www.asix.com.pl

Asix.Evo - Obiekty zewnętrzne

Dok. Nr PLP7E007
Wersja: 2014-03-18

ASKOM

ASKOM Sp. z o. o., ul. Józefa Sowińskiego 13, 44-121 Gliwice,
tel. +48 32 3018100, fax +48 32 3018101,

http://www.askom.com.pl, e-mail: biuro@askom.com.pl

Spis treści

1 Dostarczanie własnych obiektów do aplikacji Asix.Evo

Zewnętrzne obiekty dla aplikacji Asix Evo należy dostarczać w postaci skompilowanych bibliotek .NET jako pliki (*.dll*). Biblioteki te oraz wszystkie inne powiązane z nimi biblioteki i zasoby należy umieszczać w katalogu **Addons**, który jest podkatalogiem aplikacji Asixa Evo. Nowe lub zmienione obiekty i biblioteki będą widoczne po zrestartowaniu programu AsixEvo i aplikacji. Nazwa biblioteki zawierającej obiekty dla Asixa Evo musi rozpoczynać się przedrostkiem *„EvoObject.”* np.: EvoObject.TabeleSQL.dll.

UWAGA:
Ponieważ obiekty zewnętrzne dostarczane są w postaci bibliotek .NET, więc mogą zawierać dowolny kod, w szczególności kod niebezpieczny lub niestabilny. Kontrolki należy pisać unikając mocno obciążających operacji, zawieszających działanie wątku głównego (synchroniczne wywołania oczekujące), powodujących wycieki pamięci i inne niepożądane efekty. Za błędne lub niestabilne działanie dodatków do Asixa Evo dostarczonych przez użytkowników firma ASKOM nie ponosi odpowiedzialności.

2 Tworzenie kontrolek obiektów

2.1 Środowisko programistyczne

Do tworzenia własnych kontrolek obiektów zalecane jest używanie narzędzi programistycznych Microsoft: Visual Studio (2008 lub nowszy) lub darmowy Visual C# Express, zgodnych z platformą .NET Framework 3.5.

2.2 Klasa bazowa kontrolki

```
/// <summary>
/// Base class for Asix Evo diagram external objects
/// / Klasa bazowa dla zewnętrznych obiektów diagramowych
/// </summary>

public partial class EvoObjectBase : UserControl, AsixEvoAddon

{

  /// <summary>
   /// Override to return unique EvoObject ID. The proper ID is required.
   /// / Należy przeciążyć aby zwracać identyfukator obiektu. Wymagane!
   /// </summary>
   public virtual string EvoObjectID { get { return null; } }

   /// <summary>
   /// Override to return evo object description
   /// / Można prtzeciążyć aby zwracać nazwę (krótki opips) obiektu
   /// </summary>
   public virtual string EvoObjectDescription { get { return null; } }

   /// <summary>
   /// Override to initialize or reset control (e.g. register variable events)
   /// / Można przeciążyć aby dokonać inicjalizacji lub reinicjalizacji kontrolki
   /// (n.p. rejestrowanie zdarzeń zmiennych)
   /// </summary>
   public virtual void Reset() { }

   /// <summary>
   /// Override to cleanup when control is inactive/hidden etc. (e.g. unregister variable e
vents)
   /// / Można przeciążyć aby posprzątać (n.p. wyrejestrowanie zdarzeń zemiennych)
   /// </summary>
   public virtual void Disconnect() { }

   /// <summary>
   /// Override and return true to enable transparency for object in Control mode
```

```
        /// / Można przeciążyć i zwrócić true aby włączyć obsługę przeźroczystości dla
        /// obiektu kontrolkowego
        /// </summary>
        public virtual bool EvoObjectUsesTransparency { get { return false; } }

        /// <summary>
        /// Override and return true to determine if object is in OwnerDraw mode (vector, not co
ntrol)
        /// / Należy przeciażyć i zwrócić true aby określić że obiekt
        /// jest w trybie OwnerDraw (wektorowy, nie kontrolkowy)
        /// </summary>
        public virtual bool OwnerDraw { get { return false; } }

        /// <summary>
        /// Overide to draw object if object is in OwnerDraw mode
        /// / Należy przeciażyć aby malować obiekt gdy jest on w trybie OwnerDraw
        ///  (wektorowy, nie kontrolkowy)
        /// </summary>
        /// <param name="aGraphics">GDI + interface / Interfejs GDI+</param>
        /// <param name="aArea">Object bounds / Połozenie i rozmiary obiektu</param>
        public virtual void Paint(Graphics aGraphics, Rectangle aArea) { }

        /// <summary>
        /// Overide to handle mouse click events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia klikniecia myszy
        /// </summary>
        /// <param name="e">Mouse events data / Szczegóły zdarzenia myszy</param>
        /// <param name="aArea">Current object area / Bieżący obszar obiektu</param>
        public virtual void PerformMouseClick(MouseEventArgs e, Rectangle aArea) { }

        /// <summary>
        /// Overide to handle object focused events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia aktywacji obiektu
        /// </summary>
        public virtual void PerformFocus() { }

        /// <summary>
        /// Overide to handle object focus lost events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia dezaktywacji obiektu
        /// </summary>
        public virtual void PerformFocusLost() { }

        /// <summary>
        /// Overide to handle mouse double click events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia podwójnego klikniecia myszy
        /// </summary>
        /// <param name="e">Mouse events data / Szczegóły zdarzenia myszy</param>
        /// <param name="aArea">Current object area / Bieżący obszar obiektu</param>
        public virtual void PerformMouseDoubleClick(MouseEventArgs e, Rectangle aArea) { }

        /// <summary>
        /// Overide to handle mouse down events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia wciśniecia klawisza myszy
        /// </summary>
        /// <param name="e">Mouse events data / Szczegóły zdarzenia myszy</param>
        /// <param name="aArea">Current object area / Bieżący obszar obiektu</param>
        public virtual void PerformMouseDown(MouseEventArgs e, Rectangle aArea) { }

        /// <summary>
        /// Overide to handle mouse up events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia puszczenia myszy
        /// </summary>
        /// <param name="e">Mouse events data / Szczegóły zdarzenia myszy</param>
        /// <param name="aArea">Current object area / Bieżący obszar obiektu</param>
        public virtual void PerformMouseUp(MouseEventArgs e, Rectangle aArea) { }


        /// <summary>
        /// Overide to handle mouse move events
```

```
        /// / Można przeciążyc aby obsłuyżyć zdarzenia ruchu myszy
        /// </summary>
        /// <param name="aPoint">Cursor location on object/ Położenie kursora nad obiektam</para
m>
        /// <param name="aArea">Current object area / Bieżący obszar obiektu</param>
        public virtual void PerformMouseMove(System.Drawing.Point aPoint, Rectangle aArea) { }

        /// <summary>
        /// Overide to handle mouse enter events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia najechania kursora myszy na obiekt
        /// </summary>
        public virtual void PerformMouseEnter() { }

        /// <summary>
        /// Overide to handle mouse leave events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia opuszczenia przez kursor obszaru obiektu
        /// </summary>
        public virtual void PerformMouseLeave() { }

        /// <summary>
        /// Overide to handle mouse hold events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia przytrzymania myszy nad obiektem
        /// </summary>
        /// <param name="e">Mouse events data / Szczegóły zdarzenia myszy</param>
        /// <param name="aArea">Current object area / Bieżący obszar obiektu</param>
        public virtual void PerformMouseHold(MouseEventArgs e, Rectangle aArea) { }

        /// <summary>
        /// Overide to handle key down events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia wciśnięcia klawisza
        /// </summary>
        /// <param name="e">Key events data / Szczegóły zdarzenia klawisza</param>
        public virtual void PerformKeyDown(KeyEventArgs e) { }

        /// <summary>
        /// Overide to handle key up events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia puszczenia klawisza
        /// </summary>
        /// <param name="e">Key events data / Szczegóły zdarzenia klawisza</param>
        public virtual void PerformKeyUp(KeyEventArgs e) { }

        /// <summary>
        /// Overide to handle key pressed events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia wciśniecia klawisza znaku
        /// </summary>
        /// <param name="e">Key events data / Szczegóły zdarzenia klawisza</param>
        public virtual void PerformKeyPressed(KeyPressEventArgs e) { }




    }
```

2.3 Tworzenie klasy kontrolki

Klasy kontrolek obiektów dla Asixa Evo najlepiej umieszczać w bibliotece utworzonej jako projekt typu Class Library. Klasa taka musi być publiczną klasą pochodną klasy *Askom.AddonManager.EvoObjectBase,* której definicja znajduje się w bibliotece *AsixEvoAddonManager.dll*. Referencja do biblioteki *AsixEvoAddonManager.dll* powinna być dodana do projektu. Plik *AsixEvoAddonManager.dll* znajduje się w katalogu instalacyjnym programu Asix Evo.

Najbardziej zalecaną metodą tworzenia kontrolki jest wykorzystanie w projekcie kreatora *Add NewItem->Windows Forms->Inherited User Control* , wskazanie w tym kreatorze (Browse…) pliku *AsixEvoAddonManager.dll,* a następnie wybranie z listy komponentu *EvoObjectBase*.

W klasie kontrolki (pochodnej klasy *Askom.AddonManager.EvoObjectBase)* należy obowiązkowo przeciążyć właściwość *EvoObjectID* , tak aby zwracała unikalny tekstowy identyfikator obiektu. Obiekty nie zwracające tego identyfikatora, zwracające pusty lub już wcześniej używany identyfikator, nie będą widoczne w aplikacji.

Opcjonalnie można dodatkowo przeciążyć właściwość *EvoObjectDescription*, tak aby zwracała krótką nazwę opisową obiektu, która będzie widoczna w przyborniku oraz jako nazwa typu obiektów na diagramach. Jeżeli nie zdefiniujemy tej nazwy, to będzie używany identyfikator z *EvoObjectID*.

Obiekt może pracować w trybie kontrolkowym lub wektorowym. W trybie kontrolkowym obiekt stanowi kontrolkę .NET, na której można umieszczać inne standardowe lub własne kontrolki .NET. W trybie wektorowym obiekt nie jest kontrolką, ale jest malowany za pomocą interfejsu GDI+ dla .NET, a odpowiednie zdarzenia obsługuje przeciążając właściwe metody wirtualne. Domyślnie obiekt pracuje w trybie kontrolkowym. Aby obiekt pracował w trybie wektorowym, to należy przeciążyć właściwość *OwnerDraw*, tak aby zwracała wartość *true*.

Jeżeli chcemy wykonywać w obiekcie jakieś czynności inicjalizacyjne czy czyszczące w chwilach, gdy obiekt jest podłączany do diagramu (np. podczas otwierania diagramu), to należy przeciążyć metodę *Reset()*.

Czyszczenie i zwalnianie zasobów przy zamknięciu i zwalnianiu kontrolki można wykonać podłączając się do zdarzenia *Disposed,* dostępnego w klasie bazowej, lub przeciążając metodę *Dispose*.

2.4 Definiowanie własnych właściwości obiektu

Jeżeli chcemy, aby pewne właściwości obiektu były dostępne dla aplikacji Asix Evo, należy je zadeklarować w klasie kontrolki jako zwykłe publiczne właściwości z obowiązkowymi akcesorami typu *set* oraz opcjonalnymi akcesorami typu *get*. Akcesory *get* mogą służyć do pozyskiwania wartości domyślnych dla właściwości obiektów Evo.

Każda właściwość komponentu, aby była widoczna w Aplikacji Evo, musi zostać oznakowana za pomocą atrybutu *Askom.AddonManager.EvoObjectProperty*. W atrybucie tym możemy opcjonalnie zdefiniować krótki opis tekstowy właściwości za pomocą parametru *Description*. Jeżeli tego nie zrobimy, opis ten będzie taki sam jak nazwa właściwości. Domyślnie właściwość będzie ulokowana w grupie właściwości podstawowych obiektu aplikacji Evo. Jeżeli chcemy, aby właściwość znajdowała się w grupie stanowej, to należy użyć w atrybucie opcjonalnego parametru *PropertyGroup* i nadać mu wartość *Askom.AddonManager.EvoObjectPropertyGroup.StateGroup*. Istnieje także możliwość umieszczenia właściwości we własnych grupach. W tym celu należy użyć w atrybucie opcjonalnego parametru *PropertyGroup* i nadać mu wartość *Askom.AddonManager.EvoObjectPropertyGroup.CustomGroup* oraz za pomocą parametru *CustomGroupName* określić nazwę własnej grupy. Jeżeli kilka właściwości będzie używało tej samej nazwy grupy, to będą one wspólnie znajdować się tej grupie.

Za pomocą parametru atrybutu *Askom.AddonManager.EditorType* możemy wskazać, jaki edytor wartości ma być używany w edytorze właściwości. Jeżeli nie użyjemy tego parametru, lub ustawimy go na wartość *Default,* to prosty edytor wartości zostanie określony na podstawie typu właściwości.

Edytor wartości	Zalecany typ
Default	string
Text	int
Logical	int
Color	int
Cursor	Int
Integer	Int
Double	bool
Variable	bool
FontName	int
FontStyle	int
PictureName	string
Shortcut	string
Attribute	string

2.5 Dostęp do standardowych właściwości obiektu

Dostęp do standardowych właściwości obiektu (takich jak np.: *Zmienna główna*, *Aktywność*, itp.) realizujemy poprzez zadeklarowanie w klasie kontrolki publicznych właściwości o dowolnej nazwie z obowiązkowymi akcesorami typu *set* oraz oznaczenie ich za pomocą atrybutu *Askom.AddonManager.EvoObjectStandardProperty*. Obowiązkowym parametrem atrybutu *EvoObjectStandardProperty* jest nazwa właściwości standardowej (typ wyliczeniowy *Askom.AddonManager.EvoObjectStandardPropertyName*). Dla każdej z dostępnych właściwości standardowych powinien być używany odpowiedni typ właściwości:

Nazwa właściwości	Wymagany typ	Opis
Name	string	Nazwa obiektu
X	int	Współrzędna X (wartość względna dla diagramu od 0 do 1 000000)
Y	int	Współrzędna X (wartość względna dla diagramu od 0 do 1 000000)
Width	int	Szerokość (wartość względna dla diagramu od 0 do 1 000000)
Height	Int	Wysokość X (wartość względna dla diagramu od 0 do 1 000000)
Z	Int	Warstwa
Active	bool	Informuje czy obiekt jest aktywny
Visible	bool	Informuje czy obiekt jest widoczny
MinVisibleWidth	int	Minimalna szerokość widzialności (w pikselach)
MinVisibleHeight	int	Minimalna wysokość widzialności (w pikselach)
MainVar	string	Nazwa zmiennej głównej obiektu
ControlVar	string	Nazwa zmiennej sterowanej obiektu
Hint	string	Tekst dymka
CursorType	string	Nazwa typu kursora

Z punktu widzenia kontrolki zewnętrznej, właściwości standardowe są tylko do odczytu. Ustawianie ich wartości przez kontrolkę nie ma wpływu na działanie aplikacji.

2.6 Definiowanie własnych zdarzeń obiektu

Aby w aplikacji Asix Evo były widoczne określone zdarzenia obiektu, to należy je w klasie kontrolki zdefiniować jako publiczne zdarzenia (*event*) typu *Askom.AddonManager.EvoObjectEventHandler* oraz oznakować atrybutem *Askom.AddonManager.EvoObjectEvent*. W atrybucie tym, podobnie jak dla własnych właściwości, można opcjonalnie zdefiniować krótki opis za pomocą parametru *Description*.

2.7 Używanie przeźroczystości w obiektach kontrolkowych

Aby obiekt zewnętrzny w trybie kontrolkowym obsługiwał przeźroczystość (brak tła), to należy w klasie kontrolki (pochodnej klasy *Askom.AddonManager.EvoObjectBase*) przeciążyć właściwość *EvoObjectUsesTransprancy,* tak aby zawsze zwracała wartość *true*. Dodatkowo aby kontrolka sama nie malowała tła, konieczne jest przeciążenie właściwości systemowej *CreateParams* w sposób przedstawiony poniżej:

```
protected override CreateParams CreateParams
```

```
{
    get
    {
        const int WS_EX_TRANSPARENT = 0x20;
        CreateParams cp = base.CreateParams;
        cp.ExStyle |= WS_EX_TRANSPARENT;
        return cp;
    }
}
```

2.8 Pobieranie stanu i wartości zmiennych

Do pobrania wartości bieżącej zmiennej należy użyć funkcji:

```
VariableState GetVariableState(string aVarName);
```

aVarName - nazwa zmiennej

Funkcja te jest dostępna bezpośrednio w klasie bazowej kontrolki oraz w klasie *AsixEvo.Application*. Jeżeli zmienna istnieje, to funkcja zwraca obiekt deskryptora stanu zmiennej. Deskryptor zawiera informacje o aktualnej wartości (surowej, zwykłej i sformatowanej), o czasie zmiennej oraz status OPC. Gdy zmienna o danej nazwie nie istnieje, to funkcja zwróci *null*. Funkcja nie gwarantuje, że pobrany stan będzie odpowiadał rzeczywistej wartości aktualnej - jeżeli zmienna nie została zarejestrowana do nasłuchiwania (odświeżania).

2.9 Pobieranie wartości atrybutów zmiennych

Do pobrania wartości atrybutów zmiennej należy użyć funkcji:

```
object GetVariableAttribute(string aVarName, string aAttributeName);
```

aVarName - nazwa zmiennej
aAttributeName - nazwa atrybutu

Funkcja te jest dostępna bezpośrednio w klasie bazowej kontrolki oraz w klasie *AsixEvo.Application*. Jeżeli zmienna oraz atrybut istnieją, to funkcja zwraca wartość atrybutu zmiennej, w przeciwnym przypadku funkcja zwróci *null*.

2.10 Sterowanie zmiennych

Do sterowania zmiennych należy wykorzystać funkcję *AsixEvo.Application.SetVariableValue(&ldots;)*. Funkcja jest dostępna w następujących wersjach:

```
bool SetVariableValue(string aVarName, object aValue, bool aVerifyPrivileges);

bool SetVariableValue(string aVarName, object aValue, uint aStatus, bool aVerifyPrivileges);

bool SetVariableValue(string aVarName, object aValue, bool aVerifyPrivileges, out string
aErrorMsg);

bool SetVariableValue(string aVarName, object aValue, uint aStatus, bool aVerifyPrivileges,
out string aErrorMsg);
```

aVarName	- nazwa zmiennej
aValue	- nowa wartość
aStatus	- status OPC
aVarifyPrivileges	- określa, czy sprawdzać uprawnienia do sterowania. Jeżeli sterujemy ze sprawdzaniem uprawnień, to w przypadku braku uprawnień zostanie wyświetlone okienko umożliwiające tymczasowe (na czas sterowania) zalogowanie na użytkownika posiadającego uprawnienia.

Wersja funkcji bez argumentu *aStatus* domyślnie ustawia status OPC jako dobry.

Wersja funkcji z argumentem *aErrorMsg* zwraca za pomocą tego argumentu informacje o błędzie jeżeli sterowanie się nie powiedzie zamiast wyświetlać okienko dialogowe z opisem błędu.

Funkcje te są dostępne również bezpośrednio w klasie bazowej kontrolki.

2.11 Rejestrowanie i wyrejestrowywanie zdarzeń zmiany stanu zmiennych

Do zarejestrowania zdarzeń zmiany stanu zmiennej należy użyć funkcji:

```
bool RegisterVariable(string aVarName);
```

aVarName - nazwa zmiennej

Funkcja jest dostępna w klasie bazowej kontrolki. Jeżeli zmienna istnieje i udało się podłączyć do rejestrowania jej zdarzeń, to funkcja zwróci *true*.

Aby w kontrolce przechwytywać zdarzenia od zmiany stanu zarejestrowanych zmiennych, należy w klasie zdefiniować funkcję o dowolnej nazwie, ale zgodną z poniższym schematem:

```
void VarRefreshTest_VariableStateChangedEvent(string aVariableName, VariableState aVariableSta
te)
{

}
```

A następnie w konstruktorze podłączyć do niej zdarzenia korzystając ze znajdującego się w klasie bazowej kontrolki zdarzenia *VariableStateChangedEvent* :

```
VariableStateChangedEvent += VarRefreshTest_VariableStateChangedEvent;
```

Tak zdefiniowana funkcja będzie przechwytywała zdarzenia zmiany wartości, stanu lub czasu wszystkich zarejestrowanych zmiennych. Pierwszy argument funkcji *aVariableName* informuje o tym, której zmiennej dotyczy zmiana. Drugi argument zawiera deskryptor stanu zmiennej.

Aby wyrejestrować przechwytywanie zdarzeń dla wskazanej zmiennej, należy wykorzystać funkcję:

```
void UnregisterVariable(string aVarName);
```

aVarName - nazwa zmiennej

Funkcja jest dostępna w klasie bazowej kontrolki.

Aby wyrejestrować przechwytywanie zdarzeń dla wszystkich zarejestrowanych zmiennych, należy użyć funkcji:

```
void UnregisterAllRegisteredVariables();
```

Funkcja jest dostępna w klasie bazowej kontrolki.

2.12 Dostęp do wartości archiwalnych zmiennych

Do wysłania zapytania o wartości archiwalne zmiennych służy funkcja:

```
IEvoAddonArcQueryResult OrderVariableArchData(string aVarName, string aAggregate, DateTime aStartTime,
DateTime aEndTime, TimeSpan aAggregateInterval);
```

aVarName - nazwa zmiennej
aAggregate - nazwa funkcji agregującej („none”, „average”, „start” itp.)
aStartTime - czas początku zakresu danych archiwalnych
aEndTime - czas końca zakresu danych archiwalnych
aAggregateInterval - interwał agregacji

Powyższa metoda zwraca interfejs deskryptora wyniku *IEvoAddonArcQueryResult*, który umożliwia sprawdzenie statusu realizacji zapytania, pobranie jego wyników oraz rejestrację funkcji zwrotnej, która będzie wywołana po zmianie statusu, np. po wykonaniu zapytania lub jego błędzie). Po zakończeniu należy wywołać funkcję *Dispose()*.

```
public interface IEvoAddonArcQueryResult
{
	string VariableName { get; }
	EvoAddonArcResultStatus ResultStatus { get; }
	EvoAddonArcVariableState[] Values { get; }
	event EvoAddonArchQueryResultEvent ArchQueryResultEvent;
	void Dispose();
}
```

VariableName - nazwa zmiennej, na której wykonano zapytanie

ResultStatus - status realizacji zapytania (*Processing*, *Ready*, *Error*)

Values - tablica wartości wynikowych

ArchQueryResultEvent - zdarzenie zmiany statusu

2.13 Dostęp do tekstów wielojęzycznych aplikacji

Do pobierania napisów aplikacji w bieżącym języku służy funkcja:

```
string Askom.AddonManager.AsixEvo.Application.GetApplicationText(string aTextId);
```

Kod numeryczny bieżącego języka aplikacji można pobrać za pomocą właściwości:

```
int Askom.AddonManager.AsixEvo.Application.CurrentApplicationLanguageCode
```

Do przechwytywania zdarzeń zmiany języka aplikacji należy wykorzystać:

```
event EventHandler Askom.AddonManager.AsixEvo.Application.CurrentApplicationLanguageChanged;
```

2.14 Dostęp do obrazków aplikacji

Do pobierania obrazków z puli obrazków aplikacji należy używać funkcji:

```
Image EventHandler Askom.AddonManager.AsixEvo.Application.GetImage(string aImageName)
```

2.15 Obsługa zdarzeń w trybie wektorowym

Aby obiekt pracował w trybie wektorowym, to należy przeciążyć właściwość *OwnerDraw*, tak aby zwracała wartość *true*.

Podstawową metodą, którą należy przeciążyć i obsłużyć, gdy obiekt pracuje w trybie wektorowym, jest:

```
public virtual void Paint(Graphics aGraphics, Rectangle aArea) { }
```

W funkcji tej można malować zawartość obiektu za pomocą obiektu klasy Graphics, stanowiący w .NET odpowiednik GDI+. Należy zawsze uwzględniać aktualne rozmiary obiektu dostępne w parametrze *aArea,* tak aby obiekt wyglądał prawidłowo bez względu na skalę i położenie.

Pozostałe metody wirtualne dostępne w klasie bazowej (wypisane i opisane na str. 2) można przeciążać i wykorzystywać do obsługi zdarzeń interakcyjnych, jak np.: kliknięć myszą, wciśnięć klawiszy itp.

2.16 Przykład 1 – Definiowanie właściwości i zdarzeń

Poniżej zamieszczono przykładową definicję klasy własnego obiektu dla Asix Evo. Obiekt składa się z dwóch przycisków w układzie pionowym, dwóch pól edycyjnych tekstu oraz pola wyboru oraz pól etykiet. Pierwszy przycisk o stałej wysokości umieszczono na górze. Jego naciśnięcie powoduje wykonanie akcji określonej przez projektanta diagramu. Drugi przycisk pokrywa środkową część obszaru i służy do wykonywania sterowań zmiennych. Pierwsze pole edycyjne służy do wprowadzania nazwy zmiennej, domyślnie pobiera wartość z właściwości standardowej „Zmienna sterowana" obiektu. Drugi przycisk pozwala wprowadzać wartość dla zmiennych. Pole wyboru określa, czy sterowanie ma następować ze sprawdzaniem uprawnień. Dla kontrolki zdefiniowano własne właściwości i zdarzenia widoczne w aplikacji Asix Evo oraz odwołania do właściwości standardowych. Dodatkowo kontrolka pobiera z aplikacji określone napisy aplikacji (jeżeli istnieją w aplikacji) oraz reaguje na zmiany bieżącego języka.

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Drawing;

using Askom.AddonManager;
```

```
namespace TestowaKontrolkaEvo
{
   /// <summary>
   /// Przykładowa klasa kontrolki obiektu dla Asix Evo.
   /// </summary>
   public class SampleEvoControl : Askom.AddonManager.EvoObjectBase
   {
      private System.Windows.Forms.Button button1;
      private System.Windows.Forms.Button button2;
      private System.Windows.Forms.TextBox textBox1;
      private System.Windows.Forms.TextBox textBox2;
      private System.Windows.Forms.CheckBox checkBox1;
      private System.Windows.Forms.Label label1;
      private System.Windows.Forms.Label label2;

      public SampleEvoControl()
      {
         // Inicjalizacja...

         SuspendLayout();

         // Przycisk 1
         button1 = new System.Windows.Forms.Button();
         button1.Dock = System.Windows.Forms.DockStyle.Top;
         button1.Name = "button1";
         button1.Text = "Rób coś";
         button1.UseVisualStyleBackColor = true;
         button1.Click += button1_Click;
         Controls.Add(button1);

         // Przycisk 2
         button2 = new System.Windows.Forms.Button();
         button2.Dock = System.Windows.Forms.DockStyle.Fill;
```

```
button2.Name = "button2";
button2.Text = "Steruj";
button2.Click += button2_Click;
Controls.Add(button2);

// Etykieta 1
label1 = new System.Windows.Forms.Label();
label1.Dock = System.Windows.Forms.DockStyle.Bottom;
label1.Name = "label1";
label1.Text = "Zmienna:";
Controls.Add(label1);

// Pole tekstowe 1
textBox1 = new System.Windows.Forms.TextBox();
textBox1.Dock = System.Windows.Forms.DockStyle.Bottom;
textBox1.Name = "textbox1";
Controls.Add(textBox1);

// Etykieta 2
label2 = new System.Windows.Forms.Label();
label2.Dock = System.Windows.Forms.DockStyle.Bottom;
label2.Name = "label2";
label2.Text = "Wartość:";
Controls.Add(label2);

// Pole tekstowe 2
textBox2 = new System.Windows.Forms.TextBox();
textBox2.Dock = System.Windows.Forms.DockStyle.Bottom;
textBox2.Name = "textbox2";
Controls.Add(textBox2);

// Pole wyboru
checkBox1 = new System.Windows.Forms.CheckBox();
checkBox1.Dock = System.Windows.Forms.DockStyle.Bottom;
```

```
            checkBox1.Name = "checkBox1";
            checkBox1.Text = "Uprawnienia";
            checkBox1.Checked = true;
            Controls.Add(checkBox1);

            Name = "SampleEvoControl";

            ResumeLayout(false);

            if (AsixEvo.Application != null)
            {
                // Aktualizujemy opipsy na podstawie napisów aplikacji
                UpdateTexts();

                AsixEvo.Application.CurrentApplicationLanguageChanged += Application_CurrentApplic
ationLanguageChanged;
            }
        }

        public override string EvoObjectID
        {
            get
            {
                return "SampleEvoControl";  // Unikalny identyfikator obiektu
            }
        }

        public override string EvoObjectDescription
        {
            get
            {
                return "Kontrolka z przyciskami";  // Krótki opis obiektu
            }
        }
```

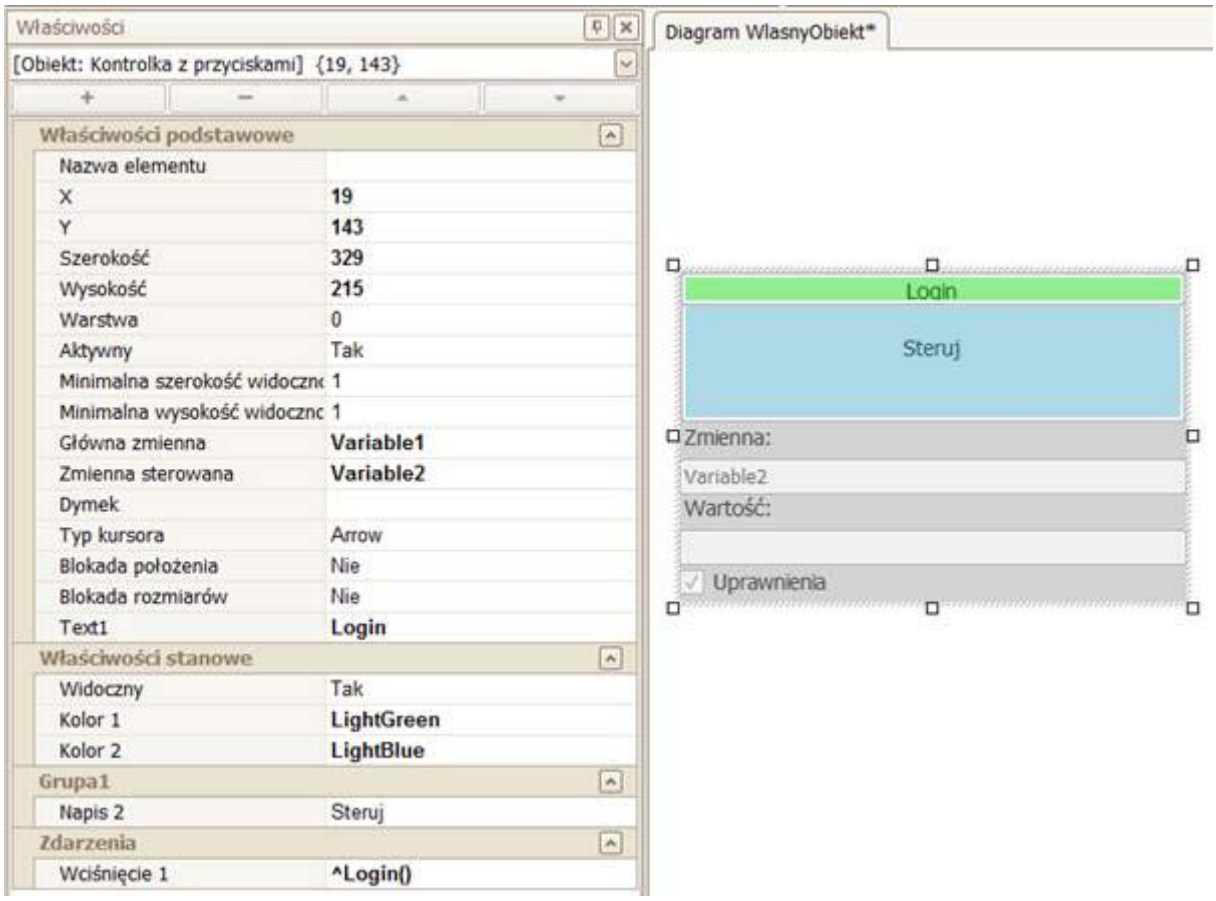
```
public override void Reset()
{
   // Reinicjalizacja obiektu ...

   base.Reset();
}

/// <summary>
/// Przechwytywanie zdarzenia zmiany języka
/// </summary>
void Application_CurrentApplicationLanguageChanged(object sender, EventArgs e)
{
   UpdateTexts();
}

/// <summary>
/// Aktualizacja opipsów elemntów na podstawie napisów aplikacji
/// </summary>
void UpdateTexts()
{
   if (AsixEvo.Application != null)
   {
      string txt = AsixEvo.Application.GetApplicationText("action");
      if (!string.IsNullOrEmpty(txt))
         button1.Text = txt;

      txt = AsixEvo.Application.GetApplicationText("control");
      if (!string.IsNullOrEmpty(txt))
         button2.Text = txt;

      txt = AsixEvo.Application.GetApplicationText("var");
      if (!string.IsNullOrEmpty(txt))
         label1.Text = txt;
```

```
            txt = AsixEvo.Application.GetApplicationText("val");
            if (!string.IsNullOrEmpty(txt))
               label2.Text = txt;

            txt = AsixEvo.Application.GetApplicationText("priv");
            if (!string.IsNullOrEmpty(txt))
               checkBox1.Text = txt;
         }
      }

      /// <summary>
      /// Właściwość napisu na pierwszym przycisku. Grupa podstawowa. Widoczna w aplikacji jak
o 'Text1'.
      /// </summary>
      [Askom.AddonManager.EvoObjectProperty]
      public string Text1
      {
         get { return button1.Text; }
         set { button1.Text = value; }
      }

      /// <summary>
      /// Właściwość napisu na drugim przycisku. Grupa własna o nazwie "Grupa1". Widoczna w ap
likacji jako 'Napis 2'.
      /// </summary>
      [Askom.AddonManager.EvoObjectProperty(Description = "Napis 2",
         PropertyGroup = Askom.AddonManager.EvoObjectPropertyGroup.CustomGroup, CustomGroupNam
e="Grupa1")]
      public string Text2
      {
         get { return button2.Text; }
         set { button2.Text = value; }
      }

      /// <summary>
```

```
      /// Właściwość koloru tła pierwszego przycisku. Grupa stanowa. Widoczna w aplikacji jako
 'Kolor 1'.

      /// </summary>

      [Askom.AddonManager.EvoObjectProperty(Description = "Kolor 1", PropertyGroup = Askom.Add
onManager.EvoObjectPropertyGroup.StateGroup)]

      public Color Color1

      {

         get { return button1.BackColor; }

         set { button1.BackColor = value; }

      }



      /// <summary>

      /// Właściwość koloru tła drugiego przycisku. Grupa stanowa. Widoczna w aplikacji jako '
Kolor 2'.

      /// </summary>

      [Askom.AddonManager.EvoObjectProperty(Description = "Kolor 2", PropertyGroup = Askom.Add
onManager.EvoObjectPropertyGroup.StateGroup)]

      public Color Color2

      {

         get { return button2.BackColor; }

         set { button2.BackColor = value; }

      }



      /// <summary>

      /// Dostęp do właściwosci standardowej "Zmienna główna" -
 wartość wyświetlana w polu edycyjnym

      /// </summary>

      [Askom.AddonManager.EvoObjectStandardProperty(Askom.AddonManager.EvoObjectStandardProper
tyName.MainVar)]

      public string MainVariable

      {

         set;

         get;

      }



      /// <summary>

      /// Dostęp do właściwosci standardowej "Zmienna sterowana" -
 skojarzenie jej z polem tekstowym textBox1
```

```
        /// </summary>
        [Askom.AddonManager.EvoObjectStandardProperty(Askom.AddonManager.EvoObjectStandardProperty
Name.ControlVar)]
        public string ControlVariable
        {
            get { return textBox1.Text; }
            set { textBox1.Text = value; }
        }

        /// <summary>
        /// Zdarzenie kliknięcia pierwszego przycisku. Grupa zdarzeń. Widoczne w aplikacji jako
'Wciśnięcie 1'.
        /// </summary>
        [Askom.AddonManager.EvoObjectEvent(Description = "Wciśnięcie 1")]
        public event Askom.AddonManager.EvoObjectEventHandler click1;

        /// <summary>
        /// Wciśnięcie pierwszego przycisku
        /// </summary>
        private void button1_Click(object sender, EventArgs e)
        {
            // Jeżeli jakieś funkcje są zarejestrowane to wywołujemy zdarzenie
            if (click1 != null)
                click1();
        }

        /// <summary>
        /// Wciśnięcie drugiego przycisku
        /// </summary>
        private void button2_Click(object sender, EventArgs e)
        {
            // Wysłanie sterowania zmiennej
            if(AsixEvo.Application != null)
                AsixEvo.Application.SetVariableValue(ControlVariable, textBox2.Text, checkBox1.Che
cked);
        }
```

```
      protected override void Dispose(bool disposing)
      {
         // Sprzątamy...
         if (disposing)
         {
            if (AsixEvo.Application != null)
            {
               AsixEvo.Application.CurrentApplicationLanguageChanged -
= Application_CurrentApplicationLanguageChanged;
            }

            button1.Click -= button1_Click;
            button2.Click -= button2_Click;
         }

         base.Dispose(disposing);
      }
   }
}
```



2.16 Przykład 2 – Praca ze zmiennymi

Poniższy kod prezentuje definicję obiektu dla AsixEvo, w którym wykorzystano opcję bezpośredniego dostępu do zmiennych, ich stanu, wartości i atrybutów. Obiekt składa się z trzech przycisków. Tekst pierwszego przycisku prezentuje wartości bieżące zmiennej systemowej „Counter”. Tekst drugiego przycisku pokazuje wartości bieżące zmiennej, której nazwa została podana jako wartość właściwości „Inna zmienna” obiektu. Dodatkowo, wartość ta jest poprzedzona tekstem pobranym z atrybutu „Opis” tej zmiennej. Tekst trzeciego przycisku pokazuje wartości bieżące zmiennej głównej obiektu.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;
using Askom.AddonManager;

namespace TestowaKontrolkaEvo
{
   /// <summary>
```

```
   /// Przykładowa klasa kontrolki obiektu dla Asix Evo korzystająca bezpośrednio ze zmiennych
.
   /// </summary>
   public class VarRefreshTest : Askom.AddonManager.EvoObjectBase
   {
      private System.Windows.Forms.Button button1;
      private System.Windows.Forms.Button button2;
      private System.Windows.Forms.Button button3;

      public VarRefreshTest()
      {
         // Inicjalizacja...

         SuspendLayout();

         // Przycisk 1
         button1 = new System.Windows.Forms.Button();
         button1.Dock = System.Windows.Forms.DockStyle.Top;
         button1.Name = "button1";
         button1.Text = "";
         Controls.Add(button1);

         // Przycisk 2
         button2 = new System.Windows.Forms.Button();
         button2.Dock = System.Windows.Forms.DockStyle.Fill;
         button2.Name = "button2";
         button2.Text = "";
         Controls.Add(button2);

         // Przycisk 3
         button3 = new System.Windows.Forms.Button();
         button3.Dock = System.Windows.Forms.DockStyle.Bottom;
         button3.Name = "button3";
         button3.Text = "";
         Controls.Add(button3);

         Name = "Odswiezanie";

         ResumeLayout(false);

         VariableStateChangedEvent += VarRefreshTest_VariableStateChangedEvent;
      }

      public override string EvoObjectID
      {
         get
         {
            return "Odswiezanie";  // Unikalny identyfikator obiektu
         }
      }

      public override string EvoObjectDescription
      {
         get
         {
            return "Nasluchiwanie zmiennych";  // Krótki opis obiektu
         }
      }

      string mDefaultVarName = "Counter";  // Counter jest zmienną systemową więc zawsze istni
eje.
      public override void Reset()
      {
         VariableState vs = GetVariableState(mDefaultVarName);
         if (vs != null)
         {
            RegisterVariable(mDefaultVarName); // Rejestrujemy zmienną wbudowaną
            button1.Text = vs.FormattedValue;  // Ustawiamy wartość aktualną
         }

         base.Reset();
      }

      public override void Disconnect()
      {
         UnregisterVariable(mDefaultVarName); // Wyrejestrowujemy zmienną wbudowaną
         base.Disconnect();
```

```
      }

      string mMainVarName = null;
      /// <summary>
      /// Dostęp do właściwosci standardowej "Zmienna główna" -
 wartość wyświetlana w polu edycyjnym
      /// </summary>
      [Askom.AddonManager.EvoObjectStandardProperty(Askom.AddonManager.EvoObjectStandardProper
tyName.MainVar)]
      public string MainVariable
      {
         set
         {
            UnregisterVariable(mMainVarName);  // Wyrejestrowujemy poprzednia zmienną
            mMainVarName = value;

            VariableState vs = GetVariableState(mMainVarName);
            if (vs != null)
            {
               RegisterVariable(mMainVarName);   // Rejestrujemy nową zmienną
               button3.Text = vs.FormattedValue; // Ustawiamy wartość aktualną
            }
         }
      }

      string mSecondVarName = null;
      string mSecondVarDescription = string.Empty;
      //// <summary>
      /// Właściwość napisu na drugim przycisku.
      /// </summary>
      [Askom.AddonManager.EvoObjectProperty(Description = "Inna zmienna", EditorType = EditorT
ype.Variable)]
      public string Var2
      {
         set
         {
            UnregisterVariable(mSecondVarName);  // Wyrejestrowujemy poprzednia zmienną
            mSecondVarName = value;

            VariableState vs = GetVariableState(mSecondVarName);
            if (vs != null)
            {
               object tob = GetVariableAttribute(mSecondVarName, "Description");
               if (tob != null)
                  mSecondVarDescription = tob.ToString();

               RegisterVariable(mSecondVarName); // Rejestrujemy nową zmienną
               button2.Text = mSecondVarDescription + " : " + vs.FormattedValue; // Ustawiamy
wartość aktualną
            }
         }

         get
         {
            return "DateTime"; // Wartość domyślna
         }
      }

      void VarRefreshTest_VariableStateChangedEvent(string aVariableName, VariableState aVaria
bleState)
      {
         if (string.Equals(aVariableName, mDefaultVarName, StringComparison.OrdinalIgnoreCase)
)
            button1.Text = aVariableState.FormattedValue;

         if (string.Equals(aVariableName, mSecondVarName, StringComparison.OrdinalIgnoreCase))
            button2.Text = mSecondVarDescription + " : " + aVariableState.FormattedValue;

         if (string.Equals(aVariableName, mMainVarName, StringComparison.OrdinalIgnoreCase))
            button3.Text = aVariableState.FormattedValue;
      }
   }
}
```

2.18 Przykład 3 – Obiekt wektorowy

Poniższy kod prezentuje definicję obiektu wektorowego dla AsixEvo. Obiekt jest wyświetlany jako prostokąt z ramką, zawierający czarny mniejszy prostokąt. Pozycja mniejszego prostokąta zależy od stanu obiektu. Stan obiektu można zmieniać, klikając w czarny prostokąt lub używając klawisza 'S'.

```
public class PaintedObjectTest : Askom.AddonManager.EvoObjectBase
   {
      bool switched = false;  // Stan obiektu

      public PaintedObjectTest()
      {
         // Inicjalizacja...
      }

      public override bool OwnerDraw
      {
         get
         {
            return true;  // Obiekt w trybie wektorowym (a nie kontrolkowym)
         }
      }

      public override string EvoObjectID
      {
         get
         {
            return "ObiektWektorowy";  // Unikalny identyfikator obiektu
         }
      }

      public override string EvoObjectDescription
```

```
        {
            get
            {
                return "Obiekt wektorowy";  // Krótki opis obiektu
            }
        }

        Color mColor1 = Color.Blue;
        //// <summary>
        /// Właściwość koloru tła
        /// </summary>
        [Askom.AddonManager.EvoObjectProperty(Description = "Kolor", EditorType = EditorType.Col
or, PropertyGroup = EvoObjectPropertyGroup.StateGroup)]
        public Color Color1
        {
            set
            {
                mColor1 = value;
                InvalidateObject();
            }

            get
            {
                return Color.Blue; // Wartość domyślna
            }
        }

        /// <summary>
        /// Obsługujemy zdarzenie malowania
        /// </summary>
        public override void Paint(Graphics aGraphics, Rectangle aArea)
        {
            using (Brush b = new SolidBrush(mColor1))
            {
                aGraphics.FillRectangle(b, aArea);  //Tło w kolorze właściwości
```

```
            }

            aGraphics.DrawRectangle(Pens.Red, aArea);  // Czerwona ramka

            // Obszar małego czarnego prostokąta zależy od stanu
            Rectangle swRect = Rectangle.Empty;
            if (switched)
            {
               swRect = new Rectangle((int)Math.Round((double)aArea.X + 0.25d * aArea.Width),
                                      (int)Math.Round((double)aArea.Y + 0.25d * aArea.Heig
ht),
                                      (int)Math.Round((double)0.25d * aArea.Width),
                                      (int)Math.Round((double)0.25d * aArea.Height));
            }
            else
            {
               swRect = new Rectangle((int)Math.Round((double)aArea.X + 0.5d * aArea.Width),
                                      (int)Math.Round((double)aArea.Y + 0.5d * aArea.Heigh
t),
                                      (int)Math.Round((double)0.25d * aArea.Width),
                                      (int)Math.Round((double)0.25d * aArea.Height));
            }

            aGraphics.FillRectangle(Brushes.Black, swRect); // Czarny prostokąt

         }

         /// <summary>
         /// Obsługujemy zdarzenie klikniecia myszą w obiekt
         /// </summary>
         public override void PerformMouseClick(System.Windows.Forms.MouseEventArgs e, Rectangle
aArea)
         {
            Rectangle swRect = Rectangle.Empty;  // Aktualny obszar małego czarnego prostokąta
            if (switched)
            {
```

```
            swRect = new Rectangle((int)Math.Round((double)aArea.X + 0.25d * aArea.Width),
                                   (int)Math.Round((double)aArea.Y + 0.25d * aArea.Heig
ht),
                                   (int)Math.Round((double)0.25d * aArea.Width),
                                   (int)Math.Round((double)0.25d * aArea.Height));
         }
         else
         {
            swRect = new Rectangle((int)Math.Round((double)aArea.X + 0.5d * aArea.Width),
                                   (int)Math.Round((double)aArea.Y + 0.5d * aArea.Heigh
t),
                                   (int)Math.Round((double)0.25d * aArea.Width),
                                   (int)Math.Round((double)0.25d * aArea.Height));
         }

         if (swRect.Contains(e.Location))
         {  // Jeżeli kliknieto w obszar małego czarnego prostokąta to przełączamy stan
            switched = !switched;
            InvalidateObject();
         }
      }

      public override void PerformKeyDown(System.Windows.Forms.KeyEventArgs e)
      {
         if (e.KeyCode == System.Windows.Forms.Keys.S)
         {  // Jeżeli wciśnięto klawisz 'S' to przełączamy stan
            switched = !switched;
            InvalidateObject();
         }
      }
   }
```