

1. Tworzymy w Visual Studio nowy projekt (File->New Project-> Visual C# -> Windows Forms Application).
2. Na formacie kładziemy kilkanaście wierszy kontrolki wg następującego schematu: po lewej TextBox, w środku Button, po prawej TextBox. Każdy wiersz będzie służył do obsługi jednej funkcji, przyciski proszę ponazywać - właściwość Name i podpisać - właściwość Text. TextBoxom pozmieniamy tylko nazwy - właściwość Name. Schemat nazw można przyjąć następujący: dla serwisu Quotes: ServiceReferenceQuotes, txtQuotesInput, btnQuotes,txtQuotesOutput. **Nazwy należy zmieniać przed rozpoczęciem wykonywania konkretnego punktu.**
3. Quotes. Pobrać z serwisu <http://ws.cdyne.com/delayedstockquote/delayedstockquote.asmx> dane akcji wpisanej w polu txtQuotesInput i wyświetlić wartość kursu w polu txtQuotesOutput. Aby to zrobić należy:
  1. Do referencji dodać referencje do powyższego serwisu: Solution Explorer -> PPM na References-> Add Service Reference-> w polu Address podajemy powyższy adres -> Go. Rozwijamy drzewko Services->wybieramy DelayedStockQuoteSoap. W polu Namespace wpisujemy: ServiceReferenceQuotes. OK.
  2. Obok gałęzi References w Solution Explorerze powinna się pojawić nowa gałąź Service References, a w niej powyższa referencja do ServiceReferenceQuotes. Po dwukliknięciu na tą referencje można obejrzeć ją w widoku drzewa, powinna się tam znajdować klasa DelayedStockQuoteSoapClient, z jej funkcji będziemy korzystać w tym i 4 punkcie.
  3. Pod przycisk btnQuotes podpiąć wywołanie funkcji pobierającej wartość pojedynczej akcji (dwuklik na przycisku):
 

```
private void btnQuotes_Click(object sender, EventArgs e)
{
    var serviceClient = new ServiceReferenceQuotes.DelayedStockQuoteSoapClient();
    txtQuotesOutput.Text = serviceClient.GetQuickQuote(txtQuotesInput.Text,
        "0").ToString();
}
```
  4. Jeżeli po odpaleniu otrzymamy taki komunikat: "Nie można załadować sekcji konfiguracji punktu końcowego dla kontraktu „ServiceReferenceQuotes.DelayedStockQuoteSoap”, ponieważ znaleziono więcej niż jedną konfigurację punktu końcowego dla tego kontraktu. Wskaż nazwę preferowanej sekcji konfiguracji punktu końcowego." To należy w pliku App.config odnaleźć odpowiednią sekcję konfiguracji (basicHttpBinding) i z niej skopiować nazwę (w moim przypadku jest to [DelayedStockQuoteSoap](#)). Kod z funkcji należy wtedy zmodyfikować:
 

```
var serviceClient = new
ServiceReferenceQuotes.DelayedStockQuoteSoapClient("DelayedStockQuoteSoap");
```
  5. Przetestować na kilku akcjach, przykładowe kody: GOOG, MSFT, pozostałe (przynajmniej 3) znaleźć w sieci. W komentarzu zapisać kod i wartość:
 

```
//GOOG - 1031.89
//MSFT - 37.57
```
4. Resolve. Adres: <http://ws.cdyne.com/ip2geo/ip2geo.asmx> Services: IP2GeoSoap, Namespace: ServiceResolve, Klasa P2GeoSoapClient.
  1. Kod przycisku:
 

```
var serviceClient = new ServiceResolve.P2GeoSoapClient("IP2GeoSoap");
var ipData = serviceClient.ResolveIP(txtResolveInput.Text, "0");
txtResolveOutput.Text = ipData.City;
```
  2. Zapisać wyniki dla 5 numerów IP (nasz IP uczelniany, 4 adresy popularnych serwisów)- Nazwa – IP - Miasto
 

```
//Onet - 213.180.141.140 - Grupa
```
5. Time. Adres <https://api.efxnow.com/demowebservices2.8/service.asmx?op=GetTime>
  1. Dla tego serwisu kod opracować samodzielnie (analogicznie do poprzednich), w tekstowym polu wyjściowym zaprezentować czas pobrany z serwisu. Metoda nie przyjmuje żadnych argumentów.
6. Echo. Adres <https://api.efxnow.com/demowebservices2.8/service.asmx?op=Echo>
  1. Dla tego serwisu kod opracować samodzielnie (analogicznie do poprzednich), w tekstowym polu wyjściowym zaprezentować wartość pobraną z serwisu. Metoda przyjmuje jeden argument, który jest zwracany wraz ze swoim kodowaniem ASCII.
7. Serwis zamówień publicznych. Adres [http://websrv.bzp.uzp.gov.pl/BZP\\_PublicWebService.asmx?WSDL](http://websrv.bzp.uzp.gov.pl/BZP_PublicWebService.asmx?WSDL)
  1. Dla tego serwisu kod opracować samodzielnie (analogicznie do poprzednich), w tekstowym polu wyjściowym zaprezentować listę wartości połączonych znakiem nowej linii. Przykładowe wywołanie znajduje się w pliku ZadErrata.txt.

**Uwaga:** jeżeli podany poniżej (w punkcie 8) adres nie będzie działał, poniżej 2 adresy, z których również można pobrać dane pogodowe (openweather wymaga rejestracji, odpowiadają po kilku godzinach):

<https://developer.yahoo.com/weather/>

`select * from weather.forecast where woeid in (select woeid from geo.places(1) where text="Szczecin, Poland")`

<http://openweathermap.org/city/3083829>

<http://openweathermap.org/appid>

8. Weather , Adres: <http://www.websvcx.com/globalweather.asmx?WSDL> Service: GlobalWeatherSoap  
Namespace: ServiceWeather.

1. Dostępne w Polsce miasta:  
//Gdansk-Rebiechowo, Krakow, Koszalin, Katowice, Poznan, Rzeszow-Jasionka,  
Szczecin, Warszawa-Okecie, Wroclaw Ii, Zielona Gora

2. Samemu wywołać metodę GetWeather

3. Podać temperaturę w 3 polskich i 2 dowolnych zagranicznych miastach.

9. Do wszystkich wywołań dodać pomiar czasu i wyświetlić go w dodatkowych textboxach (dodanych po prawej stronie od dotychczasowych).

1. Kod przed wywołaniem funkcji z webservice

```
var watch = new Stopwatch();  
watch.Start();
```

2. Po wywołaniu funkcji:

```
watch.Stop();  
txtQuotesFullWatch.Text = watch.ElapsedMilliseconds.ToString();
```

10. Dodać jeden przycisk, który:

1. Sprawdzi, czy wszystkie pola potrzebne do wykonania poszczególnych funkcji są wypełnione (jeśli nie to nie wykonujemy żadnej funkcji):

1. Przykładowy kod

```
private void btnAllFunctions_Click(object sender, EventArgs e)
```

```
{
```

```
    if (txtQuotesInput.Text == "")
```

```
    {
```

```
        MessageBox.Show("Nie wypełnione pole ze wskazaniem akcji (Quotes)", "Brak  
danych", MessageBoxButtons.OK);
```

```
        return;
```

```
    }
```

```
    if (txtQuotesFullInput.Text == "")
```

```
    {
```

```
        MessageBox.Show("Nie wypełnione pole ze wskazaniem akcji do pobrania pełnej  
informacji (QuotesFull)", "Brak danych", MessageBoxButtons.OK);
```

```
        return;
```

```
    }
```

```
}
```

2. Rozpocznie pomiar czasu jak w pkt. 9.1.

3. odpali wszystkie funkcje:

1. przykładowy kod:

```
btnQuotes_Click(sender, e);
```

```
btnQuotesFull_Click(sender, e);
```

4. Pokaże czas trwania wszystkich wywołań (jak w pkt. 9.2).

11. Do istniejącego programu dokładamy nowe kontrolki:

1. proszę podpisać wszystkie przyciski zgodnie z ich przeznaczeniem,

2. służy do tego pole Text we właściwościach.

3. poszczególne funkcjonalności oddzielić od siebie i umieścić je w podpisanych ramkach (obiekt GroupBox, właściwość Text -> podpis),

12. Odczyt XMLa z pogodą

1. powiększyć ramkę obsługującą Webservice pogodowy,

2. dołożyć tam kilka pól tekstowych,

3. główne pole tekstowe rozszerzyć, ustawić w nim właściwość Multiline na true, rozciągnąć w dół na około 6-8 wierszy tekstu,

4. dołożyć dodatkowe pole tekstowe na kod kraju

5. wpisać na stałe (właściwość Text) do tego pola kod kraju: Polska

6. wpisać na stałe (właściwość Text) do pola miasto: Szczecin

7. w kodzie, po wywołaniu i wpisaniu wyniku pogody do pola tekstowego, zapamiętać wynik do zmiennej:

```
var myXml= txtWeatheOutput.Text;
```

8. przetworzyć zawarty w tej zmiennej dokument xml,

```
var xdoc = XDocument.Load(new StringReader(myXml));
```

```
var entry = from x in xdoc.Descendants("CurrentWeather")
```

```
select new
```

```

    {
        Location = (string)x.Element("Location"),
        Time = (string)x.Element("Time"),
        Wind = (string)x.Element("Wind"),
        Visibility = (string)x.Element("Visibility"),
        SkyConditions = (string)x.Element("SkyConditions"),
        Temperature = (string)x.Element("Temperature"),
        DewPoint = (string)x.Element("DewPoint"),
        RelativeHumidity = (string)x.Element("RelativeHumidity"),
        Pressure = (string)x.Element("Pressure")
    };

```

9. wartości wynikowe wpisać do poszczególnych pól tekstowych (przynajmniej 8 wartości), np.:  
`txtWaetherOutputLocation.Text = entry.First().Location;`
13. wszystkie pola tekstowe:
  1. rozszerzyć (żeby wartości wyników się mieściły),
  2. ładnie poukładać,
  3. przed każdym dodać kontrolkę Label i ustawić na niej (oczywiście właściwość Text) opis pola tekstowego.
14. jeżeli temperatura jest ujemna, tło pola z temperaturą ma być czerwone, jeżeli dodatnia od 0st.C do 5st.C to pole ma być niebieskie, jeżeli powyżej 5 st. C to pole ma być zielone:

```

if (temperature < 0 )
    txtQuotesFullInput.BackColor = Color.Red;

```

1. na początku pliku: `using System.Drawing;`
15. Do klasy (na tym samym poziomie co funkcje) dodać tyle zmiennych ile mamy wywołań webserwisów, np.:
  1. `private Task<Decimal> taskQuote;`
  2. `private Task<double> taskTemperature;`
  3. Uwaga! Trzeba uwzględnić typ zwracany przez funkcję (xml w serwisie temperaturowym to string).
16. Dodać przycisk do wywołań asynchronicznych.

1. Podpiąć pod ten przycisk wywołanie dwóch metod i pomiar czasu:

```

var watch = new Stopwatch();
watch.Start();
StartTasks();
FinishTasks();
watch.Stop();
txtAsyncWatch.Text = watch.ElapsedMilliseconds.ToString();

```

2. Zaimplementować funkcje StartTasks i FinishTasks wg poniższego schematu dla wszystkich wywołań webserwisów:

```

private void StartTasks()
{
    ServiceReference2.DelayedStockQuoteSoapClient clientQuote = new
        ServiceReference2.DelayedStockQuoteSoapClient("DelayedStockQuoteSoap");//1
    taskQuote = client.GetQuickQuoteAsync(textBox2.Text, "0");//2

    ServiceSomething.SomethingSoapClient clientSomething = new
        ServiceSomething.SomethingSoapClient("SomethingSoap");//1
    taskSomething = client.ConvertSomethingAsync(Convert.ToDouble(textBox3.Text));//2
}

```

}

gdzie:

//1 – to kopia pierwszej linijki funkcji podpiętej pod przycisk obsługujący WS – zmieniona tylko nazwa zmiennej

client

//2 - zmodyfikowana kopia drugiej linijki, wynik jest przypisywany do odpowiedniego zadania, wywołujemy wersję Async metody komunikacyjnej.

```

private void FinishTasks()
{
    labelQuote.Text = taskQuote.Result.ToString();
    labelSomething.Text = taskSomething.Result.ToString();
}

```

17. Przetestować czas wywołań synchronicznych i asynchronicznych.