Gliwice, Kwiecień 2004

Tworzenie programów dla platformy .NET

Atrybuty

Komponenty i kontrolki

Marek Mittmann

Plan wykładu

- Metadane
- Atrybuty
- Komponenty i kontrolki
- Tworzenie kontrolek użytkownika

Metadane

- Metadane – informacje dołączane do podzespołu razem z kodem (opis interfejsów komponentów, opcje, informacje o klasach itp.)
- Umieszczanie dodatkowych danych przez użytkownika – atrybuty
- Wydobywanie informacji z podzespołu - odzwierciedlanie

Atrybuty

- Atrybuty standardowe
- Tworzenie własnych atrybutów
- Przykład atrybutu:

```
[assembly: AssemblyVersion("1.0.*")]
```

Element, z którym ma być powiązany atrybut

Nazwa atrybutu

Parametry

Atrybuty c.d.

- Elementy, dla których można stosować atrybuty:
  - Złożenie (assembly)
  - Moduł (module)
  - Klasa lub struktura (type)
  - Metoda (method)
  - Właściwość (property)
  - Zdarzenie (event)
  - Pole (field)
  - Parametr (param)
  - Wartość zwracana (return)

Tworzenie atrybutów

```
[AttributeUsage(AttributeTargets.Class,
                AllowMultiple = true)]
public class MyAttribute: System.Attribute
{
  public MyAttribute(string text) {
      this.text = text;
  }

  public string Text {
    get { return text; }
  }

  string text;
}
```

Odzwierciedlanie

```
// Użycie atrybutu MyAttribute
[MyAttribute(„Class Complex”)]
class Complex { }
// Pobieranie atrybutów
class Test {
  public static void Main() {
    Type type = typeof(Complex);
    foreach (MyAttribute atr in
      type.GetCustomAttributes(
        typeof(MyAttribute, false)))
    {
      Console.WriteLine(„Text: {0}”, atr.Text);
    }
} }
```


Komponenty i kontrolki

- Komponenty i kontrolki – wydzielone, niezależne składniki aplikacji, przeznaczone do wielokrotnego wykorzystania w różnych programach
- Kontrolki różnią się od komponentów możliwością działania jako elementy GUI (wyświetlanie, reagowanie na akcje użytkownika)

Rodzaje komponentów

- Komponenty zarządzane (.NET)
  - Wymagają do działania środowiska uruchomieniowego CLR
  - Używanie w aplikacjach niezarządzanych wymaga specjalnego „opakowania”
- Komponenty niezarządzane (COM)
  - Muszą zostać „opakowane”, aby działać w aplikacjach dla platformy .NET

Cechy komponentów .NET

- Samoopisujące – wszystkie informacje potrzebne do użycia komponentu zawarte są wewnątrz podzespołu w postaci metadanych
- Można używać ich w aplikacjach pisanych w dowolnym języku zgodnym z .NET
- Po „opakowaniu” specjalnym kodem mogą być używane w aplikacjach COM

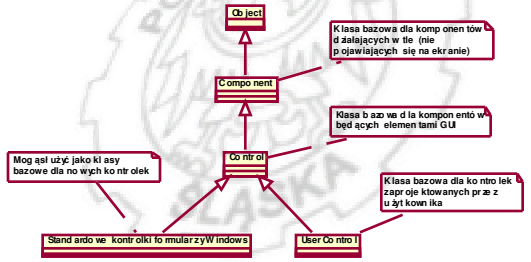
Cechy komponentów .NET c.d.

- Można dziedziczyć klasy komponentów pomiędzy różnymi językami platformy .NET
- System może jednocześnie używać kilku wersji tego samego komponentu – znika konieczność dbania o zachowanie kompatybilności z poprzednimi wersjami

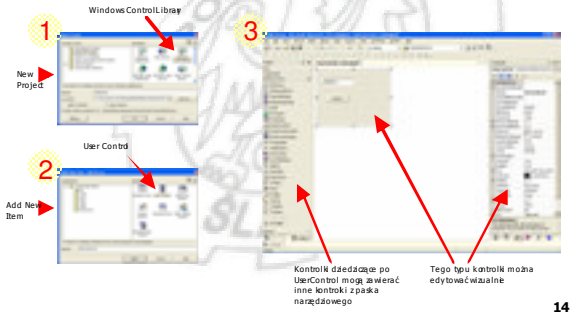
Ograniczenia komponentów

- Wymagają do działania platformy .NET (nawet jeżeli są używane w kodzie niezarządzanym)
- Niektóre składniki komponentów mogą nie być dostępne dla wszystkich języków (np. przeciążane operatory są obsługiwane tylko przez C# i C++)

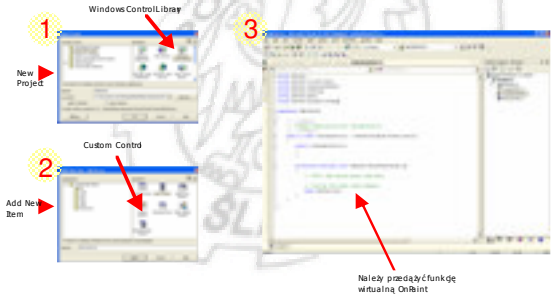
Hierarchia komponentów .NET



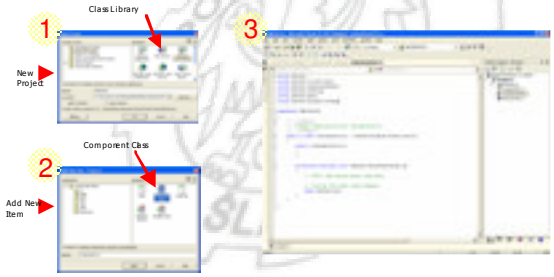
Tworzenie „User Control”



Tworzenie „Custom Control”



Tworzenie komponentów



Dodawanie właściwości i zdarzeń

```
private string _caption;
// Właściwość widoczna w edytorze właściwości
[Category(„Appearance”)]
public string Caption {
  get { return _caption; }
  set { _caption = value; }
}

protected void OnClick(EventArgs e) {
  base.OnClick(e);
  AfterClick(this, e); // wywołanie zdarzenia
}
// Zdarzenie widoczne w edytorze właściwości
public event EventHandler AfterClick;
```


Style wizualne w Windows XP

- Co jest potrzebne?
  - Biblioteka *Comctl32.dll* w wersji 6.0 (dostarczana z Windows XP)
  - Plik manifestu – plik XML dołączony do aplikacji jako zasób lub umieszczony w katalogu z plikiem wykonywalnym
  - W kontrolkach, które mają właściwość *FlatStyle*, należy ustawić wartość *FlatStyle.System*

Plik manifestu

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
<assemblyIdentity
    version="1.0.0.0"
    processorArchitecture="X86"
    name="Microsoft.Winweb.<Executable Name>"
    type="win32"
/>
<description>.NET control deployment tool</description>
<dependency>
    <dependentAssembly>
        <assemblyIdentity
            type="win32"
            name="Microsoft.Windows.Common-Controls"
            version="6.0.0.0"
            processorArchitecture="X86"
            publicKeyToken="6595b64144ccf1df"
            language="*"
        />
    </dependentAssembly>
</dependency>
</assembly>
```

Nazwa pliku wykonywanego aplikacji (bez rozszerzenia)

19

Dodawanie manifestu do aplikacji

- Jako plik zewnętrzny
  - Zmienić nazwę pliku manifestu na *NazwaAplikacji.exe.manifest*
  - Umieścić w katalogu aplikacj, razem z plikiem wykonywalnym
- Jako zasób zawarty w pliku wykonywalnym
  - Dołączyć do pliku wykonywalnego plik manifestu jako zasób typu *RT_MANIFEST*
  - Zmienić wartość identyfikatora zasobu na 1

20

Dziękuję za uwagę