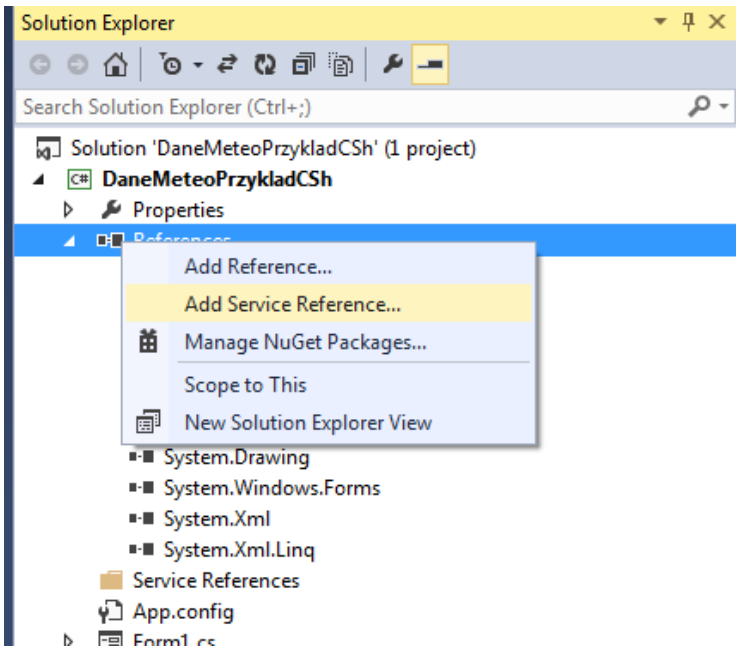
1. Pobranie informacji o serwisie

Standardowo aby dostać informacje o Web Serwis należy pobrać plik wsdl. Jest on dostępny pod adresem:

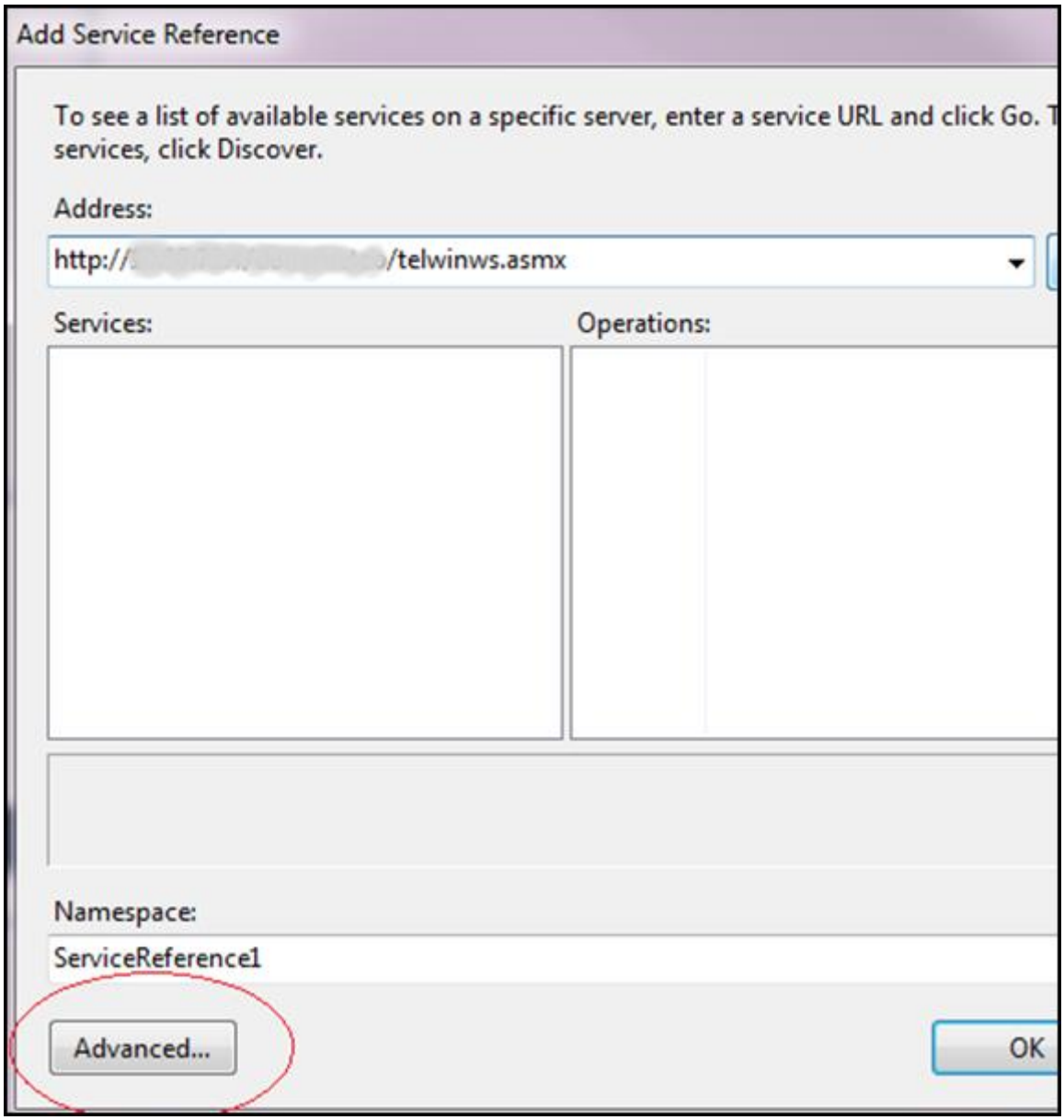
http://host/nazwa/telwinws.asmx?wsdl

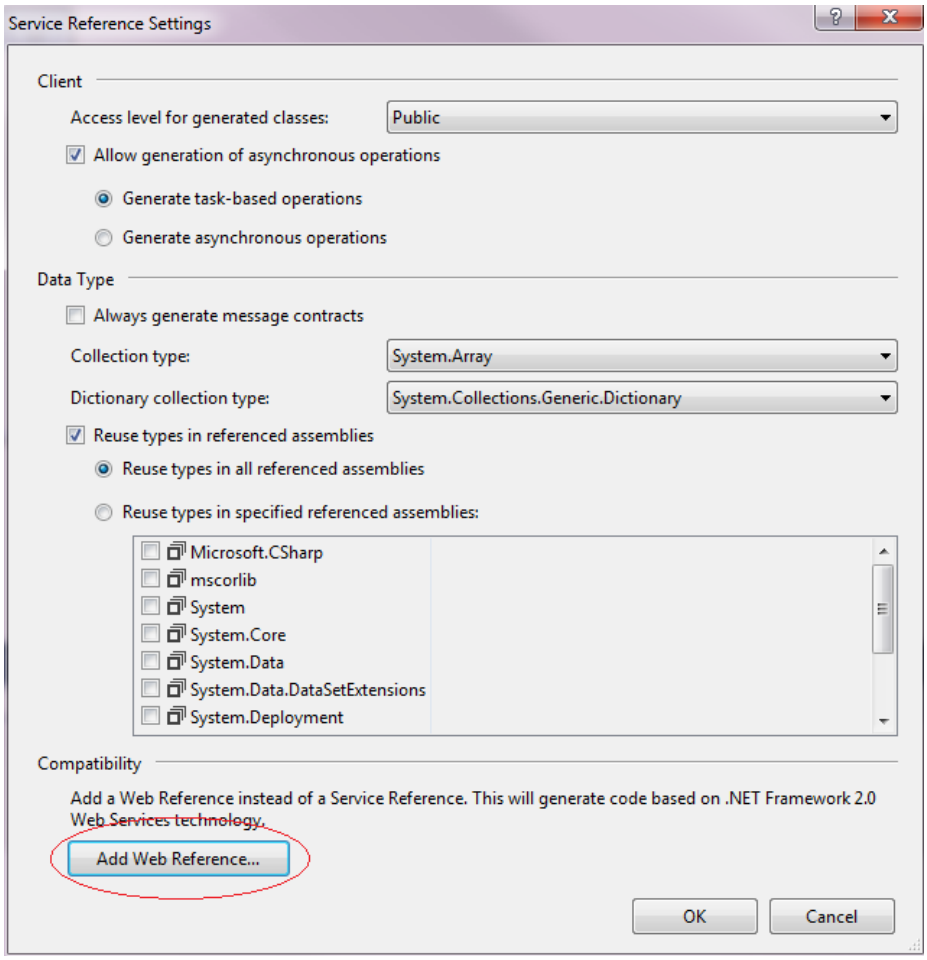
Użytkownik będzie poproszony o podanie hasła.

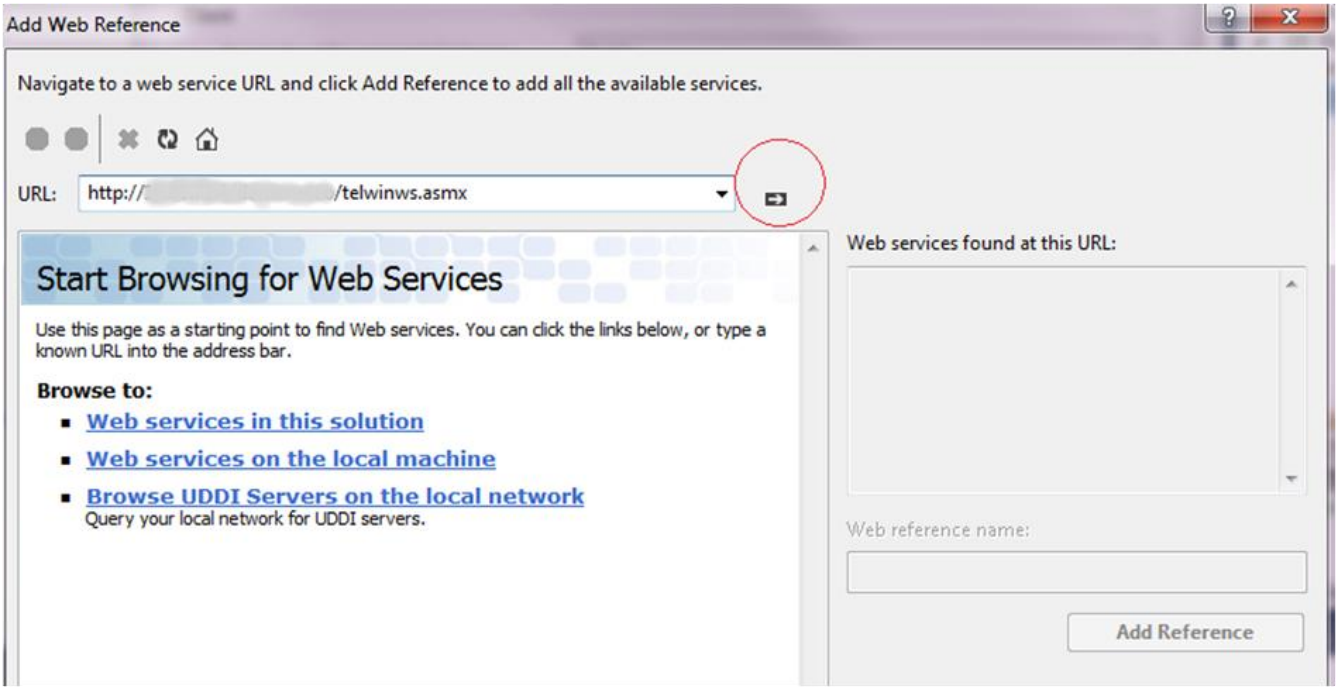
W Visual Studio wygląda to następująco:



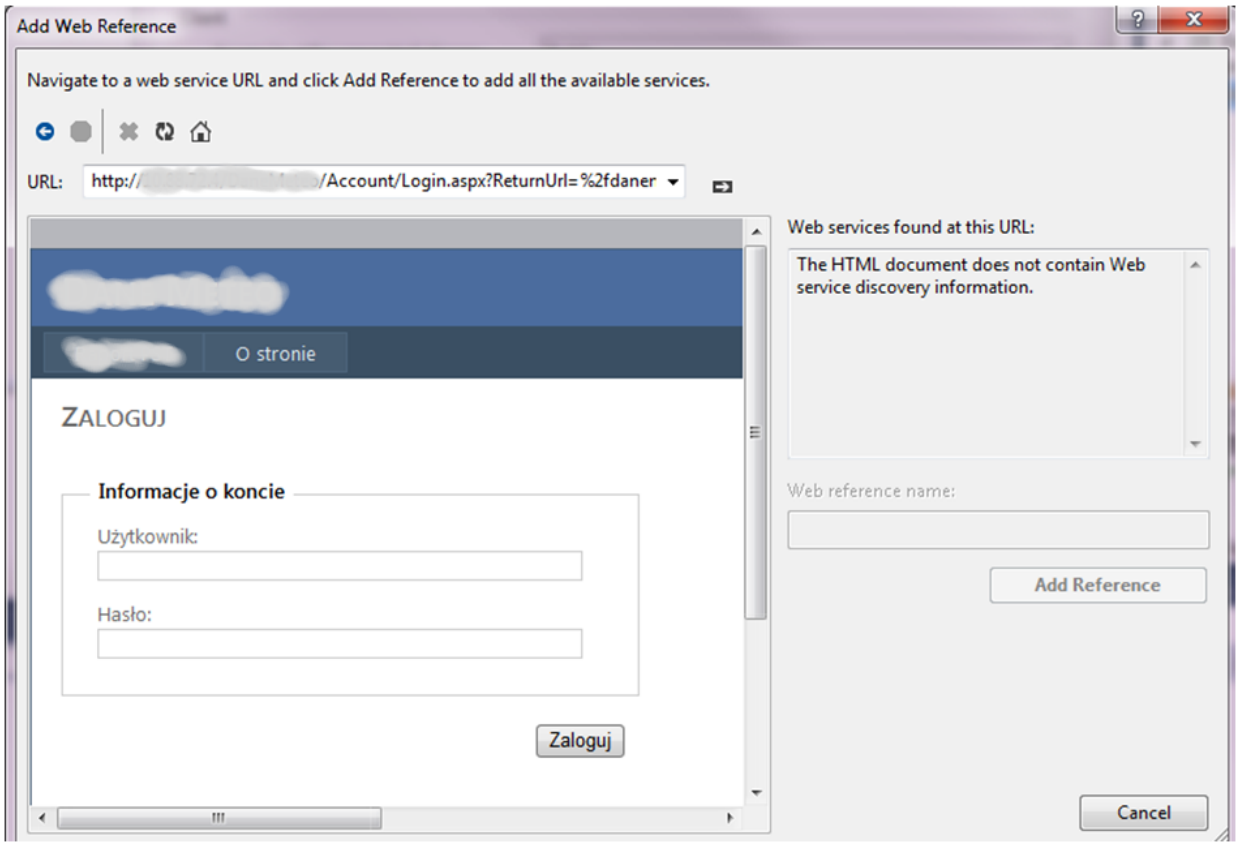
Następnie:

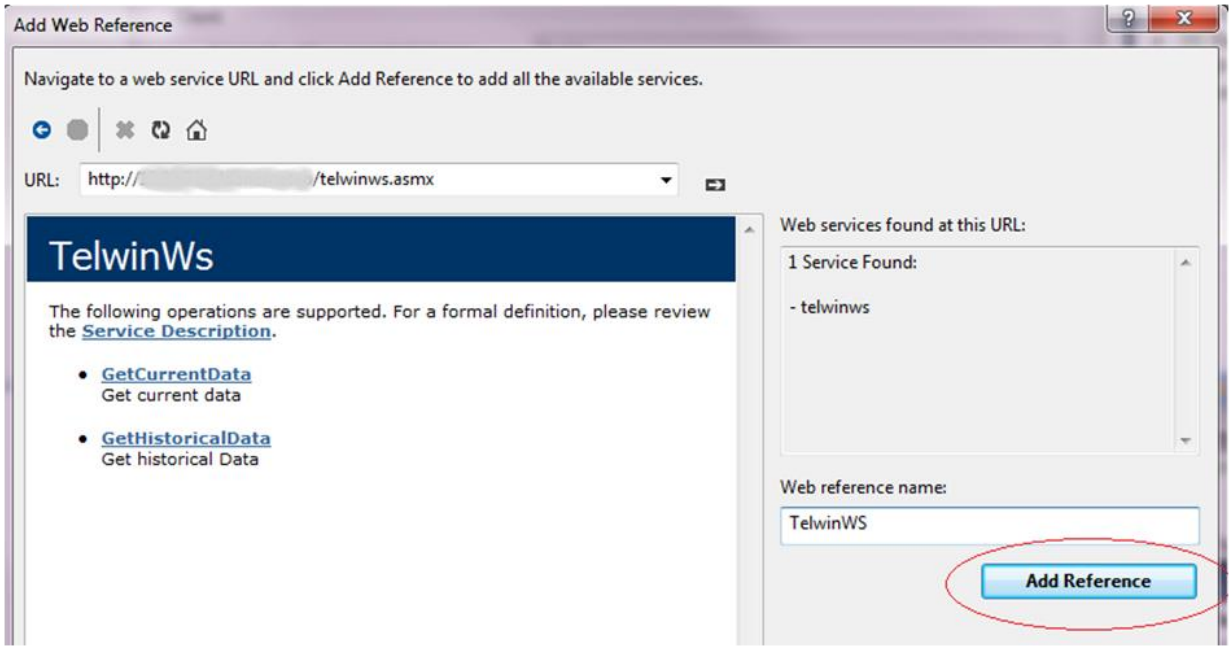

Add Service Reference
To see a list of available services on a specific server, enter a service URL and click Go. T
services, click Discover.
Address:
http://
/telwinws.asmx
Services:
Operations:
Namespace:
ServiceReference1
Advanced...
OK





System wymusza logowanie:





W tym momencie referencja powinna zostać dodana.

Ważna informacja: plik wsdl oraz **opisy** funkcji mogą być niedostępne ponieważ ze względów bezpieczeństwa są wyłączane w IIS. W razie potrzeby należy je uruchomić na czas konfiguracji

Referencja udostępnia dwie funkcje GetCurrentData i GetHistoricalData

Połączenie z web service

Połączenie z Web serwisem jest nie możliwe bez zalogowania się do strony. Przy włączonej autentykacji http Basic sprawa byłaby w miarę prosta ponieważ używamy specjalnej klasy Credentials.

W obecnym rozwiązaniu zastosowano autentykację form, która jest pewną odmianą Basic.

Poniższy przykład napisany w Csharp przedstawia algorytm,który niezależnie (jak sądzę) od platformy programistycznej pozwoli połączyć się z web serwisem:

1. Krok 1: tworzymy obiekt http request. Ustawiamy metodę na POST, natomiast w treści posta przekazujemy użytkownika i hasło. Adresem docelowym będzie strona logowania

```
var cookies = new CookieContainer();
            ServicePointManager.Expect100Continue = false;

            var request =
(HttpWebRequest)WebRequest.Create("http://host/aplikacja/Account/Login.aspx");
            request.CookieContainer = cookies;
            request.Method = "POST";

            request.ContentType = "application/x-www-form-urlencoded";
            using (var requestStream = request.GetRequestStream())
            using (var writer = new StreamWriter(requestStream))
            {
                writer.Write("user=xxxxxx&pass=yyyyy&returnto=/");
            }
            WebResponse response = request.GetResponse();
```

2. Krok 2: jeżeli uda się w trakcie wywołania strony zalogować, wówczas w obiekcie CookieContainer znajdzie się „ticket” umożliwiający dalsze korzystanie z serwisu. Trzeba go przekazać obiektowi, który będzie korzystał z web serwisu.

```
TelwinWS.TelwinWs wsSrv = new TelwinWS.TelwinWs();
            wsSrv.CookieContainer = request.CookieContainer;  //przypisuję uzyskany
cookie
```

3. Krok trzeci: wywołanie metody GetCurrentData:

```
string [] varNames = new string [5];
            varNames[0] = "MGBialystok_tsre";
            varNames[1] = "MGWarszawa_tsre";
```

```
            varNames[2] = "MGRzeszow_tsre";
            varNames[3] = "MGLodz_tsre";
            varNames[4] = "MGPoznan_tsre";


            TelwinWS.WsData[] wsData = wsSrv.GetCurrentData(1, varNames);




            for (int i = 0; i < wsData.Length; i++)
            {
                string szVarName = wsData[i].VarName; //tu mam nazwę zmiennej -
powinno być 5 zmiennych

                //odczyt wartości: Przy pobieraniu wartości bieżących tablica
wsData.Values będzie jednoelementowa, przy pobieraniu danych historycznych -
wieloelementowa
                for (int valIndex = 0; valIndex < wsData[i].Values.Length; valIndex++)
                {
                    TelwinWS.WsVal wsVal = wsData[i].Values[valIndex];

                    //teraz mam dostęp do poszczególnych wartości:
                    int status = wsVal.Status;
                    if (status % 2 > 0)
                        continue; //pomijam wartości o nieparzystych statusach

                    DateTime timeStamp = wsVal.TStamp; //stempel czasu

                    int valType = wsVal.ValType; //typ wartości int, float, string -
odpowiednie wartości opisane w mailu
                    if (valType < 170)
                    {
                        long value = wsVal.ValInt;
                        label1.Text += string.Format("{0}: {1} \r\n",
wsData[i].VarName, value);
                    }
                    else if (valType <= 172)
                    {
                        double value = wsVal.ValFlo;
                        label1.Text += string.Format("{0}: {1} \r\n",
wsData[i].VarName, value);
                    }
                    else
                    {
                        string value = wsVal.ValStr;
                        label1.Text += string.Format("{0}: {1} \r\n",
wsData[i].VarName, value);
                    }

                }


            }

        }
```

TelWinWS

TelwinWs jest oprogramowaniem, które umożliwia dostęp do danych systemu Telwin przy pomocy protokołu SOAP (technologii WebService).

Dane w systemie Telwin widziane są pod postacią zmiennych. Podstawowym atrybutem zmiennej jest jej nazwa, która jest unikalna.

Zmienna może reprezentować następujące rodzaje danych:

- Bieżąca – aktualny pomiar z urządzenia
- Archiwalna – historia zmian wartości zmiennej z określonym kwantem czasowym
- Godzinowa – agregat za określoną godzinę: suma, średnia, max min itp. Rodzaj agregowanej funkcji zależy od definicji zmiennej
- Dobowa – analogicznie jak wyżej
- Miesięcznia

W przypadku danych pogodowych pobierane powinny być dane **dobowe**.

Funkcje:

GetCurrentData

Parametry:

Long idSession – parametr dowolny

String [] varNames – tablica elementów typu string, w których znajdują nazwy zmiennych

Zwraca

WsData [] – macierz elementów WsData - definicja klasy WSData przekazywana jest w pliku wsdl. Opis klasy dalej

GetHistoricalData

Parametry:

Long idSession – parametr dowolny

Int dataType – typ danych o jakie się będziemy pytać:

- Archiwa – 1
- Godzinowe – 2

- Dobowe – 4
- Miesięczne – 8

DateTime – dataStart – okres za jaki chcemy otrzymać dane

DateTime – dateEnd

String [] varNames – tablica elementów typu string, w których znajdują nazwy zmiennych

Zwraca

WsData [] – macierz elementów WsData - analogicznie jak w poprzedniej funkcji.

Jest jeszcze WriteData ale będzie zablokowany.

Struktura klasy WSData

Właściwości klasy (Properties) – w Javie odpowiednikiem są SetValue, GetValue

Istotną właściwością jest **VarName**, gdzie odczytujemy nazwę zmiennej.

Dane składowane są w tablicy **WsVal [].** Jeżeli pytamy dane historyczne to 1 zmienną reprezentuje jeden obiekt WSData, natomiast kolejne wartości zmiennej są w kolekcji WSData.

Pozostałe można pominąć.

Struktura klasy WSVal

Int Status – suma bitowa szesnastu flag. Na tym etapie wystarczy sprawdzić czy wartość jest parzysta. Nieparzysta oznacza błędną wartość

DateTime TStamp stempel czasu

ValFlo - wartość jako liczba zmiennoprzecinkowa 8 bajtowa

ValInt – wartość jako liczba całkowita

ValStr – wartość jako string

Int ValType – typ wartości:

```
  anyVal = 100,
boolVal = 120,
int8Val = 131,
uint8Val = 132,
int16Val = 140,
uint16Val = 141,
```

```
int32Val = 150,
uint32Val = 151,
int64Val = 160,
uint64Val = 161,
float4Val = 170,
float8Val = 171,
float10Val = 172,
strVal = 190,
badVal = 255
```

Wydajność

Należy unikać sytuacji odpytywania zmiennych pojedynczo. Najlepszy efekt uzyskuje się odpytując o większą liczę zmiennych (kilkaset) i większy okres czasu.