XML Serialization

Jarosław Kuchta

Ten dokument opisuje konstrukcję serializatora XML naśladującego standardowy XmlSerializer, ale wypełniający jego braki i zapewniający większą elastyczność.

# Potrzeba alternatywnego serializatora

Serializator XML firmy Microsoft pozostaje niezmieniony od początku mimo oczywistych braków, z których najważniejsze to:

* brak obsługi kolekcji generycznych (np. List<string>),
* brak obsługi słowników,
* brak obsługi typów i właściwości interfejsowych,
* brak serializacji kolekcji tylko do odczytu,
* brak obsługi referencji zwrotnych (gdy klasa odwołuje się do samej siebie bezpośrednio lub pośrednio),
* brak możliwości zarządzania kolejnością atrybutów,
* brak możliwości własnej konwersji właściwości złożonych na tekst.

XmlSerializer został napisany w celu szybkiego przekazywania danych przez sparowane operacje serializacji i deserializacji. Przy tworzeniu instancji serializatora podaje się typ, który ma być serializowany i deserializowany, oraz inne rozpoznawane typy. Typy są poddawane analizie strukturalnej i jest generowany wyspecjalizowany moduł implementujący operacje serializacji i deserializacji, które są wywoływane przez XmlSerializer. Moduł ten jest niewidoczny dla programisty, ale za to zapewnia wysoką wydajność zapisu/odczytu.

Problemy się pojawiają wówczas, gdy programista chce dopasować uzyskany format XML do swoich potrzeb, np. chce wygenerować plik XML do czytania przez inną aplikację, która narzuca swój format. Jest to trudne, a często wręcz niemożliwe. Jeśli struktura typu poddawanego analizie ma elementy, które nie są obsługiwane lub brak jest danych umożliwiających poprawną deserializację, to moduł implementujący nie zostanie wygenerowany i serializacja jest niemożliwa.

Inna kwestia to czytelność uzyskanego kodu XML dla człowieka. Standardowy serializator XML potrafi wstawiać do kodu nieoczekiwane atrybuty, jak własne deklaracje przestrzeni nazw czy określenia formatu dla dat. Domyślnie ignoruje spacje na początku i końcu łańcuchów tekstowych. Kolejność generowania atrybutów jest nieokreślona. To wszystko nieco zmniejsza czytelność.

W związku z powyższymi brakami powstało przynajmniej kilka alternatywnych serializatorów innych autorów. Najbardziej popularne to:

* SharpSerializer (910 tys. pobrań) – autor Paweł Idzikowski,
* ExtendedXmlSerializer (626 tys. pobrań) – autorzy Wojciech Nagórski i Michael DeMond,
* Platform.Xml.Serialization (321 tys. pobrań) – autor Thong Nguyen,
* XmlSerializerHelper (108 tys. pobrań) – autor Thomas Galliker,
* XSerializer (451 tys. pobrań) – autor Brian Friesen.

Jako alternatywę dla XML trzeba też rozważyć XAML, który jest specyficzną formą XML i został wprowadzony przez Microsoft jako format zapisu projektów komponentów wizualnych aplikacji WPF.

Warto też wspomnieć o alternatywnych względem XML formatach, jak JSON czy YAML.

## SharpSerializer

SharpSerializer to rozwiązanie open source dla zapisu danych w XML lub we własnym formacie binarnym. Ma on takie same możliwości jak standardowy XmlSerializer, a ponadto obsługuje:

* tablice wielowymiarowe, tablice zagnieżdżone, tablice tablic,
* właściwości polimorficzne, tj. takie, których wartości mogą być pochodne od zadeklarowanego typu właściwości,
* typy generyczne z atrybutami polimorficznymi (np. GenericClass<AbstractClass>),
* kolekcje i słowniki generyczne (np. List<string>, Dictionary<string, int>),
* kolekcje polimorficzne (np. Dictionary<IKeyInterface, AbstractClass>),
* kolekcje generyczne z argumentami polimorficznymi (np. IMyGenericInterface<MyAbstractClass>[,,]).

SharpSerializer został napisany dla potrzeb technologii Silverlight, gdzie nie było serializatora binarnego, zapewniając znacznie większą wydajność przekazywania danych niż w formacie tekstowym XML. Dzisiaj to rozwiązanie jest przestarzałe.

## ExtendedXmlSerializer

ExtendedXmlSerializer to konfigurowany serializer XML wspierający:

* deserializację XML uzyskanego ze standardowego serializatora Microsoftu,
* serializację klas, struktur, klas generycznych, typów prymitywnych, list i słowników generycznych, tablic i typów wyliczeniowych,
* serializację klas z właściwościami interfejsowymi,
* serializację referencji zwrotnych i referencji przez identyfikatory,
* deserializację starszych wersji XML,
* szyfrowanie właściwości,
* serializację dopasowaną przez użytkownika,
* rozpoznawanie atrybutów identyfikacyjnych: XmlElementAttribute, XmlRootAttribute, XmlTypeAttribute,
* rozpoznawanie dodatkowych atrybutów: XmlIgnoreAttribute, XmlAttributeAttribute, XmlEnumAttribute,
* własne metody zapisu/odczytu nawet, gdy klasa nie implementuje interfejsu IXmlSerializable.

Konfigurowalność serializatora ExtendedXmlSerializer oznacza, że można (i należy) tworzyć instancje serializatora przez utworzenie instancji klasy ConfigurationContainer i wywołanie szeregu jej metod. Mechanizm ten jest nazywany Fluent API. Przykład

IExtendedXmlSerializer serializer = new ConfigurationContainer().ConfigureType<Subject>()

.Member(x => x.Message)

.Name("Text")

.Create();

Takie podejście jest bardzo elastyczne, jednak pracochłonne. Przy dużej liczbie składowych użycie konfiguratora staje się osobnym projektem.

## Platform.Xml.Serialization

Ten serializator stanowi część biblioteki między-platformowej zawierającej użyteczne klasy i rozszerzenia dla C# i .NET. Biblioteka została napisana w latach 2003-2008 i wydana jako open source na licencji BSD w ramach organizacji Platform.NET. Copyright obejmuje lata 2003-2013.

Rozwiązanie przestarzałe. Na dodatek dokumentacja nie istnieje.

## XmlSerializerHelper

XmlSerializerHelper to nakładka na standardowy XmlSerializer. Daje możliwości dodawania metod rozszerzających. Dostępny na GitHubie. Dopasowany do .NET Standard 2.0.

Brak dokumentacji dyskwalifikuje projekt.

## XSerializer

XSerializer to uniwersalny serializator XML/JSON. Obsługuje właściwości i typy, których nie obsługuje standardowy serializator, takie jak interfejsy i słowniki. Może zapewnić proste zastąpienie standardowego serializatora Microsoftu. Używa tych samych atrybutów [XmlElement], [XmlAttribute] itp.

Przy serializacji JSON zapewnia większą wydajność niż serializator firmy Newtonsoft. Daje też lepsze wsparcie przy deserializacji zmiennych i pól typu dynamic.

Wspiera koncepcję szyfrowania pól i właściwości opatrzonych atrybutem [Encrypt] poprzez interfejs IEncryptionMechanism.

Aktualna wersja ma numer 0.4.4. Z tego wniosek, że sam autor nie uznaje projektu za dojrzały.

## XAML Reader/Writer

Alternatywą dla serializacji XML może być zapis/odczyt danych w formacie XAML. Ten format został wprowadzony przez Microsoft jako język oparty na XML opisu interfejsu użytkownika dla potrzeb Windows Presentation Foundation. XAML charakteryzuje się wysokim stopniem uporządkowania i niezawodną obsługą różnych konstrukcji programistycznych. Od „zwykłego” XML odróżnia go przede wszystkim:

* Automatyczna obsługa wielu różnych przestrzeni nazw – zarówno standardowych w .NET, jak i tworzonych przez programistę. Każdy zapisany obiekt jest przypisany do odpowiedniej przestrzeni nazw, w tym do nazw własnych przestrzeni aplikacji.
* Automatyczne zapisywanie prostych właściwości w postaci atrybutów XML.
* Możliwość definiowania konwerterów klas złożonych na wartości tekstowe (i z powrotem), co upraszcza zapisywanie takich klas.

Jeśli właściwość jest złożona, to jest zapisywana jako element XML (z nazwą identyfikującą klasę wartości) wewnątrz elementu, którego nazwa identyfikuje samą właściwość i obiekt nadrzędny. Daje to łatwą serializację właściwości polimorficznych, gdyż komponent zagnieżdżony może być różnych klas pochodnych. Przykład:

<MyClass …>

<MyClass.MyProperty>

<MyComponent …>

</MyComponent>

</MyClass.MyProperty>

</MyClass>

* Obsługa właściwości dołączanych (attached), czyli właściwości pewnej klasy (zazwyczaj zamkniętej) zaimplementowanych w innej klasie. Przykładem są właściwości Grid.Row i Grid.Column dołączane do każdego komponentu wizualnego.
* Obsługa rozszerzeń składniowych przez zapis w nawiasach klamrowych. Najczęściej wykorzystywane są rozszerzenia powiązań komponentów przez {Binding}. Powiązania są utrzymywane po załadowaniu komponentów i w ten sposób właściwości jednych komponentów wiążą się z innymi właściwościami innych komponentów tak, że zmiana jednych pociąga za sobą zmianę innych. Dla przykładu można powiązać treść wyświetlanego komponentu z zasobem aplikacji. Jeśli zasoby się zmienią (np. poprzez wybór języka użytkownika), to treść wyświetlana też się zmieni.

Początkowo pliki formacie XAML były edytowane ręcznie lub tworzone za pomocą narzędzi projektowania graficznego Visual Studio. Odczyt plików był automatyczny przy ładowaniu komponentów interfejsu użytkownika lub ręczny poprzez klasę XamlReader. Z czasem pojawiła się też klasa XamlWriter umożliwiająca programistom samodzielne zapisywanie zarówno komponentów interfejsu, jak instancji zwykłych klas (POCO).

Użycie formatu XAML jest ograniczone do .NET Framework, gdyż wymaga biblioteki PresentationFramework (niedostępnej w .NET Core). Podejmowane są próby otwartoźródłowych implementacji zapisu XAML np. w Portable.Xaml.XamlWriter.

## Formaty JSON i YAML

Format JSON (JavaScript Object Notation) to lekki format zapisu i wymiany danych obiektowych w postaci tekstowej będący podzbiorem składni języka JavaScript. Zarówno XML, jak i JSON są stosowane w technologiach usług webowych. Starsze technologie (SOAP, WCF) wykorzystywały XML, nowsze (REST) wykorzystują JSON. Pliki (i pakiety danych) JSON są mniejsze niż odpowiadające im pliki XML, gdyż nie zawierają znaczników końcowych. Co do łatwości czytania danych przez człowieka, zdania są różne. Z jednej strony struktura JSON jest bardziej przejrzysta, z drugiej trudniej jest odnaleźć koniec struktury (właśnie przez brak znaczników końcowych), co może mieć znaczenie przy dłuższych plikach.

Podobnym w strukturze do JSON jest YAML. Jest nieco bardziej czytelny dla człowieka, ale trudniejszy w parsowaniu.

Aktualnie dostępne są dwie implementacje serializatora JSON – jedna firmy Microsoft w przestrzeni nazw System.Text.Json, druga firmy Newtonsoft w przestrzeni nazw Newtonsoft.Json. Serializator Newtonsoft jest bardziej zaawansowany. Podstawowe różnice są następujące:

* Newtonsoft.Json potrafi ignorować wielkość liter przy deserializacji.
* Newtonsoft.Json zezwala na komentarze i spacje końcowe.
* System.Text.Json stosuje specjalny zapis znaków nie-ASCII i znaków charakterystycznych dla HTML dla zapobiegania atakom typu cross site scripting.
* System.Text.Json ma ograniczenie głębokości zagnieżdżenia struktur obiektowych do 64. Newtonsoft.Json nie ma ograniczenia.
* Newtonsoft.Json domyślnie serializuje pola (a nie tylko właściwości).
* Newtonsoft.Json i System.Text.Json stosują inne opcje do ignorowania wartości null.
* Newtonsoft.Json pozwala na serializację i deserializację liczb w cudzysłowach. Przy System.Text.Json można to osiągnąć przez opcje i atrybuty.
* System.Text.Json nie obsługuje atrybutów z przestrzeni nazw System.Runtime.Serialization, takich jako [DataMember] i [IgnoreDataMember].
* Oba serializatory różnią się w obsłudze referencji zwrotnych.
* System.Text.Json nie obsługuje typów Datatable, DbNull, TimeSpan, TimeZoneInfo, BigInteger.
* System.Text.Json jest bardziej wydajny przy serializacji i deserializacji od Newtonsoft.Json.

Brak znaczników klas elementów w JSON utrudnia obsługę właściwości i kolekcji polimorficznych. Wymaga to zaimplementowania własnych konwerterów rozróżniających klasy elementów przez serializację dodatkowych pól informujących o typach.

## Testy i wnioski

Przeprowadzono testy trzech serializatorów XML (dla których dostępna jest dokumentacja) oraz serializatora Newtonsoft.Json. Testu z XAML nie udało się przeprowadzić na projekcie, którego targetem jest .Net 6 (brak biblioteki).

Standardowy **XmlSerializer** pokazał błąd przy tworzeniu instancji:

**NotSupportedException**: Cannot serialize member DocumentModel.Settings.DocVars of type DocumentModel.DocumentVariables, because it implements IDictionary.

**SharpSerializer** wygenerował plik o nieoczekiwanej strukturze:

<Complex name="Root" type="DocumentModel.Document, DocumentModel, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null" id="1">

<Properties>

<Null name="Name" />

<Null name="Path" />

<Simple name="FullName" value="" />

<Complex name="DocumentProperties">

<Properties>

<Reference name="Document" id="1" />

<Simple name="Title" value="Tytuł dokumentu" />

**ExtendedXmlSerializer** wyświetlił komunikat:

**System.InvalidOperationException**: 'An attempt was made to format 'System.Nullable`1[System.Int32] ActiveRecord' as an attribute, but there is not a registered converter that can convert its values to a string. Please ensure a converter is registered for the type 'System.Nullable`1[System.Int32]' by adding a converter for this type to the converter collection in the ConverterExtension.'

Znaleziono w sieci wersję nowszą (3.7.6), jeszcze nie opublikowaną, co do której autor (Mike-E-angelo) zadeklarował:

Accounted for converter registration with nullable structures #534

Mimo tego, błąd występował dalej.

**Newtonsoft. JSON** serializator wygenerował plik poprawny, ale dla czytelności brakuje w nim nazw elementów:

{

"DocumentProperties": {

"Title": "Tytuł dokumentu",

"Subject": "Jakiś temat",

"Category": "Jakaś kategoria",

"ContentStatus": "Jakiś stan",

"Description": "To jest jakiś komentarz, nie bardzo wiadomo jaki.",

"Keywords": "CustomProperties, ExtendedProperties",

"Creator": "Jarosław Kuchta",

"Created": "2021-06-08T13:25:00+02:00",

"LastModifiedBy": "Jarosław Kuchta",

"Modified": "2021-06-08T13:32:00+02:00",

"LastPrinted": "2021-06-08T13:29:00+02:00",

"Revision": "3",

"Application": "Microsoft Office Word",

"ApplicationVersion": "16.0000",

"Company": "Jakaś firma",

"DocumentSecurity": "RecommendedReadOnly",

"HeadingPairs": [

{

"Name": "Tytuł",

"Count": 1

}

],

**Wniosek**: Żaden z analizowanych tu serializatorów nie nadaje się do testowanego rozwiązania. Modyfikacja istniejącego rozwiązania (open source) jest zazwyczaj trudna, a przy rozwiązaniu *żywym* (ExtendedXmlSerializer) nie wchodzi w grę.

**Wniosek końcowy**: potrzebny jest własny serializator XML.

# Wymagania dla własnego serializatora XML

Wymagania dla własnego serializatora XML są następujące:

1. Musi być możliwy do podstawienia w miejsce standardowego serializatora, tzn. musi mieć takie same konstruktory, właściwości i metody publiczne.
2. Musi mieć wszystkie możliwości standardowego serializatora, w tym:
   1. obsługiwać wszystkie atrybuty konfiguracyjne z przestrzeni nazw System.Xml.Serialization,
   2. obsługiwać metody serializacji warunkowej (ShouldSerialize),
   3. obsługiwać metody własnej serializacji z interfejsu IXmlSerializable,
   4. obsługiwać zdarzenia deserializacji (XmlDeserializationEvents),
   5. obsługiwać mapowanie typów (XmlMapping),
3. Musi wypełnić braki obsługi typów standardowego serializatora, w tym:
   1. przy serializacji kolekcji generycznych (np. List<string>) zapewnić opcjonalne pomijanie właściwości Capacity,
   2. przy deserializacji kolekcji elementów abstrakcyjnych i polimorficznych (np. List<AbstractClass>) rozpoznawać klasy pochodne umieszczone w tym samym asemblacie i dopuścić podawanie klas pochodnych z innych asemblatów,
   3. umożliwiać serializację i deserializację kolekcji tylko do odczytu,
   4. przy serializacji i deserializacji słowników umożliwić wybór jednej lub kilku właściwości elementu jako klucza tak, by można było zmienić słownik na kolekcję,
4. Musi zapewniać obsługę typów i właściwości interfejsowych przy podaniu klas implementujących.
5. Musi zapewniać wierność serializacji i deserializacji właściwości tekstowych:
   1. zawierających znaki Unicode,
   2. rozpoczynających się lub kończących się spacjami,
   3. zawierających znaki niewidoczne lub specjalne dla notacji XML.
6. Powinien zapewniać obsługę referencji, w tym referencji zwrotnych przy podaniu właściwości (jednej lub kilku) reprezentujących klucz obiektu.
7. Powinien zapewnić programiście możliwość zarządzania kolejnością atrybutów, a domyślnie generować je w kolejności deklaracji właściwości przy pierwszeństwie właściwości odziedziczonych z klas nadrzędnych.
8. Powinien zapewnić możliwość własnej konwersji właściwości złożonych na tekst.
9. Powinien zapewnić możliwość deklaracji wartości domyślnej [DefaultValue] właściwości (pomijanej przy serializacji).
10. Powinien zapewnić możliwość deklaracji właściwości reprezentującej zawartość elementu [ContentProperty], dla której serializator nie generuje nazwy właściwości.
11. Powinien zapewnić szerokie możliwości wyboru opcji serializacji i deserializacji, w tym:
    1. wyboru sposobu formatowania nazw elementów, atrybutów i wartości wyliczanych (bez zmian, zamiana na wielkie/małe litery, pierwsza litera wielka/mała) przy serializacji oraz uwzględniania/ignorowania wielkości liter przy deserializacji.
    2. wyboru lub pomijania pól wszystkich przy serializacji,
    3. wyboru lub pomijania właściwości nieoznaczonych atrybutami [XmlElement], [XmlAttribute],
    4. rozpoznawania lub ignorowania wybranych atrybutów z przestrzeni nazw System.Runtime.Serialization i System.ComponentModel.DataAnnotations,
    5. wyboru formatu serializacji daty i czasu (domyślnie międzynarodowy W3C),
    6. wyboru tekstu reprezentującego wartości logiczne true i false przy serializacji, a przy deserializacji kilku zestawów,
    7. kultury zapisu i odczytu danych liczbowych.
12. Wydajność serializacji i deserializacji nie jest krytyczna, ale wskazana jest optymalizacja wydajności przy zachowaniu funkcjonalności.

# Standardowe atrybuty sterujące serializacją

Ponieważ własny serializator ma zapewnić maksymalną możliwą zgodność ze standardowym serializatorem, więc potrzeba zapewnić obsługę atrybutów z przestrzeni nazw System.Xml.Serialization. Ponadto w wymaganiach znajduje się obsługa atrybutów z przestrzeni nazw System.Runtime.Serialization oraz System.­ComponentModel.DataAnnotations. Warto też wspomnieć o atrybutach z przestrzeni nazw System.ComponentModel, gdzie jest zadeklarowany atrybut DefaultValue.

Wszystkie one są przedstawione kolejnych podrozdziałach tego rozdziału.

## Atrybuty z przestrzeni System.Xml.Serialization

Tab. . Atrybuty sterujące serializacją w przestrzeni nazw System.Xml.Serialization

|  |  |  |  |
| --- | --- | --- | --- |
| Atrybut | Target | Znaczenie | Uwagi |
| XmlAnyAttribute() | property, field, parameter, returnValue | Określa, że tablica obiektów typu XmlAttribute (lub XmlNode) przy deserializacji zawiera nieznane atrybuty XML. |  |
| XmlAnyElement( Name: string, Namespace: string, Order: int) | property, field, parameter, returnValue | Określa, że tablica obiektów typu XmlElement(lub XmlNode) przy deserializacji zawiera nieznane elementy XML. | Dozwolonych wiele atrybutów XmlAnyElement |
| XmlArray( ElementName: string, Namespace: string, Form: XmlSchemaForm, IsNullable: bool, Order: int) | property, field, parameter, returnValue, które dają tablicę obiektów złożonych | Określa, że wartość jest serializowana jako tablica elementów XML. | Współdziała z atrybutami XmlArrayItem. |
| XmlArrayItem( ElementName: string, Namespace: string, Form: XmlSchemaForm, DataType: string, Type: Type?, IsNullable: bool, NestingLevel: int) | property, field, parameter, returnValue | Określa typy pochodne, które mogą być wstawiane do tabeli. | Dozwolonych wiele atrybutów XmlArrayItem. Współdziałają z atrybutem XmlArray. |
| XmlAttribute( AttributeName: string, Namespace: string, Form: XmlSchemaForm, DataType: string, Type: type) | property, field, parameter, returnValue | Wymusza serializację wartości przez atrybut XML. | Nie ma parametru Order w odróżnieniu od XmlElement i XmlArray. |
| XmlChoiceIdentifier( MemberName: string) | property, field, parameter, returnValue | Umożliwia określenie typu danych jednej składowej przez inną składową typu. | Współdziała z atrybutami XmlElement. |
| XmlElement ( ElementName: string, Namespace: string, Form: XmlSchemaForm, DataType: string, Type: type, IsNullable: bool, Order: int) | property, field, parameter, returnValue | Wymusza serializację wartości przez element XML. | Dozwolonych wiele atrybutów XmlElement. |
| XmlEnum( Name: string) | field | Umożliwia określenie nazwy, pod którą jest serializowana wartość typu wyliczeniowego. | Stosowane tylko w typach wyliczeniowych |
| Xmlgnore() | property, field | Zabrania serializacji publicznej składowej typu. |  |
| XmlInclude( Type: Type) | class, struct, interface, method | Podaje typy rozpatrywane przy serializacji i deserializacji wartości danego typu. | Dozwolonych wiele atrybutów XmlInclude. |
| XmlRoot( ElementName: string, Namespace: string, DataType: string, IsNullable: bool,) | class, struct, interface, enum, returnValue | Steruje serializacją celu atrybutu jako elementu głównego XML. | Nie ma parametu Type w odróżnieniu od innych atrybutów z parametrem DataType. |
| XmlText( DataType: string, Type: type) | property, field, parameter, returnValue | Nakazuje serializowanie wartości jako tekstu XML |  |
| XmlType( TypeName: string, Namespace: string, AnonymousType: bool, IncludeInSchema: bool) | class, struct, interface, enum | Steruje schematem XML, który jest generowany, gdy obiekt docelowy atrybutu jest serializowany. |  |

Przypisywanie atrybutów do parametrów i wartości zwracanych dotyczy metod publicznych dla dokumentów Web Services Description Language (WSDL).

Parametr Order występujący w niektórych atrybutach określa kolejność, w jakiej składowe typu są serializowane. Dotyczy to elementów, ale nie atrybutów.

Parametry Name i Namespace wymuszają nazwy i przestrzenie nazw elementów przy serializacji i filtrują elementy przy deserializacji.

Parametr typu XmlSchemaForm określa, czy nazwa elementu jest serializowana w formie nieokreślonej (None), kwalifikowanej (Qualified), czy niekwalifikowanej (Unqualified) przestrzenią nazw XML.

Parametr IsNullable wymusza serializowanie z atrybutem xsi:nil="true", gdy element jest pusty.

Parametr DataType określa standardowy typ XSD i może przyjmować jedną z wartości z tab. 2. Zamiast tego można podać odpowiadający mu typ danych .NET przez parametr Type.

|  |  |
| --- | --- |
| Tab. . Odwzorowanie typów XSD na typy danych .NET | |
| | typ XSD | typ danych .NET | | --- | --- | | anyURI | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | base64Binary | Array of [Byte](https://docs.microsoft.com/en-us/dotnet/api/system.byte?view=net-6.0) objects | | boolean | [Boolean](https://docs.microsoft.com/en-us/dotnet/api/system.boolean?view=net-6.0) | | byte | [SByte](https://docs.microsoft.com/en-us/dotnet/api/system.sbyte?view=net-6.0) | | date | [DateTime](https://docs.microsoft.com/en-us/dotnet/api/system.datetime?view=net-6.0) | | dateTime | [DateTime](https://docs.microsoft.com/en-us/dotnet/api/system.datetime?view=net-6.0) | | decimal | [Decimal](https://docs.microsoft.com/en-us/dotnet/api/system.decimal?view=net-6.0) | | double | [Double](https://docs.microsoft.com/en-us/dotnet/api/system.double?view=net-6.0) | | ENTITY | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | ENTITIES | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | float | [Single](https://docs.microsoft.com/en-us/dotnet/api/system.single?view=net-6.0) | | gDay | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | gMonth | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | gMonthDay | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | gYear | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | gYearMonth | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | hexBinary | Array of [Byte](https://docs.microsoft.com/en-us/dotnet/api/system.byte?view=net-6.0) objects | | ID | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | IDREF | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | IDREFS | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | int | [Int32](https://docs.microsoft.com/en-us/dotnet/api/system.int32?view=net-6.0) | | integer | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | | typ XSD | typ danych .NET | | --- | --- | | language | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | long | [Int64](https://docs.microsoft.com/en-us/dotnet/api/system.int64?view=net-6.0) | | Name | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | NCName | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | negativeInteger | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | NMTOKEN | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | NMTOKENS | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | normalizedString | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | nonNegativeInteger | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | nonPositiveInteger | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | NOTATION | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | positiveInteger | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | QName | [XmlQualifiedName](https://docs.microsoft.com/en-us/dotnet/api/system.xml.xmlqualifiedname?view=net-6.0) | | duration | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | string | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | short | [Int16](https://docs.microsoft.com/en-us/dotnet/api/system.int16?view=net-6.0) | | time | [DateTime](https://docs.microsoft.com/en-us/dotnet/api/system.datetime?view=net-6.0) | | token | [String](https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0) | | unsignedByte | [Byte](https://docs.microsoft.com/en-us/dotnet/api/system.byte?view=net-6.0) | | unsignedInt | [UInt32](https://docs.microsoft.com/en-us/dotnet/api/system.uint32?view=net-6.0) | | unsignedLong | [UInt64](https://docs.microsoft.com/en-us/dotnet/api/system.uint64?view=net-6.0) | | unsignedShort | [UInt16](https://docs.microsoft.com/en-us/dotnet/api/system.uint16?view=net-6.0) | |

W przestrzeni nazw System.Xml.Serialization są też zdefiniowane atrybuty zaczynające się od "Soap", które odpowiadają atrybutom zaczynającym się od "Xml". Atrybuty Soap są wykorzystywane przy serializacji XML do przekazywania zdalnych wywołań w protokole Soap, ale za pomocą atrybutów Xml też można sterować serializacją wywołań w protokole Soap.

## Atrybuty z przestrzeni System.Runtime.Serialization

Inna możliwość sterowania serializacją XML to wykorzystanie atrybutów i klas z przestrzeni nazw System.Runtime.Serialization (zadeklarowanej w System.­Runtime.­Serialization.Primitives.dll) która ma zastosowanie pierwszoplanowe do sterowania serializacją i deserializacją przez obiekt DataContractSerializer podczas przekazywania danych i wywołań w usługach WCF.

Tab. . Atrybuty z przestrzeni nazw System.Runtime.Serialization

|  |  |  |  |
| --- | --- | --- | --- |
| Atrybut | Target | Znaczenie | Uwagi |
| CollectionDataContract( Name: string, Namespace: string, ItemName: string, KeyName: string, ValueName: string, IsReference: bool …) | class, struct | Po zastosowaniu do typu kolekcji włącza niestandardową specyfikację elementów elementu kolekcji. Ten atrybut można zastosować tylko do typów rozpoznawanych przez DataContractSerializer jako prawidłowe kolekcje z możliwością serializacji. | Dalsze parametry są postaci  Is\_\_\_SetExplicitly,  gdzie w miejscu \_\_\_ występuje nazwa „zwykłego” parametru. |
| ContractNamespace( ClrNamespace: string, ContractNamespace: string) | assembly, module | Określa przestrzeń nazw CLR i przestrzeń nazw XML kontraktu danych. |  |
| DataContract( Name: string, Namespace: string, IsReference: bool …) | class, struct, enum | Określa, że typ definiuje lub implementuje kontrakt danych i jest serializowalny przez serializator, taki jak DataContractSerializer. Aby można było serializować ich typ, autorzy typów muszą zdefiniować kontrakt danych dla ich typu. | Dalsze parametry są postaci  Is\_\_\_SetExplicitly,  gdzie w miejscu \_\_\_ występuje nazwa „zwykłego” parametru. |
| DataMember( Name: string, EmitDefaultValue: bool, IsRequired: bool, Order: int, IsNameSetExplicitly: bool) | property, field | Określa przestrzeń nazw CLR i przestrzeń nazw XML kontraktu danych. |  |
| EnumMember( Value: string, IsValueSetExplicitly: bool) | field | Określa, że pole typu wyliczeniowego powinno być serializowane z daną nazwą. |  |
| IgnoreDataMember() | property, field | Określa, że element członkowski nie jest częścią kontraktu danych i nie jest serializowany. |  |
| KnownType( Type: Type, MethodName: string) | class, struct | Określa typy, które powinny być rozpoznawane przez DataContractSerializer podczas serializacji lub deserializacji danego typu. |  |
| OnDeserialized() | method | Określa, że metoda jest wywoływana natychmiast po deserializacji obiektu na grafie obiektu. |  |
| OnDeserializing() | method | Określa, że metoda jest wywoływana podczas deserializacji obiektu na grafie obiektu. |  |
| OnSerialized() | method | Określa, że metoda jest wywoływana po serializacji obiektu na grafie obiektu. |  |
| OnSerializing() | method | Określa, że metoda jest wywoływana podczas serializacji obiektu na grafie obiektu. |  |
| OptionalField( Type: Type, MethodName: string) | field | Określa, że jeśli pola brakuje w strumieniu serializacji, to elementy BinaryFormatter i SoapFormatter nie zgłaszają wyjątku. |  |

## Atrybuty z przestrzeni System.ComponentModel

W przestrzeni nazw System.ComponentModel jest m.in. zadeklarowany atrybut DefaultValue, który może być wykorzystywane przy serializacji. Wiele innych atrybutów dotyczy czasu projektowania komponentów, okna właściwości, kontrolki PropertyGrid, wiązań danych, czy też źródeł danych. W Tab. 4 przedstawiono ich kompletną listę jako potencjalne źródło inspiracji przy sterowaniu serializacją we własnym serializatorze.

Tab. . Atrybuty z przestrzeni nazw System.ComponentModel

|  |  |  |  |
| --- | --- | --- | --- |
| Atrybut | Target | Znaczenie | Uwagi |
| AmbientValue(…) | all | Określa wartość, która ma być przekazywana do właściwości, aby spowodować, że właściwość pobierze jej wartość z innego źródła. |  |
| AttributeProvider(…) | property | Włącza przekierowywanie atrybutów. |  |
| Bindable(…) | all | Określa, czy składowa jest zwykle używana do powiązania. | dotyczy wiązań danych |
| Browsable(…) | all | Określa, czy właściwość lub zdarzenie powinny być wyświetlane w oknie Właściwości. | dotyczy okna właściwości |
| Category(…) | all | Określa nazwę kategorii, w której ma być grupowana właściwość lub zdarzenie, gdy jest wyświetlane w kontrolce PropertyGrid ustawionej na Tryb kategorii. | dotyczy kontrolki PropertyGrid |
| ComplexBindingProperties(…) | class | Określa właściwości źródła danych i składowej danych dla komponentu obsługującego złożone powiązanie danych. | dotyczy wiązań danych |
| DataObject(…) | class | Identyfikuje typ jako obiekt odpowiedni do powiązania z obiektem ObjectDataSource . | dotyczy źródeł danych |
| DataObjectField(…) | property | Udostępnia metadane dla właściwości reprezentującej pole danych. | dotyczy źródeł danych |
| DataObjectMethod(…) | method | Określa metodę operacji danych uwidacznianą przez typ, typ operacji wykonywanej przez metodę i określa, czy metoda jest domyślną metodą danych. | dotyczy źródeł danych |
| DefaultBindingProperty(…) | class | Określa domyślną właściwość powiązania dla komponentu. | dotyczy wiązań danych |
| DefaultEvent(…) | class | Określa domyślne zdarzenie komponentu. |  |
| DefaultProperty(…) | class | Określa właściwość domyślną komponentu |  |
| DefaultValue(…) | all | Określa wartość domyślną właściwości. |  |
| Description(…) | all | Określa opis właściwości lub zdarzenia. |  |
| Designer(…) | class, interface | Określa klasę używaną do implementowania usług czasu projektowania dla komponentu. | dotyczy czasu projektowania |
| DesignerCategory(…) | class | Określa, że projektant klasy należy do określonej kategorii. | dotyczy czasu projektowania |
| DesignerSerializationVisibility(…) | property, field, event, method | Określa typ trwałości do użycia podczas serializacji właściwości w komponencie w czasie projektowania. | dotyczy czasu projektowania |
| DesignOnly(…) | all | Określa, czy właściwość można ustawić tylko w czasie projektowania. | dotyczy czasu projektowania |
| DesignTimeVisible(…) | class, interface | Określa, czy projektant wizualizacji może pokazać ten komponent. | dotyczy czasu projektowania |
| DisplayName(…) | class, property, event, method, | Określa nazwę wyświetlaną właściwości, zdarzenia lub metody bezparametrowej |  |
| Editor(…) | all | Określa edytor do zmiany właściwości. | dotyczy czasu projektowania |
| EditorBrowsable(…) | class, struct, interface, enum,  property, field, event, delegate, method, constructor, | Określa, że element jest widoczny w edytorze. | dotyczy czasu projektowania |
| ExtenderProvidedProperty(…) | all | Określa właściwość oferowaną przez dostawcę rozszerzenia. |  |
| ImmutableObject(…) | all | Określa, że obiekt nie ma podwłaściwości, które mogą być edytowane. |  |
| Inheritance(…) | property, field, event | Wskazuje, czy składowa została odziedziczona z klasy bazowej. |  |
| InitializationEvent(…) | class | Określa, które zdarzenie jest zgłaszane podczas inicjowania. |  |
| InstallerType(…) | class | Określa instalator typu, który instaluje komponenty. |  |
| LicenseProvider(…) | class | Określa dostawcę licencji , który ma być używany z klasą. |  |
| ListBindable(…) | all | Określa, że lista może być używana jako źródło danych. | dotyczy wiązań danych |
| Localizable(…) | all | Określa, czy właściwość lub parametr powinny być zlokalizowane. |  |
| LookupBindingProperties(…) | class | Określa właściwości, które obsługują powiązanie oparte na odnośnikach. | dotyczy wiązań danych |
| MergableProperty(…) | all | Określa, że właściwość może być połączona z właściwościami należącymi do innych obiektów w oknie Właściwości. | dotyczy czasu projektowania |
| NotifyParentProperty(…) | property | Wskazuje, że właściwość nadrzędna jest powiadamiana o modyfikacji wartości właściwości, do którego zastosowano ten atrybut. |  |
| ParenthesizePropertyName(…) | all | Wskazuje, czy nazwa skojarzonej właściwości jest wyświetlana z nawiasami w oknie Właściwości. | dotyczy okna właściwości |
| PasswordPropertyText(…) | all | Wskazuje, że reprezentacja tekstu obiektu jest zaciemniana znakami, takimi jak gwiazdki. |  |
| PropertyTab(…) | all | Identyfikuje kartę właściwości lub karty do wyświetlenia dla określonej klasy lub klas. | dotyczy kontrolki PropertyGrid |
| ProvideProperty(…) | class | Określa nazwę właściwości, którą klasa implementująca interfejs IExtenderProvider ofert oferuje innym komponentom. |  |
| ReadOnly(…) | all | Określa, czy właściwość jest tylko do odczytu, czy też do odczytu/zapisu. |  |
| RecommendedAsConfigurable(…) | property | Określa, że właściwość może być używana jako ustawienie aplikacji. |  |
| RefereshProperties(…) | all | Wskazuje, że siatka właściwości powinna zostać odświeżona po zmianie skojarzonej wartości właściwości. | dotyczy kontrolki PropertyGrid |
| RunInstaller(…) | class | Określa, czy podczas instalowania asemblatu należy wywołać instalator akcji niestandardowej. |  |
| SettingsBindable(…) | property | Określa, kiedy właściwość komponentu może być powiązana z ustawieniem aplikacji. | dotyczy wiązań danych |
| ToolboxItem(…) | all | Reprezentuje atrybut elementu przybornika. | dotyczy przybornika |
| ToolboxItemFilter(…) | class | Określa łańcuch filtru i typ filtru do użycia dla elementu przybornika. | dotyczy przybornika |
| TypeConverter(…) | all | Określa typ, który ma być używany jako konwerter dla elementu. |  |
| TypeDescriptionProvider(…) |  | Określa niestandardowego dostawcę opisu typu dla klasy. | klasa abstrakcyjna |

## Atrybuty System.ComponentModel.DataAnnotations

Kolejny zestaw atrybutów jest zdefiniowany w przestrzeni nazw System.­ComponentModel.DataAnnotations. Są to atrybuty, które nie sterują serializacją i deserializacją bezpośrednio, ale niektóre z nich mogą być wykorzystane pośrednio do tego celu. Atrybuty można podzielić na kilka kategorii. Są atrybuty pochodne od Validation, które służą do walidacji danych. Są pochodne DataType, które określają typ danych. Jest kilka atrybutów określających sposób prezentacji, czy też edycji danych w interfejsie użytkownika. Najbardziej obiecujący jest atrybut Key, który może być wykorzystywany przy serializacji słownika.

Tab. . Atrybuty z przestrzeni nazw System.ComponentModel.DataAnnotation

|  |  |  |  |
| --- | --- | --- | --- |
| Atrybut | Target | Znaczenie | Uwagi |
| Association(…) | field, property | Określa, że składowa reprezentuje relację danych, taką jak relacja klucza obcego. | przestarzałe |
| Compare(…) | property | Umożliwia walidację przez porównanie z inną właściwością. | pochodna Validation |
| ConcurrencyCheck() | field, property | Określa, że właściwość uczestniczy w optymistycznych kontrolach współbieżności. |  |
| CreditCard() | field, property, parameter | Określa, że wartość pola danych jest numerem karty kredytowej. | pochodna DataType |
| CustomValidation(…) | class, field, method, parameter, property | Określa niestandardową metodę weryfikacji, która jest używana do weryfikowania właściwości lub wystąpienia klasy. | pochodna Validation |
| DataType(…) | property, field, method, parameter | Określa nazwę dodatkowego typu do skojarzenia z polem danych. | pochodna Validation |
| Display(…) | property, field, method, parameter | Określa sposób wyświetlania składowej |  |
| DisplayColumn(…) | class | Określa kolumnę wyświetlaną w tabeli określanej jako kolumna klucza obcego. |  |
| DisplayFormat(…) | property, field | Określa sposób wyświetlania i formatowania pól danych. |  |
| Editable(…) | property, field | Wskazuje, czy pole danych można edytować. |  |
| EmailAddress() | property, field, parameter | Weryfikuje pole jako adres e-mail. | pochodna DataType |
| EnumDataType() | property, field, parameter, method | Umożliwia mapowanie typu wyliczeniowego na kolumnę danych. | pochodna DataType |
| FileExtension(…) | property, field, parameter | Weryfikuje rozszerzenia nazw plików. | pochodna DataType |
| FilterUIHint( | property, field | Określania zachowania filtrowania dla kolumny. | przestarzałe |
| Key() | property, field | Określa co najmniej jedną właściwość, która jednoznacznie identyfikuje jednostkę. |  |
| MaxLength(…) | property, field, parameter | Określa maksymalną długość danych tablicy lub łańcucha we właściwości. | pochodna Validation |
| MetadataType(…) | class | Określa klasę metadanych do skojarzenia z klasą modelu danych. |  |
| MinLength(…) | property, field, parameter | Określa minimalną długość danych tablicy lub łańcucha we właściwości. | pochodna Validation |
| Phone() | property, field, parameter | Określa, że wartość pola danych jest poprawnie sformułowanym numerem telefonu. | pochodna DataType |
| Range(…) | property, field, parameter | Określa ograniczenia zakresu liczbowego dla wartości pola danych. | pochodna Validation |
| RegularExpression(…) | property, field, parameter | Określa, że wartość pola danych musi być zgodna z określonym wyrażeniem regularnym. | pochodna Validation |
| Required() | property, field, parameter | Określa, że wartość pola danych jest wymagana. | pochodna Validation |
| ScaffoldColumn(…) | property, field | Określa, czy klasa lub kolumna danych używa szkieletu (scaffolding). |  |
| StringLength(…) | property, field, parameter | Określa minimalną i maksymalną długość znaków, które są dozwolone w polu danych. | pochodna Validation |
| Timestamp() | property, field | Określa, że dana kolumna określa wersję wiersza. |  |
| UIHint(…) | property, field | Określa szablon lub kontrolkę użytkownika używaną do wyświetlania pola danych. |  |
| Url(…) | property, field, parameter | Zapewnia walidację adresu URL. | pochodna Validation |
| Validation(…) |  | Służy jako klasa podstawowa dla wszystkich atrybutów weryfikacji. | klasa abstrakcyjna |

# Standardowe interfejsy serializacji

Dla utrzymania zgodności własnego serializatora z klasą standardowego serializatora (System.Xml.Serialization.XmlSerializer) konieczne jest dopasowanie się do interfejsu tej klasy. Interfejs tego serializatora składa się z konstruktorów, metod publicznych i zdarzeń (nie ma właściwości) i jest przedstawiony w podrozdziale pierwszym.

## Interfejs klasy XmlSerializer

Przy tworzeniu serializatora klasy XmlSerializer podaje się typ (klasę) obiektu , który będzie serializowany jako korzeń drzewa XML. Można też podać inne typy. Typy te są analizowane poprzez wyszukanie pól i właściwości publicznych i generowana jest klasa XmlSerializerImplementation, która jest kompilowana do osobnego asemblatu.

Nazwę wygenerowanego asemblatu można pobrać przez metodę GetXmlSerializerAssemblyName. W teorii można taki asemblat załadować, ale Microsoft zastrzega, że metody klasy XmlSerializerImplementation nie są przeznaczone do samodzielnego wywoływania.

Zamiast tego wywołuje się przeciążone metody Serialize i Deserialize instancji klasy XmlSerializer, które jest przekazywane do wygenerowanej klasy XmlSerializerImplementation w sposób przezroczysty dla programisty.

W poniższym przykładzie najpierw tworzona jest instancja klasy XmlSerializer dla określonego złożonego typu danych, a następnie poprzez metodę Serialize obiekt tego typu danych jest serializowana do dokumentu XML, który jest zapisywany do pliku tekstowego poprzez element TextWriter. Obiekt jest serializowany jako korzeń drzewa XML. Wraz z obiektem są serializowane wszystkie jego obiekty składowe jako elementy podrzędne.

DataType dataObject = …;

XmlSerializer serializer = new XmlSerializer(typeof(DataType));

using (TextWriter writer = File.CreateText(filename))

{

serializer.Serialize(writer, dataObject);

}

Analogicznie metoda Deserialize wczytuje korzeń drzewa XML z dokumentu XML wraz elementami zagnieżdżonymi. Jeśli wszystko się zgadza, to tworzy obiekt typu podanego przy inicjacji serializatora, tworzy i przypisuje mu obiekty składowe. Ponieważ wynik metody Deserialize jest ogólnego typu Object, więc musi być rzutowany do oczekiwanego typu:

DataType dataObject;

XmlSerializer serializer = new XmlSerializer(typeof(DataType));

using (TextReader reader = File.OpenText(filename))

{

dataObject = (DataType)serializer.Deserialize(reader);

}

XmlSerializer nie implementuje interfejsu IDisposable, więc sam nie może wystąpić w instrukcji using. Za to elementy przechowujące dokument XML przy serializacji i deserializacji są oparte o strumień (Stream), mogą wystąpić w instrukcji using, co zazwyczaj powoduje zamknięcie strumienia po zakończeniu serializacji/deserializacji.

Instancje klasy XmlSerializer tworzy się albo za pomocą jednego z konstruktorów pokazanych w tab. 6, albo za pomocą jednej z trzech metod statycznych z tab. 7.

Tab. . Konstruktory klasy XmlSerializer

|  |  |
| --- | --- |
| Konstruktor | Znaczenie |
| XmlSerializer () | Inicjuje nowe wystąpienie klasy XmlSerializer. |
| XmlSerializer (Type type) | Inicjuje nowe wystąpienie klasy XmlSerializer, które może serializować obiekty określonego typu w dokumentach XML i deserializować dokumenty XML do obiektów określonego typu. |
| XmlSerializer (Type type, String defaultNamespace) | Inicjuje nowe wystąpienie klasy XmlSerializer, które może serializować obiekty określonego typu w dokumentach XML i deserializować dokumenty XML do obiektów określonego typu. Określa domyślną przestrzeń nazw dla wszystkich elementów XML. |
| XmlSerializer (Type type, Type[] extraTypes) | Inicjuje nowe wystąpienie klasy XmlSerializer, które może serializować obiekty określonego typu w dokumentach XML i deserializować dokumenty XML do obiektu określonego typu. Jeśli właściwość lub pole zwraca tablicę, to parametr extraTypes określa obiekty, które można wstawić do tej tablicy. |
| XmlSerializer (Type type, XmlAttributeOverrides overrides) | Inicjuje nowe wystąpienie klasy XmlSerializer, które może serializować obiekty określonego typu w dokumentach XML i deserializować dokumenty XML do obiektów określonego typu. Każdy obiekt do serializacji może zawierać wystąpienia klas, które to przeciążenie może zastąpić innymi klasami. |
| XmlSerializer (Type type, XmlRootAttribute root) | Inicjuje nowe wystąpienie klasy XmlSerializer, które może serializować obiekty określonego typu w dokumentach XML i deserializować dokument XML do obiektu określonego typu. Określa również klasę stosowaną jako element główny XML. |
| XmlSerializer (Type type, XmlAttributeOverrides overrides, Type[] extraTypes, XmlRootAttribute root, String  defaultNamespace) | Inicjuje nowe wystąpienie klasy XmlSerializer, które może serializować obiekty typu Object w wystąpieniach dokumentów XML i deserializować wystąpienia dokumentów XML do obiektów typu Object. Każdy obiekt, który ma być serializowany, może zawierać wystąpienia klas, które to przeciążenie zastępuje przez inne klasy. To przeciążenie określa również domyślną przestrzeń nazw dla wszystkich elementów XML i klasę stosowaną jako element główny XML. |
| XmlSerializer (Type type, XmlAttributeOverrides overrides, Type[] extraTypes, XmlRootAttribute root, String defaultNamespace, String location) | Inicjuje nowe wystąpienie klasy XmlSerializer, które może serializować obiekty typu Object w wystąpieniach dokumentów XML i deserializować wystąpienia dokumentów XML do obiektów typu Object. Każdy obiekt, który ma być serializowany, może zawierać wystąpienia klas, które to przeciążenie zastępuje przez inne klasy. To przeciążenie określa również domyślną przestrzeń nazw dla wszystkich elementów XML i klasę stosowaną jako element główny XML. Ostatni parametr określa lokalizację typów: |
| XmlSerializer (XmlTypeMapping) | Inicjuje wystąpienie klasy XmlSerializer przy użyciu obiektu, który mapuje jeden typ na inny. |

Tab. . Metody statyczne klasy XmlSerializer

|  |  |
| --- | --- |
| Metoda | Znaczenie |
| FromMappings(XmlMapping[], Type) | Metoda statyczna. Zwraca instancję klasy XmlSerializer utworzoną na podstawie tablicy mapowania XML oraz podanego typu. |
| FromMappings(XmlMapping[]) | Metoda statyczna. Zwraca tablicę instancji klasy XmlSerializer utworzoną na podstawie tablicy mapowania XML. |
| FromTypes (Type[]? types) | Metoda statyczna. Zwraca tablicę instancji klasy XmlSerializer utworzoną na podstawie tablicy typów. |

Metody publiczne klasy XmlSerializer to przede wszystkim szereg przeciążonych metod Serialize i Deserialize o różnych zestawach parametrów. Zapis dokumentu XML tworzonego przy serializacji następuje przez obiekty typu Stream, TextWriter XmlWriter, albo XmlSerializerWriter. Odczyt dokumentu XML przy deserializacji następuje z obiektów typu Stream, TextReader, XmlReader, albo XmlSerializerReader. Tak elastyczny dobór przeznaczenia/źródła dokumentu XML umożliwia serializację/deserializację do/z pliku, pamięci, łańcucha znaków, przy zapisie tekstowym lub binarnym. Umożliwia też podawanie opcji zapisu/odczytu XML (np. serializację z wcięciami).

Metodą CanDeserialize można też sprawdzić, czy dokument XML nadaje się do deserializacji.

Tab. . Metody publiczne klasy XmlSerializer

|  |  |
| --- | --- |
| Metoda | Znaczenie |
| bool CanDeserialize (XmlReader) | Określa czy ten serializator może deserializować dokument dostępny przez dany element XmlReader. |
| XmlSerializationReader CreateReader () | Po przesłonięciu w klasie pochodnej zwraca obiekt używany do deserializacji. W klasie standardowego serializatora nie zaimplementowana. |
| XmlSerializationWriter CreateWriter () | Po przesłonięciu w klasie pochodnej zwraca obiekt używany do serializacji. W klasie standardowego serializatora nie zaimplementowana. |
| Deserialize (Stream) | Deserializuje dokument XML dostępny przez określony element Stream. |
| Deserialize (TextReader) | Deserializuje dokument XML dostępny przez określony element TextReader. |
| Deserialize (XmlSerializationReader) | Deserializuje dokument XML dostępny przez określony element XmlSerializationReader. |
| Deserialize (XmlReader  ) | Deserializuje dokument XML dostępny przez określony element XmlReader. |
| Deserialize (XmlReader, String style) | Deserializuje dokument XML dostępny przez określony element XmlReader w określonym stylu kodowania. |
| Deserialize (XmlReader , XmlDeserializationEvents) | Deserializuje dokument XML dostępny przez określony element XmlReader umożliwiając zastępowanie zdarzeń występujących podczas deserializacji. |
| Deserialize (XmlReader , String style, XmlDeserializationEvents) | Deserializuje dokument XML dostępny przez określony element XmlReader w określonym stylu kodowania umożliwiając zastępowanie zdarzeń występujących podczas deserializacji. |
| GetXmlSerializerAssemblyName(Type) | Zwraca nazwę asemblatu zawierającego jedną lub kilka wersji klasy XmlSerializer utworzonej specjalnie do serializowania i deserializowania określonego typu. |
| GetXmlSerializerAssemblyName(Type, String) | Zwraca nazwę asemblatu zawierającego jedną lub kilka wersji klasy XmlSerializerImplementation utworzonej specjalnie do serializowania i deserializowania określonego typu w określonej przestrzeni nazw. |
| Serialize(XmlWriter, Object) | Serializuje określony obiekt i zapisuje dokument XML w pliku przy użyciu określonego elementu XmlWriter. |
| Serialize(XmlWriter, Object, XmlSerializerNamespaces, String encodingStyle, String id) | Serializuje określony obiekt i zapisuje dokument XML w pliku przy użyciu elementu XmlWriter, przestrzeni nazw XML i kodowania. W przypadku komunikatów zakodowanych za pomocą protokołu SOAP używa bazowego id do generowania atrybutów identyfikatorów. |
| Serialize(XmlWriter, Object, XmlSerializerNamespaces, String) | Serializuje określony obiekt i zapisuje dokument XML w pliku przy użyciu elementu XmlWriter, przestrzeni nazw XML i kodowania. |
| Serialize(XmlWriter, Object, XmlSerializerNamespaces) | Serializuje określony obiekt i zapisuje dokument XML w pliku przy użyciu elementu XmlWriter i przestrzeni nazw XML. |
| Serialize(TextWriter, Object, XmlSerializerNamespaces) | Serializuje określony obiekt i zapisuje dokument XML w pliku przy użyciu elementu TextWriter i przestrzeni nazw XML. |
| Serialize(Object, XmlSerializationWriter) | Serializuje określony obiekt i zapisuje dokument XML przy użyciu elementu XmlSerializationWriter. |
| Serialize(TextWriter, Object) | Serializuje określony obiekt i zapisuje dokument XML w pliku przy użyciu określonego elementu TextWriter. |
| Serialize(Stream, Object) | Serializuje określony obiekt i zapisuje dokument XML w pliku przy użyciu określonego strumienia. |
| Serialize(Stream, Object, XmlSerializerNamespaces) | Serializuje określony obiekt i zapisuje dokument XML w pliku przy użyciu określonego strumienia i określonych przestrzeni nazw. |

Parametr id podany dla jednej z metod serializacji oznacza łańcuch bazowy używany do tworzenia identyfikatorów protokołu SOAP. Domyślnie są to "id1", "id2" itd. Jeśli jednak parametr ma wartość "myBase", to generowane są identyfikatory "myBaseid1", "myBaseid2" itd.

Parametr encodingStyle podawany dla niektórych metod przeciążonych powinien mieć wartość "http://schemas.xmlsoap.org/soap/encoding/" dla kodowania SOAP w wersji 1.1. albo "http://www.w3.org/2001/12/soap-encoding" dla kodowania SOAP w wersji 1.2.

Klasy XmlSerializationReader i XmlSerializationWriter należą do infrastruktury serializatora tworzonymi podczas inicjacji instancji klasy XmlSerializer. Nie powinny być używane bezpośrednio przez programistów. Dwie wirtualne metody, CreateReader i CreateWriter, umożliwiają utworzenie własnych instancji XmlSerializerReader i XmlSerializerWriter w klasach potomnych od XmlSerializer, ale z kolei klasy XmlSerializerReader i XmlSerializerWriter nie są przeznaczone do tworzenia klas potomnych.

Klasa XmlNamespaces, podawana często przy serializacji, umożliwia utrzymanie jednolitego odwzorowania prefiksów nazw XML na przestrzenie nazw XML przy wielokrotnej serializacji różnych obiektów. Domyślnie serializer, jeśli ma używać nazwy kwalifikowane w elementach XML, samodzielnie przypisuje prefiksy różnym przestrzeniom nazw XML. Programista może jednak utworzyć obiekt XmlNamespaces, który jest kolekcją nazw kwalifikowanych typu XmlQualifiedName, może go wypełnić i podać przy serializacji. W ten sposób różne dokumenty XML mają takie same prefiksy nazw XML dla takich samych przestrzeni nazw XML.

Obiekt klasy XmlDeserializationEvents podawany przy pewnych metodach Deserializacji umożliwia pewną reakcję aplikacji na napotykane w dokumencie XML nieznane atrybuty, elementy i węzły XML. Zdarzenia przekazywane w ten sposób są wyszczególnione w tab. 9

Tab. . Zdarzenia przekazywane przez klasę XmlDeserializationEvents

|  |  |
| --- | --- |
| Metoda | Znaczenie |
| XmlAttributeEventHandler UnknownAttribute | Występuje, gdy serializator napotka atrybut XML nieznanego typu podczas deserializacji. |
| XmlElementEventHandler UnknownElement | Występuje, gdy serializator napotka element XML nieznanego typu podczas deserializacji. |
| XmlNodeEventHandler UnknownNode | Występuje, gdy serializator napotka węzeł XML nieznanego typu podczas deserializacji. |
| UnreferencedObjectEventHandler UnreferencedObject | Występuje podczas deserializacji strumienia XML zakodowanego przy użyciu protokołu SOAP, gdy serializator napotka rozpoznany typ, który nie jest używany lub do którego nie ma odwołań. |

## Interfejs klasy DataContractSerializer

Wprawdzie w wymaganiach dotyczących własnego serializatora nie ma utrzymania zgodności z serializatorem DataContractSerializer z przestrzeni nazw System.Runtime.Serialization, ale ten serializator obsługuje sytuacje nieobsługiwane przez XmlSerializer, więc warto zapoznać się z interfejsem tego serializatora.

Konstruktory klasy DataContractSerializer (tab. 10) są podobne do konstruktorów klasy XmlSerializer (tab. 6). Parametry klasy XmlDictionaryString wyrażają łańcuchy znaków przechowywane w słowniku i identyfikowane przez klucz.

Tab. . Konstruktory klasy DataContractSerializer

|  |  |
| --- | --- |
| Konstruktor | Znaczenie |
| DataContractSerializer(Type) | Inicjuje nowe wystąpienie klasy DataContractSerializer w celu serializacji lub deserializacji obiektu określonego typu. |
| DataContractSerializer(Type, String, String, IEnumerable<Type>) | Inicjuje nowe wystąpienie klasy DataContractSerializer w celu serializacji lub deserializacji obiektu określonego typu. Ta metoda określa również główny element XML i przestrzeń nazw w dwóch parametrach, a także listę znanych typów, które mogą być obecne w grafie obiektu. |
| DataContractSerializer(Type, XmlDictionaryString, XmlDictionaryString) | Inicjuje nowe wystąpienie klasy DataContractSerializer w celu serializacji lub deserializacji obiektu określonego typu przy użyciu elementu głównego XML i przestrzeni nazw określonej za pomocą parametrów typu XmlDictionaryString. |
| DataContractSerializer(Type, String, String) | Inicjuje nowe wystąpienie klasy DataContractSerializer w celu serializacji lub deserializacji obiektu określonego typu przy użyciu podanego elementu głównego XML i przestrzeni nazw. |
| DataContractSerializer(Type, DataContractSerializerSettings) | Inicjuje nowe wystąpienie klasy DataContractSerializer, aby serializować lub deserializować obiekt określonego typu i ustawień. |
| DataContractSerializer(Type, IEnumerable<Type>) | Inicjuje nowe wystąpienie klasy DataContractSerializer w celu serializacji lub deserializacji obiektu określonego typu oraz kolekcji znanych typów, które mogą znajdować się w grafie obiektu. |
| DataContractSerializer(Type, XmlDictionaryString, XmlDictionaryString, IEnumerable<Type>) | Inicjuje nowe wystąpienie klasy DataContractSerializer w celu serializacji lub deserializacji obiektu określonego typu. Ta metoda określa również główny element XML i przestrzeń nazw w dwóch XmlDictionaryString parametrach, a także listę znanych typów, które mogą być obecne w grafie obiektu. |

Klasa DataContractSerializerSettinges reprezentuje ustawienia serializacji (tab. 11). Ustawienia te, z wyjątkiem RootName i RootNamespace, są też właściwościami klasy DataContractSerializer.

Tab. . Ustawienia serializacji reprezentowane przez klas DataContractSerializerSettings

|  |  |
| --- | --- |
| Właściwość | Znaczenie |
| DataContractResolver | Pobiera lub ustawia składnik używany do dynamicznego mapowania deklaracji xsi:type na znane typy kontraktów. |
| IgnoreExtensionDataObject | Pobiera lub ustawia wartość określającą, czy ignorować dane dostarczone przez rozszerzenie klasy, gdy klasa jest serializowana lub deserializacji. |
| KnownTypes | Pobiera lub ustawia kolekcję typów, które mogą być obecne w grafie obiektu serializowane przy użyciu tego wystąpienia elementu DataContractSerializerSettings. |
| MaxItemsInObjectGraph | Pobiera lub ustawia maksymalną liczbę elementów w grafie obiektu w celu serializacji lub deserializacji. |
| PreserveObjectReferences | Pobiera lub ustawia wartość określającą, czy używać nietypowych konstrukcji XML w celu zachowania danych referencyjnych obiektów. |
| RootName | Pobiera lub ustawia nazwę główną wybranego obiektu. |
| RootNamespace | Pobiera lub ustawia przestrzeń nazw katalogu głównego dla określonego obiektu. |
| SerializeReadOnlyTypes | Pobiera lub ustawia wartość określającą, czy serializować typy tylko do odczytu. |

Metody klasy DataContractSerializer (tab. 12) to szereg przeciążonych metod WriteObject i ReadObject odpowiadających metodom Serialize i Deserialize klasy XmlSerializer oraz metody IsStartObject odpowiadające metodom CanDeserialize.

W odróżnieniu od metod Serialize i Deserialize klasy XmlSerializer metody WriteObject i ReadObject klasy DataContractSerializer mają wersje wywołania z parametrem XmlDictionaryWriter i XmlDictionaryReader dla zapisu/odczytu binarnych wersji XML.

Parametr klasy DataContractResolver reprezentuje narzędzie rozpoznawania dla deklaracji mapowania xsi:type w czasie wykonania.

Klasa DataContractSerializer jest klasą pochodną od abstrakcyjnej klasy XmlObjectSerializer i wiele z jej metod (tab. 12) to implementacje abstrakcyjnych metod klasy XmlObjectSerializer (co pokazano w ostatniej kolumnie).

Tab. . Metody klasy DataContractSerializer

|  |  |  |
| --- | --- | --- |
| Metoda | Znaczenie | ^ |
| IsStartObject (XmlDictionaryReader) | Określa, czy obiekt XmlDictionaryReader jest umieszczony na obiekcie, który można deserializować. | • |
| IsStartObject (XmlReader) | Określa, czy obiekt XmlReader jest umieszczony na obiekcie, który można deserializować. | • |
| ReadObject (Stream) | Odczytuje strumień XML lub dokument z obiektem Stream i zwraca zdeserializowany obiekt. | • |
| ReadObject (XmlDictionaryReader) | Odczytuje dokument XML lub strumień za pomocą obiektu XmlDictionaryReader i zwraca obiekt deserializowany. | • |
| ReadObject (XmlDictionaryReader, Boolean) | Odczytuje strumień XML z obiektem XmlDictionaryReader i zwraca obiekt deserializowany, a także określa, czy jest wykonywane sprawdzenie nazwy obiektu przed odczytaniem jego wartości. | • |
| ReadObject (XmlDictionaryReader, Boolean, DataContractResolver) | Odczytuje dokument XML lub strumień dokumentu i zwraca obiekt deserializowany. Metoda zawiera parametr określający, czy nazwa obiektu jest weryfikowana, oraz narzędzie rozpoznawania deklaracji mapowania xsi:type w czasie wykonania. |  |
| ReadObject (XmlReader) | Odczytuje strumień XML z obiektem XmlReader i zwraca obiekt deserializowany. | • |
| ReadObject (XmlReader, Boolean) | Odczytuje strumień XML z obiektem XmlReader i zwraca obiekt deserializowany, a także określa, czy jest wykonywane sprawdzenie nazwy obiektu przed odczytaniem jego wartości. | • |
| WriteEndObject (XmlDictionaryWriter) | Zapisuje zamykający element XML przy użyciu elementu XmlDictionaryWriter. | • |
| WriteEndObject (XmlWriter) | Zapisuje zamykający element XML przy użyciu elementu XmlWriter. | • |
| WriteObject (Stream, Object) | Zapisuje pełną zawartość (początek, zawartość i koniec) obiektu w dokumencie XML lub strumieniu przy użyciu określonego elementu Stream. | • |
| WriteObject (XmlDictionaryWriter, Object) | Zapisuje pełną zawartość (początek, zawartość i koniec) obiektu w dokumencie XML lub strumieniu przy użyciu określonego elementu XmlDictionaryWriter. | • |
| WriteObject (XmlDictionaryWriter, Object, DataContractResolver) | Zapisuje wszystkie dane obiektu (początkowy element XML, zawartość i element otaczający) do dokumentu XML lub strumienia przy użyciu określonego elementu XmlDictionaryWriter. Metoda zawiera narzędzie rozpoznawania dla deklaracji mapowania xsi:type w czasie wykonania. |  |
| WriteObject (XmlWriter, Object) | Zapisuje wszystkie dane obiektu (początkowy element XML, zawartość i element zamykający) do dokumentu XML lub strumienia za pomocą elementu XmlWriter. |  |
| WriteObjectContent (XmlDictionaryWriter, Object) | Zapisuje zawartość XML przy użyciu elementu XmlDictionaryWriter. |  |
| WriteObjectContent (XmlWriter, Object) | Zapisuje zawartość XML przy użyciu elementu XmlWriter. | • |
| WriteStartObject (XmlDictionaryWriter, Object) | Zapisuje otwierający element XML przy użyciu elementu XmlDictionaryWriter. |  |
| WriteStartObject (XmlWriter, Object) | Zapisuje otwierający element XML przy użyciu elementu XmlWriter. |  |

## Klasa XmlDictionaryWriter

Klasa XmlDictionaryWriter jest klasą abstrakcyjną, pochodzącą od klasy XmlWriter, która ułatwia serializację w Windows Communication Foundation (WCF). Większość właściwości (tab. 13) i metod (tab. 14) dziedziczy po klasie XmlWriter (co pokazano w ostatniej kolumnie).

Tab. . Właściwości klasy XmlDictionaryWriter

|  |  |  |
| --- | --- | --- |
| Właściwość | Znaczenie | ^ |
| CanCanonicalize | Ta właściwość zawsze zwraca wartość false. Klasy pochodne mogą ją przesłonić, aby zwracać true , jeśli obsługują kanonizację. |  |
| Settings | Obiekt użyty do utworzenia ustawień XmlWriterSettings tego wystąpienia XmlWriter. | • |
| WriteState | Po zastąpieniu w klasie pochodnej pobiera stan składnika zapisywania. | • |
| XmlLang | Po zastąpieniu w klasie pochodnej pobiera bieżący zakres xml:lang. | • |
| XmlSpace | Po zastąpieniu w klasie pochodnej pobiera element XmlSpace reprezentujący bieżący zakres xml:space. | • |

Tab. . Metody klasy XmlDictionaryWriter

|  |  |  |
| --- | --- | --- |
| Metoda | Znaczenie | ^ |
| Close () | Gdy przesłonięta w klasie pochodnej, zamyka ten strumień i bazowy strumień. | • |
| CreateBinaryWriter (Stream) | Tworzy wystąpienie klasy XmlDictionaryWriter, które zapisuje dane w binarnym formacie XML. |  |
| CreateBinaryWriter (Stream, IXmlDictionary) | Tworzy wystąpienie klasy XmlDictionaryWriter, które zapisuje dane w binarnym formacie XML z użyciem słownika IXmlDictionary. |  |
| CreateBinaryWriter (Stream, IXmlDictionary, XmlBinaryWriterSession) | Tworzy wystąpienie klasy XmlDictionaryWriter, które zapisuje dane w binarnym formacie XML z użyciem słownika IXmlDictionary i poprzedniej sesji serializacji. |  |
| CreateBinaryWriter (Stream, IXmlDictionary, XmlBinaryWriterSession, Boolean) | Tworzy wystąpienie klasy XmlDictionaryWriter, które zapisuje dane w binarnym formacie XML z użyciem słownika IXmlDictionary, poprzedniej sesji serializacji i możliwym zamknięciem strumienia po zakończeniu zapisywania. |  |
| CreateDictionaryWriter (XmlWriter) | Tworzy wystąpienie klasy XmlDictionaryWriter na podstawie istniejącego elementu XmlWriter. |  |
| CreateMtomWriter (Stream, Encoding, Int32, String) | Tworzy wystąpienie klasy XmlDictionaryWriter, które zapisuje kod XML w formacie MTOM. |  |
| CreateMtomWriter (Stream, Encoding, Int32, String, String, String, Boolean, Boolean) | Tworzy wystąpienie klasy XmlDictionaryWriter, które zapisuje kod XML w formacie MTOM. |  |
| CreateTextWriter (Stream) | Tworzy wystąpienie klasy XmlDictionaryWriter, które zapisuje dane w formacie tekstowym XML. |  |
| CreateTextWriter (Stream, Encoding) | Tworzy wystąpienie klasy XmlDictionaryWriter, które zapisuje dane w formacie tekstowym XML z użyciem podanego kodowania. |  |
| CreateTextWriter (Stream, Encoding, Boolean) | Tworzy wystąpienie klasy XmlDictionaryWriter, które zapisuje dane w formacie tekstowym XML z użyciem podanego kodowania i możliwym zamknięciem strumienia po zakończeniu zapisywania. |  |
| Dispose() | Zwalnia wszystkie zasoby używane przez bieżące wystąpienie klasy XmlWriter. | • |
| Dispose(Boolean) | Zwalnia zasoby niezarządzane używane przez element XmlWriter i opcjonalnie zwalnia zasoby zarządzane. | • |
| DisposeAsync() | Wykonuje zadania zdefiniowane przez aplikację skojarzone z zwalnianiem, zwalnianiem lub resetowaniem niezarządzanych zasobów asynchronicznie. | • |
| DisposeAsyncCore() | Wykonuje zadania zdefiniowane przez aplikację skojarzone z zwalnianiem, zwalnianiem lub resetowaniem niezarządzanych zasobów asynchronicznie. | • |
| EndCanonicalization () | Po zaimplementowaniu przez klasę pochodną zatrzymywany jest tryb zapisu kanonicznego rozpoczęta przez odpowiednie wywołanie StartCanonicalization(Stream, Boolean, String[]). |  |
| Flush () | Po zastąpieniu w klasie pochodnej opróżnia wszystkie elementy w buforze do bazowych strumieni, a także opróżnia bazowy strumień. | • |
| FlushAsync () | Asynchronicznie opróżnia dowolny element w buforze do bazowych strumieni, a także opróżnia bazowy strumień. | • |
| LookupPrefix (String) | Po zastąpieniu w klasie pochodnej zwraca najbliższy prefiks zdefiniowany w bieżącym zakresie przestrzeni nazw dla identyfikatora URI przestrzeni nazw. | • |
| StartCanonicalization (Stream, Boolean, String[]) | Po zaimplementowaniu przez klasę pochodną wchodzi w tryb zapisu kanonicznego. |  |
| WriteArray (String, String, String, Boolean[], Int32, Int32) | Zapisuje węzły z tablicy Boolean. |  |
| WriteArray (String, String, String, DateTime[], Int32, Int32) | Zapisuje węzły z tablicy DateTime. |  |
| WriteArray (String, String, String, Decimal[], Int32, Int32) | Zapisuje węzły z tablicy Decimal. |  |
| WriteArray (String, String, String, Double[], Int32, Int32) | Zapisuje węzły z tablicy Double. |  |
| WriteArray (String, String, String, Guid[], Int32, Int32) | Zapisuje węzły z tablicy Guid. |  |
| WriteArray (String, String, String, Int16[], Int32, Int32) | Zapisuje węzły z tablicy Int16. |  |
| WriteArray (String, String, String, Int32[], Int32, Int32) | Zapisuje węzły z tablicy Int32. |  |
| WriteArray (String, String, String, Int64[], Int32, Int32) | Zapisuje węzły z tablicy Int64. |  |
| WriteArray (String, String, String, Single[], Int32, Int32) | Zapisuje węzły z tablicy Single. |  |
| WriteArray (String, String, String, TimeSpan[], Int32, Int32) | Zapisuje węzły z tablicy TimeSpan. |  |
| WriteArray (String, XmlDictionaryString, XmlDictionaryString, Boolean[], Int32, Int32) | Zapisuje węzły z tablicy Boolean. |  |
| WriteArray (String, XmlDictionaryString, XmlDictionaryString, DateTime[], Int32, Int32) | Zapisuje węzły z tablicy DateTime. |  |
| WriteArray (String, XmlDictionaryString, XmlDictionaryString, Decimal[], Int32, Int32) | Zapisuje węzły z tablicy Decimal. |  |
| WriteArray (String, XmlDictionaryString, XmlDictionaryString, Double[], Int32, Int32) | Zapisuje węzły z tablicy Double. |  |
| WriteArray (String, XmlDictionaryString, XmlDictionaryString, Guid[], Int32, Int32) | Zapisuje węzły z tablicy Guid. |  |
| WriteArray (String, XmlDictionaryString, XmlDictionaryString, Int16[], Int32, Int32) | Zapisuje węzły z tablicy Int16. |  |
| WriteArray (String, XmlDictionaryString, XmlDictionaryString, Int32[], Int32, Int32) | Zapisuje węzły z tablicy Int32. |  |
| WriteArray (String, XmlDictionaryString, XmlDictionaryString, Int64[], Int32, Int32) | Zapisuje węzły z tablicy Int64. |  |
| WriteArray (String, XmlDictionaryString, XmlDictionaryString, Single[], Int32, Int32) | Zapisuje węzły z tablicy Single. |  |
| WriteArray (String, XmlDictionaryString, XmlDictionaryString, TimeSpan[], Int32, Int32) | Zapisuje węzły z tablicy TimeSpan. |  |
| WriteAttributes(XmlReader, Boolean) | Po przesłonięciu w klasie pochodnej zapisuje wszystkie atrybuty znalezione w bieżącej pozycji w obiekcie XmlReader. | • |
| WriteAttributesAsync (XmlReader, Boolean) | Asynchronicznie zapisuje wszystkie atrybuty znalezione w bieżącej pozycji w elemencie XmlReader. | • |
| WriteAttributeString (String, String) | Po zastąpieniu w klasie pochodnej zapisuje atrybut z określoną lokalną nazwą i wartością. | • |
| WriteAttributeString (String, String, String) | Po zastąpieniu w klasie pochodnej zapisuje atrybut z określoną lokalną nazwą, identyfikatorem URI przestrzeni nazw i wartością. |  |
| WriteAttributeString (String, String, String, String) | Po zastąpieniu w klasie pochodnej zapisuje atrybut z określonym prefiksem, nazwą lokalną, identyfikatorem URI przestrzeni nazw i wartością. | • |
| WriteAttributeString (String, XmlDictionaryString, XmlDictionaryString, String) | Zapisuje kwalifikowaną nazwę i wartość atrybutu. |  |
| WriteAttributeString (XmlDictionaryString, XmlDictionaryString, String) | Zapisuje kwalifikowaną nazwę i wartość atrybutu. |  |
| WriteAttributeStringAsync (String, String, String, String) | Asynchronicznie zapisuje atrybut z określonym prefiksem, nazwą lokalną, identyfikatorem URI przestrzeni nazw i wartością. | • |
| WriteBase64 (Byte[], Int32, Int32) | Po zastąpieniu w klasie pochodnej koduje określone bajty binarne jako Base64 i zapisuje wynikowy tekst. | • |
| WriteBase64Async (Byte[], Int32, Int32) | Asynchronicznie koduje określone bajty binarne jako Base64 i zapisuje wynikowy tekst. |  |
| WriteBinHex (Byte[], Int32, Int32) | Po zastąpieniu w klasie pochodnej koduje określone bajty binarne jako BinHex i zapisuje wynikowy tekst. | • |
| WriteBinHexAsync (Byte[], Int32, Int32) | Asynchronicznie koduje określone bajty binarne jako BinHex i zapisuje wynikowy tekst. | • |
| WriteCData (String) | W przypadku zastąpienia w klasie pochodnej zapisuje element <![ CDATA[...]]> zawierający określony tekst. | • |
| WriteCDataAsync (String) | Asynchronicznie zapisuje element <![ CDATA[...]]>zawierający określony tekst. | • |
| WriteCharEntity (Char) | Po zastąpieniu w klasie pochodnej wymusza generowanie encji znaku dla określonej wartości znaku Unicode. | • |
| WriteCharEntityAsync (Char) | Asynchronicznie wymusza generowanie encji znaku dla określonej wartości znaku Unicode. | • |
| WriteChars (Char[], Int32, Int32) | Podczas zastępowania w klasie pochodnej zapisuje tekst jeden bufor naraz. | • |
| WriteCharsAsync (Char[], Int32, Int32) | Asynchronicznie zapisuje tekst po jednym znaku naraz. | • |
| WriteComment (String) | Podczas zastępowania w klasie pochodnej zapisuje komentarz <!--...--> zawierający określony tekst. | • |
| WriteCommentAsync (String) | Asynchronicznie zapisuje komentarz <!--...--> zawierający określony tekst. | • |
| WriteDocType (String, String, String, String) | Po zastąpieniu w klasie pochodnej, zapisuje deklarację DOCTYPE z określoną nazwą i atrybutami opcjonalnymi. | • |
| WriteDocTypeAsync (String, String, String, String) | Asynchronicznie zapisuje deklarację DOCTYPE o określonej nazwie i atrybutach opcjonalnych. | • |
| WriteElementString (String, String) | Zapisuje element o określonej nazwie lokalnej i wartości. | • |
| WriteElementString (String, String, String) | Zapisuje element o określonej nazwie lokalnej, identyfikatorze URI przestrzeni nazw i wartości. | • |
| WriteElementString (String, String, String, String) | Zapisuje element z określonym prefiksem, nazwą lokalną, identyfikatorem URI przestrzeni nazw i wartością. | • |
| WriteElementString (String, XmlDictionaryString, XmlDictionaryString, String) | Zapisuje element z zawartością tekstową. |  |
| WriteElementString (XmlDictionaryString, XmlDictionaryString, String) | Zapisuje element z zawartością tekstową. |  |
| WriteElementStringAsync (String, String, String, String) | Asynchronicznie zapisuje element z określonym prefiksem, nazwą lokalną, identyfikatorem URI przestrzeni nazw i wartością. | • |
| WriteEndAttribute () | Po zastąpieniu w klasie pochodnej zamyka poprzednie wywołanie WriteStartAttribute(String, String). | • |
| WriteEndAttributeAsync () | Asynchronicznie zamyka poprzednie wywołanie WriteStartAttribute(String, String). | • |
| WriteEndDocument () | Po zastąpieniu w klasie pochodnej, zamyka wszystkie otwarte elementy lub atrybuty i wprowadza moduł zapisywania z powrotem w stan Start. | • |
| WriteEndDocumentAsync () | Asynchronicznie zamyka wszystkie otwarte elementy lub atrybuty i przywraca stan modułu zapisywania. | • |
| WriteEndElement () | Po zastąpieniu w klasie pochodnej zamyka jeden element i wyświetla odpowiadający mu zakres przestrzeni nazw. | • |
| WriteEndElementAsync () | Asynchronicznie zamyka jeden element i wyświetla odpowiedni zakres przestrzeni nazw. | • |
| WriteEntityRef (String) | Po zastąpieniu w klasie pochodnej zapisuje odwołanie do encji jako &name;. | • |
| WriteEntityRefAsync (String) | Asynchronicznie zapisuje odwołanie do encji jako &name;. | • |
| WriteFullEndElement () | Po zastąpieniu w klasie pochodnej zamyka jeden element i przywraca odpowiadający mu zakres przestrzeni nazw. | • |
| WriteFullEndElementAsync () | Asynchronicznie zamyka jeden element i przywraca odpowiedni zakres przestrzeni nazw. | • |
| WriteName (String) | Po zastąpieniu w klasie pochodnej zapisuje określoną nazwę, upewniając się, że jest prawidłową nazwą zgodnie z zaleceniem W3C XML 1.0 (https://www.w3.org/TR/1998/REC-xml-19980210#NT-Name). | • |
| WriteNameAsync (String) | Asynchronicznie zapisuje określoną nazwę, upewniając się, że jest prawidłową nazwą zgodnie z zaleceniem W3C XML 1.0 (https://www.w3.org/TR/1998/REC-xml-19980210#NT-Name). | • |
| WriteNmToken (String) | Po zastąpieniu w klasie pochodnej zapisuje określoną nazwę, upewniając się, że jest to prawidłowy token NmToken zgodnie z zaleceniem W3C XML 1.0 (https://www.w3.org/TR/1998/REC-xml-19980210#NT-Name). | • |
| WriteNmTokenAsync (String) | Asynchronicznie zapisuje określoną nazwę, zapewniając, że jest prawidłowym elementem NmToken zgodnie z zaleceniem W3C XML 1.0 (https://www.w3.org/TR/1998/REC-xml-19980210#NT-Name). | • |
| WriteNode (XmlDictionaryReader, Boolean) | Zapisuje bieżący węzeł XML z elementu XmlDictionaryReader. |  |
| WriteNode (XmlReader, Boolean) | Zapisuje bieżący węzeł XML z elementu XmlReader. |  |
| WriteNode (XPathNavigator, Boolean) | Kopiuje wszystko z obiektu XPathNavigator do modułu zapisywania. Pozycja w XPathNavigator pozostaje niezmieniona. | • |
| WriteNodeAsync (XmlReader, Boolean) | Asynchronicznie kopiuje wszystko z XmlReader do XmlWriter i przenosi XmlReader na początek następnego elementu rodzeństwa. | • |
| WriteNodeAsync (XPathNavigator, Boolean) | Asynchronicznie kopiuje wszystko z obiektu XPathNavigator do modułu zapisywania. Pozycja w XPathNavigator pozostaje niezmieniona. | • |
| WriteProcessingInstruction (String, String) | Po przesłonięciu w klasie pochodnej zapisuje instrukcję przetwarzania z odstępem między nazwą a tekstem w następujący sposób: <?name text?>. | • |
| WriteProcessingInstructionAsync (String, String) | Asynchronicznie zapisuje instrukcję przetwarzania z odstępem między nazwą a tekstem w następujący sposób: <?name text?>. | • |
| WriteQualifiedName (String, String) | Po przesłonięciu w klasie pochodnej zapisuje nazwę kwalifikowaną przez przestrzeń nazw. Ta metoda wyszukuje prefiks, który znajduje się w zakresie dla danej przestrzeni nazw. | • |
| WriteQualifiedName (XmlDictionaryString, XmlDictionaryString) | Zapisuje kwalifikowaną nazwę przestrzeni nazw. Ta metoda wyszukuje prefiks, który jest w zakresie dla danej przestrzeni nazw. |  |
| WriteQualifiedNameAsync (String, String) | Asynchronicznie zapisuje nazwę kwalifikowaną przez przestrzeń nazw. Ta metoda wyszukuje prefiks, który znajduje się w zakresie dla danej przestrzeni nazw. | • |
| WriteRaw (Char[], Int32, Int32) | Po przesłonięciu w klasie pochodnej ręcznie zapisuje nieprzetworzone znaczniki z bufora znaków. | • |
| WriteRaw (String) | Po przesłonięciu w klasie pochodnej ręcznie zapisuje nieprzetworzone znaczniki z łańcucha. | • |
| WriteRawAsync (Char[], Int32, Int32) | Asynchronicznie zapisuje nieprzetworzone znaczniki ręcznie z buforu znaków. | • |
| WriteRawAsync (String) | Asynchronicznie zapisuje nieprzetworzone znaczniki ręcznie z ciągu. | • |
| WriteStartAttribute (String) | Zapisuje początek atrybutu o określonej nazwie lokalnej. | • |
| WriteStartAttribute (String, String) | Zapisuje początek atrybutu z określoną lokalną nazwą i identyfikatorem URI przestrzeni nazw. | • |
| WriteStartAttribute (String, String, String) | Po zastąpieniu w klasie pochodnej zapisuje początek atrybutu z określonym prefiksem, nazwą lokalną i identyfikatorem URI przestrzeni nazw. | • |
| WriteStartAttribute (String, XmlDictionaryString, XmlDictionaryString) | Zapisuje początek atrybutu z określonym prefiksem, nazwą lokalną i identyfikatorem URI przestrzeni nazw. | • |
| WriteStartAttribute (XmlDictionaryString, XmlDictionaryString) | Zapisuje początek atrybutu o określonej nazwie lokalnej i identyfikatorze URI przestrzeni nazw. |  |
| WriteStartAttributeAsync (String, String, String) | Asynchronicznie zapisuje początek atrybutu z określonym prefiksem, nazwą lokalną i identyfikatorem URI przestrzeni nazw. | • |
| WriteStartDocument () | Podczas zastępowania w klasie pochodnej zapisuje deklarację XML z wersją "1.0". | • |
| WriteStartDocument (Boolean) | Gdy zastąpisz klasę pochodną, zapisuje deklarację XML z wersją "1.0" i atrybutem autonomicznym. | • |
| WriteStartDocumentAsync () | Asynchronicznie zapisuje deklarację XML w wersji "1.0". | • |
| WriteStartDocumentAsync (Boolean) | Asynchronicznie zapisuje deklarację XML z wersją "1.0" i atrybutem autonomicznym. | • |
| WriteStartElement (String) | Po zastąpieniu w klasie pochodnej zapisuje tag startowy o określonej nazwie lokalnej. | • |
| WriteStartElement (String, String) | Po zastąpieniu w klasie pochodnej zapisuje określony tag startowy i kojarzy go z daną przestrzenią nazw. | • |
| WriteStartElement (String, String, String) | Po zastąpieniu w klasie pochodnej zapisuje określony tag startowy i kojarzy go z daną przestrzenią nazw i prefiksem. | • |
| WriteStartElement (String, XmlDictionaryString, XmlDictionaryString) | Zapisuje określony tag startowy i kojarzy go z daną przestrzenią nazw i prefiksem. |  |
| WriteStartElement (XmlDictionaryString, XmlDictionaryString) | Zapisuje określony tag startowy i kojarzy go z daną przestrzenią nazw. |  |
| WriteStartElementAsync (String, String, String) | Asynchronicznie zapisuje określony tag startowy i kojarzy go z daną przestrzenią nazw i prefiksem. | • |
| WriteString (String) | Podczas zastępowania w klasie pochodnej zapisuje daną zawartość tekstową. | • |
| WriteString (XmlDictionaryString) | Zapisuje daną zawartość tekstowa. |  |
| WriteStringAsync (String) | Asynchronicznie zapisuje daną zawartość tekstowa. | • |
| WriteSurrogateCharEntity (Char, Char) | Po zastąpieniu w klasie pochodnej generuje i zapisuje encję znaku zastępczego dla pary znaków zastępczych. | • |
| WriteSurrogateCharEntityAsync (Char, Char) | Asynchronicznie generuje i zapisuje encję znaku zastępczego dla pary znaków zastępczych. | • |
| WriteTextNode (XmlDictionaryReader, Boolean) | Zapisuje węzeł tekstowy, na który XmlDictionaryReader jest obecnie ustawiony. |  |
| WriteValue (Boolean) | Zapisuje wartość Boolean. | • |
| WriteValue (DateTime) | Zapisuje wartość DateTime. | • |
| WriteValue (DateTimeOffset) | Zapisuje wartość DateTimeOffset. | • |
| WriteValue (Decimal) | Zapisuje wartość Decimal. | • |
| WriteValue (Double) | Zapisuje wartość Double. | • |
| WriteValue (Guid) | Zapisuje wartość Guid. |  |
| WriteValue (Int32) | Zapisuje wartość Int32. | • |
| WriteValue (Int64) | Zapisuje wartość Int64. | • |
| WriteValue (IStreamProvider) | Zapisuje wartość z elementu IStreamProvider. |  |
| WriteValue (Object) | Zapisuje wartość obiektu. | • |
| WriteValue (Single) | Zapisuje liczbę zmiennoprzecinkową o pojedynczej precyzji. | • |
| WriteValue (String) | String Zapisuje wartość. |  |
| WriteValue (TimeSpan) | Zapisuje wartość TimeSpan. | • |
| WriteValue (UniqueId) | Zapisuje wartość unikatowego identyfikatora. |  |
| WriteValue (XmlDictionaryString) | Zapisuje wartość XmlDictionaryString. |  |
| WriteValueAsync (IStreamProvider) | Asynchronicznie zapisuje wartość z elementu IStreamProvider. |  |
| WriteWhitespace (String) | Po przesłonięciu w klasie pochodnej zapisuje dane białe znaki. | • |
| WriteWhitespaceAsync (String) | Asynchronicznie zapisuje dane białe znaki. | • |
| WriteXmlAttribute (String, String) | Zapisuje standardowy atrybut XML w bieżącym węźle. |  |
| WriteXmlAttribute (XmlDictionaryString, XmlDictionaryString) | Zapisuje atrybut XML w bieżącym węźle. |  |
| WriteXmlnsAttribute (String, String) | Zapisuje atrybut deklaracji przestrzeni nazw. |  |
| WriteXmlnsAttribute (String, XmlDictionaryString) | Zapisuje atrybut deklaracji przestrzeni nazw. |  |

## Klasa XmlDictionaryReader

Analogiczna klasa XmlDictionaryReader jest klasą abstrakcyjną, pochodzącą od klasy XmlReader, która ułatwia deserializację w Windows Communication Foundation (WCF). Większość właściwości (tab. 15) i metod (tab. 16) dziedziczy po klasie XmlReader (co pokazano w ostatniej kolumnie).

Tab. . Właściwości klasy XmlDictionaryReader

|  |  |  |  |
| --- | --- | --- | --- |
| Właściwość | Znaczenie | ^ | |
| AttributeCount | Po zastąpieniu w klasie pochodnej podaje liczbę atrybutów w bieżącym węźle. | | • |
| BaseURI | Po zastąpieniu w klasie pochodnej podaje podstawowy identyfikator URI bieżącego węzła. | | • |
| CanCanonicalize | Ta właściwość zawsze zwraca wartość false. Klasy pochodne mogą ją przesłonić, aby zwracać true , jeśli obsługują kanonizację. | |  |
| CanReadBinaryContent | Podaje wartość wskazującą, czy XmlReader implementuje metody odczytu zawartości binarnej. | | • |
| CanReadValueChunk | Podaje wartość wskazującą, czy XmlReader implementuje metodę ReadValueChunk(Char[], Int32, Int32) . | | • |
| CanResolveEntity | Podaje wartość wskazującą, czy ten czytelnik może analizować i rozpoznawać jednostki. | | • |
| Depth | Po zastąpieniu klasy pochodnej podaje głębokość bieżącego węzła w dokumencie XML. | | • |
| EOF | Po zastąpieniu w klasie pochodnej podaje wartość wskazującą, czy czytnik jest umieszczony na końcu strumienia. | | • |
| HasAttributes | Podaje wartość wskazującą, czy bieżący węzeł ma jakiekolwiek atrybuty. | | • |
| HasValue | Po zastąpieniu w klasie pochodnej podaje wartość wskazującą, czy bieżący węzeł może mieć wartość Value. | | • |
| IsDefault | W przypadku zastąpienia w klasie pochodnej podaje wartość wskazującą, czy bieżący węzeł jest atrybutem wygenerowanym z wartości domyślnej zdefiniowanej w DTD lub schemacie. | | • |
| IsEmptyElement | Po zastąpieniu w klasie pochodnej podaje wartość wskazującą, czy bieżący węzeł jest pustym elementem (na przykład <MyElement/>). | | • |
| Item[Int32] | Po zastąpieniu w klasie pochodnej podaje wartość atrybutu z określonym indeksem. | | • |
| Item[String, String] | Po zastąpieniu w klasie pochodnej podaje wartość atrybutu o określonej wartości LocalName i NamespaceURI. | | • |
| Item[String] | Po zastąpieniu w klasie pochodnej podaje wartość atrybutu o określonej wartości Name. | | • |
| LocalName | Po zastąpieniu w klasie pochodnej podaje lokalną nazwę bieżącego węzła. | | • |
| Name | Po zastąpieniu w klasie pochodnej podaje kwalifikowaną nazwę bieżącego węzła. | | • |
| NamespaceURI | Po zastąpieniu w klasie pochodnej podaje identyfikator URI przestrzeni nazw (zgodnie ze specyfikacją przestrzeni nazw W3C) węzła, na którym znajduje się czytnik. | | • |
| NameTable | Po przesłonięciu w klasie pochodnej XmlNameTable zostanie skojarzona z tą implementacją. | | • |
| NodeType | Po zastąpieniu w klasie pochodnej podaje typ bieżącego węzła. | | • |
| Prefix | Po zastąpieniu w klasie pochodnej podaje prefiks przestrzeni nazw skojarzony z bieżącym węzłem. | | • |
| Quotas | Podaje wartości przydziału, które mają zastosowanie do bieżącego wystąpienia tej klasy. | |  |
| QuoteChar | Po zastąpieniu w klasie pochodnej podaje znak cudzysłowu używany do ujęcia wartości węzła atrybutu. | | • |
| ReadState | Po przesłonięciu w klasie pochodnej podaje stan czytnika. | | • |
| SchemaInfo | Podaje informacje o schemacie, które zostały przypisane do bieżącego węzła w wyniku weryfikacji schematu. | | • |
| Settings | Podaje obiekt użyty do utworzenia XmlReaderSettings tego wystąpienia klasy XmlReader. | | • |
| Value | Po zastąpieniu w klasie pochodnej podaje wartość tekstową bieżącego węzła. | | • |
| ValueType | Podaje typ środowiska uruchomieniowego języka wspólnego (CLR) dla bieżącego węzła. | | • |
| XmlLang | Po zastąpieniu w klasie pochodnej podaje bieżący zakres xml:lang. | | • |
| XmlSpace | Po zastąpieniu w klasie pochodnej podaje bieżący zakres xml:space. | | • |

Tab. . Metody klasy XmlDictionaryReader

|  |  |  |
| --- | --- | --- |
| Metoda | Znaczenie | ^ |
| Close () | Gdy przesłonięta w klasie pochodnej, zamyka ten strumień i strumień bazowy. | • |
| Close() | Po przesłonięciu w klasie pochodnej zmienia stan ReadState na Closed. | • |
| CreateBinaryReader(Byte[], Int32, Int32, IXmlDictionary, XmlDictionaryReaderQuotas) | Tworzy wystąpienie klasy XmlDictionaryReader, które może odczytywać format binarny XML z tablicy bajtów. |  |
| CreateBinaryReader(Byte[], Int32, Int32, IXmlDictionary, XmlDictionaryReaderQuotas, XmlBinaryReaderSession) | Tworzy wystąpienie klasy XmlDictionaryReader, które może odczytywać format binarny XML z tablicy bajtów. |  |
| CreateBinaryReader(Byte[], Int32, Int32, IXmlDictionary, XmlDictionaryReaderQuotas, XmlBinaryReaderSession, OnXmlDictionaryReaderClose) | Tworzy wystąpienie klasy XmlDictionaryReader, które może odczytywać format binarny XML z tablicy bajtów. |  |
| CreateBinaryReader(Byte[], Int32, Int32, XmlDictionaryReaderQuotas) | Tworzy wystąpienie klasy XmlDictionaryReader, które może odczytywać format binarny XML z tablicy bajtów. |  |
| CreateBinaryReader(Byte[], XmlDictionaryReaderQuotas) | Tworzy wystąpienie klasy XmlDictionaryReader, które może odczytywać format binarny XML z tablicy bajtów. |  |
| CreateBinaryReader(Stream, IXmlDictionary, XmlDictionaryReaderQuotas) | Tworzy wystąpienie klasy XmlDictionaryReader, które może odczytywać format binarny XML ze strumienia. |  |
| CreateBinaryReader(Stream, IXmlDictionary, XmlDictionaryReaderQuotas, XmlBinaryReaderSession) | Tworzy wystąpienie klasy XmlDictionaryReader, które może odczytywać format binarny XML ze strumienia. |  |
| CreateBinaryReader(Stream, IXmlDictionary, XmlDictionaryReaderQuotas, XmlBinaryReaderSession, OnXmlDictionaryReaderClose) | Tworzy wystąpienie klasy XmlDictionaryReader, które może odczytywać format binarny XML ze strumienia. |  |
| CreateBinaryReader(Stream, XmlDictionaryReaderQuotas) | Tworzy wystąpienie klasy XmlDictionaryReader, które może odczytywać format binarny XML ze strumienia. |  |
| CreateDictionaryReader(XmlReader) | Tworzy wystąpienie klasy XmlDictionaryReader na podstawie istniejącego elementu XmlReader. |  |
| CreateMtomReader(Byte[], Int32, Int32, Encoding, XmlDictionaryReaderQuotas) | Tworzy wystąpienie klasy XmlDictionaryReader, które odczytuje kod XML w formacie MTOM. |  |
| CreateMtomReader(Byte[], Int32, Int32, Encoding[], String, XmlDictionaryReaderQuotas) | Tworzy wystąpienie klasy XmlDictionaryReader, które odczytuje kod XML w formacie MTOM. |  |
| CreateMtomReader(Byte[], Int32, Int32, Encoding[], String, XmlDictionaryReaderQuotas, Int32, OnXmlDictionaryReaderClose) | Tworzy wystąpienie klasy XmlDictionaryReader, które odczytuje kod XML w formacie MTOM. |  |
| CreateMtomReader(Byte[], Int32, Int32, Encoding[], XmlDictionaryReaderQuotas) | Tworzy wystąpienie klasy XmlDictionaryReader, które odczytuje kod XML w formacie MTOM. |  |
| CreateMtomReader(Stream, Encoding, XmlDictionaryReaderQuotas) | Tworzy wystąpienie klasy XmlDictionaryReader, które odczytuje kod XML w formacie MTOM. |  |
| CreateMtomReader(Stream, Encoding[], String, XmlDictionaryReaderQuotas) | Tworzy wystąpienie klasy XmlDictionaryReader, które odczytuje kod XML w formacie MTOM. |  |
| CreateMtomReader(Stream, Encoding[], String, XmlDictionaryReaderQuotas, Int32, OnXmlDictionaryReaderClose) | Tworzy wystąpienie klasy XmlDictionaryReader, które odczytuje kod XML w formacie MTOM. |  |
| CreateMtomReader(Stream, Encoding[], XmlDictionaryReaderQuotas) | Tworzy wystąpienie klasy XmlDictionaryReader, które odczytuje kod XML w formacie MTOM. |  |
| CreateTextReader(Byte[], Int32, Int32, Encoding, XmlDictionaryReaderQuotas, OnXmlDictionaryReaderClose) | Tworzy wystąpienie klasy XmlDictionaryReader, które może odczytywać format tekstowy XML z tablicy bajtów. |  |
| CreateTextReader(Byte[], Int32, Int32, XmlDictionaryReaderQuotas) | Tworzy wystąpienie klasy XmlDictionaryReader, które może odczytywać format tekstowy XML z tablicy bajtów. |  |
| CreateTextReader(Byte[], XmlDictionaryReaderQuotas) | Tworzy wystąpienie klasy XmlDictionaryReader, które może odczytywać format tekstowy XML z tablicy bajtów. |  |
| CreateTextReader(Stream, Encoding, XmlDictionaryReaderQuotas, OnXmlDictionaryReaderClose) | Tworzy wystąpienie klasy XmlDictionaryReader, które może odczytywać format tekstowy XML ze strumienia. |  |
| CreateTextReader(Stream, XmlDictionaryReaderQuotas) | Tworzy wystąpienie klasy XmlDictionaryReader, które może odczytywać format tekstowy XML ze strumienia. |  |
| Dispose() | Zwalnia wszystkie zasoby używane przez bieżące wystąpienie klasy XmlReader. | • |
| Dispose(Boolean) | Zwalnia zasoby niezarządzane używane przez element XmlReader i opcjonalnie zwalnia zasoby zarządzane. | • |
| EndCanonicalization() | Ta metoda nie została jeszcze zaimplementowana. |  |
| GetAttribute(Int32) | Po zastąpieniu w klasie pochodnej pobiera wartość atrybutu z określonym indeksem. | • |
| GetAttribute(String) | Po zastąpieniu w klasie pochodnej pobiera wartość atrybutu o określonej wartości Name. | • |
| GetAttribute(String, String) | Po zastąpieniu w klasie pochodnej pobiera wartość atrybutu o określonej wartości LocalName i NamespaceURI. | • |
| GetAttribute(XmlDictionaryString, XmlDictionaryString) | Po zastąpieniu w klasie pochodnej pobiera wartość atrybutu. |  |
| GetNonAtomizedNames(String, String) | Pobiera nazwy nieatomizowane. |  |
| GetValueAsync() | Asynchronicznie pobiera wartość bieżącego węzła. | • |
| IndexOfLocalName(String[], String) | Pobiera indeks nazwy lokalnej bieżącego węzła w tablicy nazw. |  |
| IndexOfLocalName(XmlDictionaryString[], XmlDictionaryString) | Pobiera indeks nazwy lokalnej bieżącego węzła w tablicy nazw. |  |
| IsLocalName(String) | Sprawdza, czy parametr localName jest lokalną nazwą bieżącego węzła. |  |
| IsLocalName(XmlDictionaryString) | Sprawdza, czy parametr localName jest lokalną nazwą bieżącego węzła. |  |
| IsNamespaceUri(String) | Sprawdza, czy parametr namespaceUri jest przestrzenią nazw bieżącego węzła. |  |
| IsNamespaceUri(XmlDictionaryString) | Sprawdza, czy parametr namespaceUri jest przestrzenią nazw bieżącego węzła. |  |
| IsStartArray(Type) | Sprawdza, czy czytnik jest umieszczony na początku tablicy. Ta klasa zwraca klasę false, ale klasy pochodne, które mają pojęcie tablic, mogą zwracać wartość true. |  |
| IsStartElement() | Wywołuje metodę MoveToContent() i sprawdza, czy bieżący węzeł zawartości jest tagiem początkowym lub pustym tagiem elementu. | • |
| IsStartElement(String) | Wywołuje metodę MoveToContent() i sprawdza, czy bieżący węzeł zawartości jest tagiem początkowym lub pustym tagiem elementu, a Name właściwość znalezionego elementu jest zgodna z danym argumentem. | • |
| IsStartElement(String, String) | Wywołuje metodę MoveToContent() i sprawdza, czy bieżący węzeł zawartości jest tagiem początkowym lub pustym tagiem elementu, a LocalName właściwości i NamespaceURI elementu znalezionego są zgodne z podanymi ciągami. | • |
| IsStartElement(XmlDictionaryString, XmlDictionaryString) | Sprawdza, czy pierwszy tag jest tagiem początkowym, czy pustym tagiem, oraz czy identyfikator URI lokalnej nazwy i przestrzeni nazw jest zgodny z identyfikatorami URI bieżącego węzła. |  |
| LookupNamespace(String) | Po przesłonięciu w klasie pochodnej rozpoznaje prefiks przestrzeni nazw w zakresie bieżącego elementu. | • |
| MoveToAttribute(Int32) | Po przesłonięciu w klasie pochodnej przechodzi do atrybutu z określonym indeksem. | • |
| MoveToAttribute(String, String) | Po przesłonięciu w klasie pochodnej przechodzi do atrybutu o określonej wartości LocalName i NamespaceURI. | • |
| MoveToContent() | Sprawdza, czy bieżący węzeł jest zawartością węzła (czymś innym niż odstęp, CDATA, Element, EndElement, EntityReference lub EndEntity). Jeśli węzeł nie jest węzłem zawartości, czytnik pomija następny węzeł zawartości lub koniec pliku. Pomija on węzły następującego typu: ProcessingInstruction, DocumentTypeComment, Whitespace i SignificantWhitespace. | • |
| MoveToContentAsync() | Asynchronicznie sprawdza, czy bieżący węzeł jest węzłem zawartości. Jeśli węzeł nie jest węzłem zawartości, czytnik pomija następny węzeł zawartości lub koniec pliku. | • |
| MoveToElement() | Po przesłonięciu w klasie pochodnej przechodzi do elementu zawierającego bieżący węzeł atrybutu. | • |
| MoveToFirstAttribute() | Po przesłonięciu w klasie pochodnej zostanie przeniesiony do pierwszego atrybutu. | • |
| MoveToNextAttribute() | Po przesłonięciu w klasie pochodnej przechodzi do następnego atrybutu. | • |
| MoveToStartElement() | Sprawdza, czy bieżący węzeł zawartości jest elementem początkowym, czy pustym elementem. |  |
| MoveToStartElement(String) | Sprawdza, czy bieżący węzeł zawartości jest elementem początkowym, czy pustym elementem, a właściwość Name elementu jest zgodna z danym argumentem. |  |
| MoveToStartElement(String, String) | Sprawdza, czy bieżący węzeł zawartości jest elementem początkowym, czy pustym elementem, a właściwości LocalName i NamespaceURI elementu są zgodne z podanymi argumentami. |  |
| MoveToStartElement(XmlDictionaryString, XmlDictionaryString) | Sprawdza, czy bieżący węzeł zawartości jest elementem początkowym, czy pustym elementem, a właściwości localName i NamespaceURI elementu są zgodne z danym argumentem. |  |
| Read() | Po przesłonięciu w klasie pochodnej odczytuje następny węzeł ze strumienia. | • |
| ReadArray(String, String, Boolean[], Int32, Int32) | Odczytuje powtarzające się wystąpienia węzłów Boolean do tablicy. |  |
| ReadArray(String, String, DateTime[], Int32, Int32) | Odczytuje powtarzające się wystąpienia węzłów DateTime do tablicy. |  |
| ReadArray(String, String, Decimal[], Int32, Int32) | Odczytuje powtarzające się wystąpienia węzłów Decimal do tablicy. |  |
| ReadArray(String, String, Double[], Int32, Int32) | Odczytuje powtarzające się wystąpienia węzłów Double do tablicy. |  |
| ReadArray(String, String, Guid[], Int32, Int32) | Odczytuje powtarzające się wystąpienia węzłów Guid do tablicy. |  |
| ReadArray(String, String, Int16[], Int32, Int32) | Odczytuje powtarzające się wystąpienia liczb całkowitych short do tablicy. |  |
| ReadArray(String, String, Int32[], Int32, Int32) | Odczytuje powtarzające się wystąpienia liczb całkowitych do tablicy. |  |
| ReadArray(String, String, Int64[], Int32, Int32) | Odczytuje powtarzające się wystąpienia liczb całkowitych long do tablicy. |  |
| ReadArray(String, String, Single[], Int32, Int32) | Odczytuje powtarzające się wystąpienia float liczb do tablicy. |  |
| ReadArray(String, String, TimeSpan[], Int32, Int32) | Odczytuje powtarzające się wystąpienia węzłów TimeSpan do tablicy. |  |
| ReadArray(XmlDictionaryString, XmlDictionaryString, Boolean[], Int32, Int32) | Odczytuje powtarzające się wystąpienia węzłów Boolean do tablicy. |  |
| ReadArray(XmlDictionaryString, XmlDictionaryString, DateTime[], Int32, Int32) | Odczytuje powtarzające się wystąpienia węzłów DateTime do tablicy. |  |
| ReadArray(XmlDictionaryString, XmlDictionaryString, Decimal[], Int32, Int32) | Odczytuje powtarzające się wystąpienia węzłów Decimal do tablicy. |  |
| ReadArray(XmlDictionaryString, XmlDictionaryString, Double[], Int32, Int32) | Odczytuje powtarzające się wystąpienia typu węzłów Double do tablicy. |  |
| ReadArray(XmlDictionaryString, XmlDictionaryString, Guid[], Int32, Int32) | Odczytuje powtarzające się wystąpienia węzłów Guid do tablicy. |  |
| ReadArray(XmlDictionaryString, XmlDictionaryString, Int16[], Int32, Int32) | Odczytuje powtarzające się wystąpienia liczb całkowitych short do tablicy. |  |
| ReadArray(XmlDictionaryString, XmlDictionaryString, Int32[], Int32, Int32) | Odczytuje powtarzające się wystąpienia liczb całkowitych do tablicy. |  |
| ReadArray(XmlDictionaryString, XmlDictionaryString, Int64[], Int32, Int32) | Odczytuje powtarzające się wystąpienia liczb całkowitych long do tablicy. |  |
| ReadArray(XmlDictionaryString, XmlDictionaryString, Single[], Int32, Int32) | Odczytuje powtarzające się wystąpienia float liczb do tablicy. |  |
| ReadArray(XmlDictionaryString, XmlDictionaryString, TimeSpan[], Int32, Int32) | Odczytuje powtarzające się wystąpienia węzłów TimeSpan do tablicy. |  |
| ReadAsync() | Asynchronicznie odczytuje następny węzeł ze strumienia. | • |
| ReadAttributeValue() | Po zastąpieniu w klasie pochodnej analizuje wartość atrybutu na co najmniej jeden węzeł Text, EntityReference lub EndEntity. | • |
| ReadBooleanArray(String, String) | Odczytuje powtarzające się wystąpienia węzłów Boolean do tablicy. |  |
| ReadBooleanArray(XmlDictionaryString, XmlDictionaryString) | Odczytuje powtarzające się wystąpienia węzłów Boolean do tablicy. |  |
| ReadContentAs(Type, IXmlNamespaceResolver) | Konwertuje zawartość węzła na określony typ. |  |
| ReadContentAsAsync(Type, IXmlNamespaceResolver) | Asynchronicznie odczytuje zawartość jako obiekt określonego typu. | • |
| ReadContentAsBase64() | Odczytuje zawartość i zwraca dekodowane bajty binarne Base64. |  |
| ReadContentAsBase64(Byte[], Int32, Int32) | Odczytuje zawartość i zwraca dekodowane bajty binarne Base64 do tablicy. | • |
| ReadContentAsBase64Async(Byte[], Int32, Int32) | Asynchronicznie odczytuje zawartość i zwraca dekodowane bajty binarne Base64. | • |
| ReadContentAsBinHex() | Odczytuje zawartość i zwraca BinHex zdekodowane bajty binarne. |  |
| ReadContentAsBinHex(Byte[], Int32, Int32) | Odczytuje zawartość i zwraca BinHex zdekodowane bajty binarne do tablicy | • |
| ReadContentAsBinHex(Int32) | Odczytuje zawartość i zwraca BinHex zdekodowane bajty binarne. |  |
| ReadContentAsBinHexAsync(Byte[], Int32, Int32) | Asynchronicznie odczytuje zawartość BinHex i zwraca zdekodowane bajty binarne do tablicy. | • |
| ReadContentAsBoolean() | Odczytuje zawartość tekstową w bieżącym położeniu jako Boolean. | • |
| ReadContentAsChars(Char[], Int32, Int32) | Odczytuje zawartość do tablicy char. |  |
| ReadContentAsDateTime() | Odczytuje zawartość tekstową w bieżącej pozycji jako obiekt DateTime. | • |
| ReadContentAsDateTimeOffset() | Odczytuje zawartość tekstową w bieżącej pozycji jako obiekt DateTimeOffset. | • |
| ReadContentAsDecimal() | Konwertuje zawartość węzła na decimal. |  |
| ReadContentAsDouble() | Odczytuje zawartość tekstową w bieżącej pozycji jako liczbę zmiennoprzecinkową o podwójnej precyzji. | • |
| ReadContentAsFloat() | Konwertuje zawartość węzła na float. |  |
| ReadContentAsGuid() | Konwertuje zawartość węzła na Guid. |  |
| ReadContentAsInt() | Odczytuje zawartość tekstową na bieżącej pozycji jako liczbę całkowitą 32-bitową ze znakiem. | • |
| ReadContentAsLong() | Odczytuje zawartość tekstową na bieżącej pozycji jako liczbę całkowitą 64-bitową ze znakiem. | • |
| ReadContentAsObject() | Odczytuje zawartość tekstową w bieżącej pozycji jako Object. | • |
| ReadContentAsObjectAsync() | Asynchronicznie odczytuje zawartość tekstową w bieżącej pozycji jako Object. | • |
| ReadContentAsQualifiedName(String, String) | Konwertuje zawartość węzła na kwalifikowaną reprezentację nazw. |  |
| ReadContentAsString() | Konwertuje zawartość węzła na ciąg. |  |
| ReadContentAsString(Int32) | Konwertuje zawartość węzła na ciąg. |  |
| ReadContentAsString(String[], Int32) | Konwertuje zawartość węzła na ciąg. |  |
| ReadContentAsString(XmlDictionaryString[], Int32) | Konwertuje zawartość węzła na ciąg. |  |
| ReadContentAsStringAsync() | Asynchronicznie odczytuje zawartość tekstową na bieżącej String pozycji jako obiekt. | • |
| ReadContentAsTimeSpan() | Konwertuje zawartość węzła na TimeSpan. |  |
| ReadContentAsUniqueId() | Konwertuje zawartość węzła na unikatowy identyfikator. |  |
| ReadDateTimeArray(String, String) | Konwertuje zawartość węzła na tablicę DateTime. |  |
| ReadDateTimeArray(XmlDictionaryString, XmlDictionaryString) | Konwertuje zawartość węzła na tablicę DateTime. |  |
| ReadDecimalArray(String, String) | Konwertuje zawartość węzła na tablicę Decimal. |  |
| ReadDecimalArray(XmlDictionaryString, XmlDictionaryString) | Konwertuje zawartość węzła na tablicę Decimal. |  |
| ReadDoubleArray(String, String) | Konwertuje zawartość węzła na tablicę Double. |  |
| ReadDoubleArray(XmlDictionaryString, XmlDictionaryString) | Konwertuje zawartość węzła na tablicę Double. |  |
| ReadElementContentAs(Type, IXmlNamespaceResolver) | Odczytuje zawartość elementu jako żądany typ. | • |
| ReadElementContentAs(Type, IXmlNamespaceResolver, String, String) | Sprawdza, czy określona lokalna nazwa i identyfikator URI przestrzeni nazw są zgodne z bieżącym elementem, a następnie odczytuje zawartość elementu jako żądany typ. | • |
| ReadElementContentAsAsync(Type, IXmlNamespaceResolver) | Asynchronicznie odczytuje zawartość elementu jako żądany typ. | • |
| ReadElementContentAsBase64() | Konwertuje zawartość węzła na tablicę bajtów Base64. |  |
| ReadElementContentAsBase64(Byte[], Int32, Int32) | Odczytuje element i dekoduje zawartość Base64. | • |
| ReadElementContentAsBase64Async(Byte[], Int32, Int32) | Asynchronicznie odczytuje element i dekoduje zawartość Base64. | • |
| ReadElementContentAsBinHex() | Konwertuje zawartość węzła na tablicę bajtów BinHex. |  |
| ReadElementContentAsBinHex(Byte[], Int32, Int32) | Odczytuje element i dekoduje zawartość BinHex. | • |
| ReadElementContentAsBinHexAsync(Byte[], Int32, Int32) | Asynchronicznie odczytuje element i dekoduje B zawartość inHex. | • |
| ReadElementContentAsBoolean() | Konwertuje zawartość elementu na element Boolean. |  |
| ReadElementContentAsBoolean(String, String) | Sprawdza, czy określona lokalna nazwa i identyfikator URI przestrzeni nazw są zgodne z bieżącym elementem, a następnie odczytuje bieżący element i zwraca zawartość jako Boolean obiekt. | • |
| ReadElementContentAsDateTime() | Konwertuje zawartość elementu na element DateTime. |  |
| ReadElementContentAsDateTime(String, String) | Sprawdza, czy określona nazwa lokalna i identyfikator URI przestrzeni nazw jest zgodna z bieżącym elementem DateTime, a następnie odczytuje bieżący element i zwraca zawartość jako obiekt. | • |
| ReadElementContentAsDecimal() | Konwertuje zawartość elementu na element Decimal. |  |
| ReadElementContentAsDecimal(String, String) | Sprawdza, czy określona nazwa lokalna i identyfikator URI przestrzeni nazw jest zgodna z bieżącym elementem Decimal, a następnie odczytuje bieżący element i zwraca zawartość jako obiekt. | • |
| ReadElementContentAsDouble() | Konwertuje zawartość elementu na element Double. |  |
| ReadElementContentAsDouble(String, String) | Sprawdza, czy określona nazwa lokalna i identyfikator URI przestrzeni nazw jest zgodna z bieżącym elementem, a następnie odczytuje bieżący element i zwraca zawartość jako liczbę zmiennoprzecinkową o podwójnej precyzji. | • |
| ReadElementContentAsFloat() | Konwertuje zawartość elementu na liczbę zmiennoprzecinkową (Single). |  |
| ReadElementContentAsFloat(String, String) | Sprawdza, czy określona lokalna nazwa i identyfikator URI przestrzeni nazw jest zgodna z bieżącym elementem, a następnie odczytuje bieżący element i zwraca zawartość jako liczbę zmiennoprzecinkową o pojedynczej precyzji. | • |
| ReadElementContentAsGuid() | Konwertuje zawartość elementu na element Guid. |  |
| ReadElementContentAsInt() | Konwertuje zawartość elementu na liczbę całkowitą (Int32). |  |
| ReadElementContentAsInt(String, String) | Sprawdza, czy określona lokalna nazwa i identyfikator URI przestrzeni nazw jest zgodna z bieżącym elementem, a następnie odczytuje bieżący element i zwraca zawartość jako 32-bitową liczbę całkowitą ze znakiem. | • |
| ReadElementContentAsLong() | Konwertuje zawartość elementu na długą liczbę całkowitą (Int64). |  |
| ReadElementContentAsLong(String, String) | Sprawdza, czy określona lokalna nazwa i identyfikator URI przestrzeni nazw jest zgodna z bieżącym elementem, a następnie odczytuje bieżący element i zwraca zawartość jako 64-bitową liczbę całkowitą ze znakiem. | • |
| ReadElementContentAsObject() | Odczytuje bieżący element i zwraca zawartość jako element Object. | • |
| ReadElementContentAsObject(String, String) | Sprawdza, czy określona lokalna nazwa i identyfikator URI przestrzeni nazw są zgodne z bieżącym elementem, a następnie odczytuje bieżący element i zwraca zawartość jako Object. | • |
| ReadElementContentAsObjectAsync() | Asynchronicznie odczytuje bieżący element i zwraca zawartość jako element Object. | • |
| ReadElementContentAsString() | Konwertuje zawartość elementu na element String. |  |
| ReadElementContentAsString(String, String) | Sprawdza, czy określona lokalna nazwa i identyfikator URI przestrzeni nazw są zgodne z bieżącym elementem, a następnie odczytuje bieżący element i zwraca zawartość jako String obiekt. | • |
| ReadElementContentAsStringAsync() | Asynchronicznie odczytuje bieżący element i zwraca zawartość jako String obiekt. | • |
| ReadElementContentAsTimeSpan() | Konwertuje zawartość elementu na element TimeSpan. |  |
| ReadElementContentAsUniqueId() | Konwertuje zawartość elementu na unikatowy identyfikator. |  |
| ReadElementString() | Odczytuje element tylko tekst. Zalecamy jednak użycie ReadElementContentAsString() metody, ponieważ zapewnia ona prostszy sposób obsługi tej operacji. | • |
| ReadElementString(String) | Sprawdza, czy Name właściwość znalezionego elementu jest zgodna z danym ciągiem przed odczytaniem elementu tylko do tekstu. Zalecane jest jednak użycie metody ReadElementContentAsString(), ponieważ zapewnia ona prostszy sposób obsługi tej operacji. | • |
| ReadElementString(String, String) | Sprawdza, czy Właściwość localName i NamespaceURI znalezionego elementu są zgodne z podanymi ciągami przed odczytaniem elementu tylko do tekstu. Zalecane jest jednak użycie metody ReadElementContentAsString(String, String), ponieważ zapewnia ona prostszy sposób obsługi tej operacji. | • |
| ReadEndElement() | Sprawdza, czy bieżący węzeł zawartości jest tagiem końcowym i przenosi czytnik do następnego węzła. | • |
| ReadFullStartElement() | Sprawdza, czy bieżący węzeł jest elementem i przenosi czytnik do następnego węzła. |  |
| ReadFullStartElement(String) | Sprawdza, czy bieżący węzeł jest elementem z danym parametrem name i przenosi czytnik do następnego węzła. |  |
| ReadFullStartElement(String, String) | Sprawdza, czy bieżący węzeł jest elementem z daną wartością localName i namespaceUri przenosi czytnik do następnego węzła. |  |
| ReadFullStartElement(XmlDictionaryString, XmlDictionaryString) | Sprawdza, czy bieżący węzeł jest elementem z daną wartością localName i namespaceUri przenosi czytnik do następnego węzła. |  |
| ReadGuidArray(String, String) | Odczytuje zawartość serii węzłów z daną localName wartością i namespaceUri do tablicy Guid. |  |
| ReadGuidArray(XmlDictionaryString, XmlDictionaryString) | Odczytuje zawartość serii węzłów z daną localName wartością i namespaceUri do tablicy Guid. |  |
| ReadInnerXml() | Po zastąpieniu w klasie pochodnej odczytuje całą zawartość, w tym znaczniki, jako łańcuch znaków. | • |
| ReadInnerXmlAsync() | Asynchronicznie odczytuje całą zawartość, w tym znaczniki, jako łańcuch znaków. | • |
| ReadInt16Array(String, String) | Odczytuje zawartość serii węzłów z daną wartością localName i namespaceUri do tablicy liczb całkowitych short (Int16). |  |
| ReadInt16Array(XmlDictionaryString, XmlDictionaryString) | Odczytuje zawartość serii węzłów z daną wartością localName i namespaceUri do tablicy liczb całkowitych short (Int16). |  |
| ReadInt32Array(String, String) | Odczytuje zawartość serii węzłów z daną wartością localName i namespaceUri do tablicy liczb całkowitych (Int32). |  |
| ReadInt32Array(XmlDictionaryString, XmlDictionaryString) | Odczytuje zawartość serii węzłów z daną wartością localName i namespaceUri do tablicy liczb całkowitych (Int32). |  |
| ReadInt64Array(String, String) | Odczytuje zawartość serii węzłów z daną wartością localName i namespaceUri do tablicy liczb całkowitych long (Int64). |  |
| ReadInt64Array(XmlDictionaryString, XmlDictionaryString) | Odczytuje zawartość serii węzłów z daną wartością localName i namespaceUri do tablicy liczb całkowitych long (Int64). |  |
| ReadOuterXml() | Po zastąpieniu w klasie pochodnej odczytuje zawartość, w tym znaczniki reprezentujące ten węzeł i wszystkie jego elementy podrzędne. | • |
| ReadOuterXmlAsync() | Asynchronicznie odczytuje zawartość, w tym znaczniki reprezentujące ten węzeł i wszystkie jego elementy podrzędne. | • |
| ReadSingleArray(String, String) | Odczytuje zawartość serii węzłów z daną wartością localName i namespaceUri do tablicy float liczb (Single). |  |
| ReadSingleArray(XmlDictionaryString, XmlDictionaryString) | Odczytuje zawartość serii węzłów z daną wartością localName i namespaceUri do tablicy float liczb (Single). |  |
| ReadStartElement() | Sprawdza, czy bieżący węzeł jest elementem i przenosi czytnik do następnego węzła. | • |
| ReadStartElement(String) | Sprawdza, czy bieżący węzeł zawartości jest elementem z daną wartością name i przenosi czytnik do następnego węzła. | • |
| ReadStartElement(String, String) | Sprawdza, czy bieżący węzeł zawartości jest elementem z daną wartością localName i namespaceURI przenosi czytnik do następnego węzła. | • |
| ReadStartElement(XmlDictionaryString, XmlDictionaryString) | Sprawdza, czy bieżący węzeł jest elementem z wartością daną localName i namespaceUri przenosi czytnik do następnego węzła. |  |
| ReadString() | Odczytuje zawartość bieżącego węzła do łańcucha znaków. |  |
| ReadString(Int32) | Odczytuje zawartość bieżącego węzła do łańcucha znaków o podanej maksymalnej długości. |  |
| ReadSubtree() | Zwraca nowe wystąpienie klasy XmlReader, które może służyć do odczytywania bieżącego węzła i wszystkich jego elementów podrzędnych. | • |
| ReadTimeSpanArray(String, String) | Odczytuje zawartość serii węzłów z daną wartością localName i namespaceUri do tablicy TimeSpan. |  |
| ReadTimeSpanArray(XmlDictionaryString, XmlDictionaryString) | Odczytuje zawartość serii węzłów z daną wartością localName i namespaceUri do tablicy TimeSpan. |  |
| ReadToDescendant(String) | Przenosi czytnik do następnego elementu potomnego z określoną kwalifikowaną nazwą. | • |
| ReadToDescendant(String, String) | Przenosi czytnik do następnego elementu podrzędnego z określoną lokalną nazwą i identyfikatorem URI przestrzeni nazw. | • |
| ReadToFollowing(String) | Odczytuje dane do momentu znalezienia elementu o określonej kwalifikowanej nazwie. | • |
| ReadToFollowing(String, String) | Odczytuje dane do momentu znalezienia elementu o określonej lokalnej nazwie i identyfikatorze URI przestrzeni nazw. | • |
| ReadToNextSibling(String) | Przenosi czytnik do następnego elementu równorzędnego z określoną kwalifikowaną nazwą. | • |
| ReadToNextSibling(String, String) | Przenosi czytnik do następnego elementu równorzędnego przy użyciu określonej lokalnej nazwy i identyfikatora URI przestrzeni nazw. | • |
| ReadValueAsBase64(Byte[], Int32, Int32) | Nie zaimplementowano. |  |
| ReadValueChunk(Char[], Int32, Int32) | Odczytuje duże strumienie tekstu osadzone w dokumencie XML. | • |
| ReadValueChunkAsync(Char[], Int32, Int32) | Asynchronicznie odczytuje duże strumienie tekstu osadzonego w dokumencie XML. | • |
| ResolveEntity() | Po zastąpieniu w klasie pochodnej rozpoznaje odwołanie do encji dla węzłów EntityReference. | • |
| Skip() | Pomija elementy podrzędne bieżącego węzła. | • |
| SkipAsync() | Asynchronicznie pomija elementy podrzędne bieżącego węzła. | • |
| StartCanonicalization(Stream, Boolean, String[]) | Ta metoda nie została jeszcze zaimplementowana. |  |
| TryGetArrayLength(Int32) | Nie zaimplementowano w tej klasie (zawsze zwraca wartość false). Może zostać zastąpiony w klasach pochodnych. |  |
| TryGetBase64ContentLength(Int32) | Nie zaimplementowano w tej klasie (zawsze zwraca wartość false). Może zostać zastąpiony w klasach pochodnych. |  |
| TryGetLocalNameAsDictionaryString(XmlDictionaryString) | Nie zaimplementowano w tej klasie (zawsze zwraca wartość false). Może zostać zastąpiony w klasach pochodnych. |  |
| TryGetNamespaceUriAsDictionaryString(XmlDictionaryString) | Nie zaimplementowano w tej klasie (zawsze zwraca wartość false). Może zostać zastąpiony w klasach pochodnych. |  |
| TryGetValueAsDictionaryString(XmlDictionaryString) | Nie zaimplementowano w tej klasie (zawsze zwraca wartość false). Może zostać zastąpiony w klasach pochodnych. |  |

# Konstrukcja własnego serializatora

Własnemu serializatorowi nadano nazwę QXmlSerializer (dla odróżnienia od standardowego serializatora).

Ze względu na wielość metod zdecydowano na utworzenie własnego serializatora w kilku plikach:

* QXmlSerializer.Base.cs – zawiera definicje skopiowane ze standardowego serializatora,
* QXmlSerializer.Main.cs – zawiera główne metody serializacji i deserializacji,
* QXmlSerializer.Serialization.cs – zawiera właściwe metody serializacji,
* QXmlSerializer.Deserialization.cs – zawiera właściwe metody deserializacji,
* QXmlSerializer.Helper.cs – zawiera metody pomocnicze.

## Definicje bazowe

Ponieważ znaczna część definicji standardowego serializatora opiera się na konstruktorach, nie można zadeklarować kompletnego interfejsu zapewniającego pełną zgodność własnego serializatora ze standardowym. W zamian za to zdefiniowano bazową część własnego serializatora w osobnym pliku QXmlSerializer.Base.cs. Część ta zawiera definicje zmiennych ze standardowego serializatora, konstruktory (patrz tab. 6), metody Serialize i Deserialize (patrz tab. 8) oraz deklaracje zdarzeń (tab. 9) standardowego serializatora.

Z prywatnych zmiennych standardowego serializatora zachowano trzy, którym nadano dostęp chroniony. Zmienna \_events z konieczności musi być polem, pozostałe zmienne to właściwości.

Tab. . Zmienne przeniesione z klasy XmlSerializer do części QXmlSerializer.Base

|  |  |  |
| --- | --- | --- |
| Metoda | Typ | Znaczenie |
| \_events | XmlDeserializationEvents | Definiuje pola zdarzeń wg tab. 9. |
| DefaultNamespace | string? | Domyślna przestrzeń nazw przekazywana przez konstruktory |
| DefaultNamespaces | XmlSerializerNamespaces | Zawiera dwie domyślne przestrzenie nazw Xml: xsi oraz xsd, wykorzystywane do wyrażania wartości nil, oraz typów danych xml. |

Aby implementacja konstruktorów była możliwa w innej części, wszystkim konstruktorom przypisano wywołanie jednej z dwóch metod inicjacji zdefiniowanych jako partial. Podobnie metody serializacji i deserializacji przypisano do metod partial.

Tab. . Metody cząstkowe zadeklarowane w części QXmlSerializer.Base

|  |  |
| --- | --- |
| Metoda | Znaczenie |
| Init (Type type, XmlAttributeOverrides? overrides, Type[]? extraTypes, XmlRootAttribute? root, string? defaultNamespace, string? location) | Metoda inicjalizacji serializatora. Parametry jak w najbardziej rozbudowanej wersji konstruktora serializatora klasy XmlSerializer (patrz tab. 6) |
| Init (XmlTypeMapping xmlTypeMapping) | Metoda inicjalizacji serializatora. Parametry jak dla jednej z wersji serializatora konstruktora klasy XmlSerializer (patrz tab. 6) |
| SerializeObject (XmlWriter xmlWriter, object? obj, XmlSerializerNamespaces? namespaces, string? encodingStyle, string? id) | Metoda serializacji obiektu. Parametry jak dla najbardziej rozbudowanej metody Serialize standardowego serializatora (patrz tab. 8) |
| object? DeserializeObject (XmlReader xmlReader, string? encodingStyle, XmlDeserializationEvents events) | Metoda deserializacji obiektu. Parametry jak dla najbardziej rozbudowanej metody Deserialize standardowego serializatora (patrz tab. 8) |
| bool CanDeserialize (XmlReader xmlReader) | Metoda sprawdzenia, czy obiekt może być deserializowany. Parametry jak dla metody CanDeserialize standardowego serializatora (patrz tab. 8) |

Deklaracje zdarzeń odwołują się wprost do pola \_events.

## Część główna

Część QXmlSerializer.Main.cs zawiera dodatkowe konstruktory oraz główne metody serializacji i deserializacji.

Dodatkowe konstruktory mają jeden dodatkowy parametr SerializationOptions, który reguluje opcje serializacji i deserializacji. Wszystkie konstruktory zadeklarowane w części bazowej są powtórzone z dodanym parametrem SerializationOptions. Podobnie obie metody Init zostały przekierowane do nowych metod Init z dodanym parametrem SerializationOptions.

Metoda Init z parametrem XmlTypeMapping na razie pozostaje niezaimplementowana.

Metoda Init z parametrem Type (i pozostałymi) jest podstawową metodą inicjacji serializatora. Tworzy ona komponent XmlSerializationInfoMapper, który analizuje typy elementów przeznaczonych do serializacji, przypisuje im nazwy elementów i atrybutów oraz rejestruje je w słowniku KnownTypes. Do analizy jest przekazany typ główny i typy dodatkowe podane jako parametry metody Init. Ponieważ XmlSerializationInfoMapper przy analizie wykorzystuje niektóre opcje serializacji, dlatego opcje te muszą być dostępne przy inicjacji i w konstruktorach. Jeśli inicjacja następuje przez standardowe konstruktory, to wykorzystywane są opcje domyślne.

## Opcje serializacji

Klasa SerializationOptions zawiera opcje serializacji i deserializacji (patrz tab. 19).

Tab. . Opcje serializacji i deserializacji w klasie SerializationOptions

|  |  |  |  |
| --- | --- | --- | --- |
| Metoda | Typ | Domyślnie | Znaczenie |
| IgnoreMissingConstructor | bool | false | Typy do deserializacji muszą mieć konstruktor bez parametrów. Jeśli typ znaleziony podczas skanowania dostępnych typów nie ma publicznego konstruktora bez parametrów, zostanie zgłoszony wyjątek, chyba że ta opcja jest ustawiona. |
| AcceptAllProperties | bool | false | Czy wszystkie właściwości publiczne nieoznaczone żadnym z atrybutów Xml są akceptowane do serializacji. |
| Simple­Properties­AsAttributes | bool | false | Czy właściwości proste nieoznaczone żadnym z atrybutów Xml są akceptowane do serializacji jako atrybuty. |
| AttributeNameCase | Serialization­Case | Unchanged | Czy i jak mają się zmienić nazwy atrybutów XML podczas serializacji. |
| ­ElementNameCase | Serialization­Case | Unchanged | Czy i jak mają się zmienić nazwy elementów XML podczas serializacji. |
| ­EnumNameCase | Serialization­Case | Unchanged | Czy i jak mają się zmienić nazwy wartości wyliczanych podczas serializacji. |
| IgnoreCaseOnEnum | bool | false | Czy ignorować wielkość liter w nazwach wartości wyliczanych.  W wartościach typu Boolean zawsze są ignorowane. |
| PrecedePropertyName­WithElementName | bool | false | Czy nazwa właściwości serializowana jako element XML ma być poprzedzona nazwą serializowanej klasy. Znak kropki ('.') jest używany jako separator. |
| ItemTag | string? | null | Ciąg znaków do oznaczania elementów w kolekcjach. Jeśli nie określono, elementy są serializowane bezpośrednio. |
| Culture | CultureInfo | Invariant­Culture | Kultura używana w liczbach i datach podczas serializacji/deserializacji. Domyślną jest InvariantCulture. |
| IgnoreUnknownElements | bool | false | Określa, że gdy deserializator znajdzie nieznany element XML, przechodzi do elementu zamykającego (lub ignoruje, jeśli jest to pusty element). |
| UseNilValue | bool | false | Określa, że gdy atrybut ma wartość null, to trzeba wypisywać specjalną wartość xsi:nil. |
| FalseString | string | "false" | Łańcuch wypisywany jako wartość False podczas serializacji. |
| TrueString | string | "true" | Łańcuch wypisywany jako wartość True podczas serializacji. |
| DateTimeFormat | string | "yyyy-MM-ddTHH:mm:sszzz" | Format zapisu daty i czasu podczas serializacji. Przy deserializacji wiele formatów jest akceptowalnych. |
| CheckMethod | string | "ShouldSerialize\*" | Nazwa metody, która określa, czy właściwość ma być serializowana. W miejscu ‘\*’ występuje nazwa właściwości. |

## Metody pomocnicze

Metody pomocnicze zostały zdefiniowane jako statyczne, aby mogły być wywoływane zarówno przez sam serializator, jak i pozostałe komponenty.

Tab. . Metody pomocnicze zadeklarowane w części QXmlSerializer.Helper

|  |  |
| --- | --- |
| Metoda | Znaczenie |
| string ChangeCase (string str, SerializationCase nameCase) | Zmiana wielkości liter nazwy w zależności od opcji. |
| string FirstLetterToLower (string text) | Zmiana pierwszej litery łańcucha na małą. |
| string FirstLetterToUpper (string text) | Zmiana pierwszej litery łańcucha na wielką. |
| bool IsFirstLetterLower (string text) | Czy pierwsza litera łańcucha jest mała? |
| bool IsFirstLetterUpper (string text) | Czy pierwsza litera łańcucha jest wielka? |

## Mapowanie typów

Mapowanie typów następuje w klasie XmlSerializationInfoMapper i polega na rekurencyjnym analizowaniu typów elementów, począwszy od typu elementu głównego, poprzez typy pól i właściwości. Ponieważ w takiej strukturze mogłyby się nie pojawić pewne typy elementów, więc brakujące typy można podać w dodatkowej tablicy i one też będą analizowane rekurencyjnie.

W czasie analizy budowana jest kolekcja znanych typów. Dla każdego typu określana jest nazwa kwalifikowana elementu XML, który będzie reprezentował dany typ. Analizowane są pola i właściwości przeznaczone do serializacji budowana jest kolekcja znanych elementów typu. Typ każdego z tych elementów również jest analizowany i współtworzy kolekcję znanych typów.

Nazwy kwalifikowane mogą zawierać przestrzenie nazw XML. Wszystkie przestrzenie nazw są rejestrowane w kolekcji znanych przestrzeni nazw i potem są im przypisywane różne prefiksy.

### Nazwa kwalifikowana

Nazwa kwalifikowana, reprezentowana przez strukturę QualifiedName, składa się z dwóch pól:

* Namespace: string,
* Name: string.

gdzie Namespace reprezentuje przestrzeń nazw języka C# (CLR), do której należy typ, a Name reprezentuje nazwę typu. Klasa QualifiedName została zadeklarowana osobno jako implementująca interfejs IComparable, aby można jej było użyć przy sortowaniu kluczy w klasie SortedDictionary. Należy ją odróżnić od standardowej klasy XmlQualifiedName, w której właściwość Namespace reprezentuje przestrzeń nazw Xml (a dokładnie prefiks nazwy elementu lub atrybutu XML).

### Kolekcja znanych typów

Kolekcja ta jest reprezentowana przez klasę KnownTypesCollection wypełnianą przez obiekt klasy XmlSerializationInfoMapper i udostępnianą przez statyczną właściwość KnownTypes. Właściwość jest statyczna i może być wykorzystywana przy wielokrotnej serializacji takiego samego zbioru typów.

Klasa KnownTypesCollection jest specjalizacją kolekcji uogólnionej TypeInfoCollection<TypeNameInfo>, gdzie typ elementu TypeNameInfo musi implementować interfejs ITypeNameInfo. Typem elementu dla KnownTypesCollection jest SerializationTypeInfo. Takie rozwiązanie wprowadzono, aby ujednolicić obsługę kolekcji KnownTypesCollection i KnownMembersCollection. Ta druga kolekcja zawiera informacje o elementach składowych (właściwościach i polach) typu i jest również specjalizacją kolekcji uogólnionej TypeInfoCollection<ItemType>, przy czym jej typem elementu jest SerializationMemberInfo. Interfejs ITypeNameInfo wiąże nazwę kwalifikowaną QualifiedName z typem. Deklaruje on tylko dwie właściwości:

* Name: QualifiedName,
* Type: Type.

Klasa TypeInfoCollection< TypeNameInfo> w istocie składa się z dwóch słowników:

* TypeIndexedItems: Dictionary<Type, TypeNameInfo>,
* NameIndexedItems: SortedDictionary<QualifiedName, TypeNameInfo>.

Pierwszy słownik dla każdego typu napotkanego przy analizie przypisuje element klasy implementującej ITypeNameInfo, a drugi tworzy indeks na podstawie nazw tych samych elementów.

### Kolekcja znanych składowych typu

Każdy element informujący o serializacji typu zawiera kolekcję KnownMembersCollection udostępniana przez właściwość KnownMembers oraz dwie kwerendy: PropertiesAsAttributes oraz PropertiesAsElements.

### Kolekcja znanych przestrzeni nazw

Przy analizie typów tworzona jest też kolekcja KnownNamespacesCollection udostępniana przez właściwość KnownNamespaces klasy XmlSerializationInfoMapper. Zawiera ona trójki wartości:

* XmlNamespace: string – reprezentuje przestrzeń nazw XML,
* ClrNamespace?: string – reprezentuje przestrzeń nazw CLR,
* Prefix?: string – reprezentuje prefiks nazw XML.

Pole XmlNamespace jest zawsze niepuste, a z pozostałych dwóch przynajmniej jedno musi być określone.

Oprócz zwykłej listy trójek kolekcja zawiera słowniki:

* ClrToXmlNamespace – odwzorowuje przestrzeń nazw Clr w przestrzeń nazw XML.
* NamespaceToPrefix – odwzorowuje przestrzeń nazw XML w prefiks,
* PrefixToNamespace – odwzorowuje prefiks w przestrzeń nazw XML.

Sposób wypełniania tej kolekcji opisano w sekcji 5.6.1 *Różne przestrzenie nazw*.

### Informacja o serializowanym typie

Informacja o typie zapisywana w kolekcji KnownTypes jest typu SerializationTypeInfo, który implementuje ITypeNameInfo.

Tab. . Właściwości klasy SerializationTypeInfo

|  |  |  |  |
| --- | --- | --- | --- |
| Właściwość | Typ | Znaczenie | Uwagi |
| Name | QualifiedName | Nazwa kwalifikowana przestrzenią nazw. Jeśli typ zawiera atrybut [XmlRoot], to nazwą jest wartość tego atrybutu, a jak nie, to nazwa samego typu. |  |
| Type | Type? | Typ, którego ta informacja dotyczy. | XmlIgnore |
| KnownConstructor | ConstructorInfo? | Informacja o konstruktorze bezparametrowym (z refleksji typu). | XmlIgnore |
| HasKnownConstructor | bool | Czy ma konstruktor bezparametrowy? | readonly |
| TypeConverter | TypeConverter? | Konwerter do/z postaci tekstowej. |  |
| XmlConverter | XmlConverter? | Konwerter do/z XML. |  |
| Properties­AsAttributes | KnownProperties­Dictionary | Informacje o właściwościach serializowanych jako atrybuty XML. |  |
| Properties­AsElements | KnownProperties­Dictionary | Informacje o właściwościach serializowanych jako elementy XML. |  |
| ContentProperty | Serialization­­Property­Info? | Informacja o właściwości reprezentującej zawartość. |  |
| TextProperty | Serialization­­Property­Info? | Informacja o właściwości reprezentującej tekst |  |
| IsCollection | bool | Czy typ jest kolekcją (ale nie słownikiem)? | readonly |
| IsDictionary | bool | Czy typ jest słownikiem? | readonly |
| CollectionInfo | CollectionInfo? | Dodatkowa informacja o typie będącym kolekcją lub słownikiem. |  |

Uwaga: Typy nullowane nie są rejestrowane w kolekcji KnownTypes. Zamiast nich są rejestrowane ich typy bazowe.

### Znany konstruktor

Konstruktor klasy musi znany, gdy instancje danej klasy są serializowane i deserializowane jako właściwości do zapisu/odczytu. Nie musi być znany, gdy klasa jest kolekcją lub słownikiem i jest dostępna jako właściwość tylko do odczytu. Nie musi być znany, gdy klasa jest typu prostego (np. liczba, łańcuch).

Konstruktor jest poszukiwany w danej klasie przez refleksję typu jako konstruktor publiczny, bezparametrowy. Czy został znaleziony, może być sprawdzane dopiero podczas deserializacji. Zależy do od opcji IgnoreMissingConstructor. Jeśli nie jest ustawiona, to zgłaszany jest wyjątek. Jeśli jest ustawiona, to wyjątek nie jest zgłaszany i obiekt może być serializowany (chociaż nie będzie mógł być deserializowany).

### Konwersja wartości

Klasa może mieć zadeklarowany własne konwertery. Są przewidziane dwa konwertery:

* TypeConverter – konwerter do/z postaci tekstowej wykorzystywany z przestrzeni nazw System.ComponentModel,
* XmlConverter – konwerter do/z postaci XML utworzony na wzór konwertera JSON.

TypeConverter został zadeklarowany przez Microsoft w przestrzeni nazw [System.ComponentModel](https://docs.microsoft.com/pl-pl/dotnet/api/system.componentmodel?view=net-6.0) z metodami wirtualnymi, które mogą być nadpisane w klasach potomnych. Aby był wykorzystywany przy serializacji/deserializacji, musi mieć możliwość konwersji wartości danego typu do/z typu string. Musi też być przypisany do typu lub właściwości przez atrybut [XmlTypeConverter].

XmlConverter jest nowozdefiniowaną klasą abstrakcyjną o następującej treści:

public virtual bool CanRead => true;

public virtual bool CanWrite => true;

public abstract void WriteXml(XmlWriter writer, object? value, QXmlSerializer? serializer);

public abstract object? ReadXml(XmlReader reader, SerializationTypeInfo objectTypeInfo, SerializationPropertyInfo? propertyInfo, SerializationItemInfo? itemInfo, QXmlSerializer? serializer);

public abstract bool CanConvert(Type objectType);

Programista może na tej podstawie napisać własny konwerter i przypisać go do wybranego typu lub właściwości atrybutem [XmlTypeConverter].

### Właściwości serializowane

W klasie SerializationTypeInfo zadeklarowane są dwa słowniki typu KnownPropertiesDictionary:

* PropertiesAsAttributes,
* PropertiesAsElements.

Słownik PropertiesAsAttributes jest tworzony na podstawie właściwości oznaczonych atrybutem [XmlAttribute], a gdy ustawione są opcje AcceptAllProperties i SimplePropertiesAsAttributes − również wszystkie inne właściwości typów prostych o ile nie są oznaczone atrybutem [XmlIgnore]. Akceptowane są tylko właściwości publiczne, które są przeznaczone do odczytu i zapisu. Jeśli atrybutem [XmlAttribute] jest oznaczona właściwość, która nie jest typu prostego, to powinna mieć ona (lub jej typ) ustalony konwerter typu [TypeConverter]. Jeśli nie ma, to właściwość może być serializowana z wykorzystaniem standardowej metody ToString(), ale będzie mogła być deserializowana.

Słownik PropertiesAsElements jest tworzony na podstawie właściwości oznaczonych atrybutem [XmlElement], a gdy ustawione są opcje AcceptAllProperties − również wszystkie inne właściwości o ile nie są oznaczone atrybutem [XmlIgnore] i nie zostały wcześniej zakwalifikowane jako atrybuty. Akceptowane są tylko właściwości publiczne, które są przeznaczone do odczytu i zapisu.

### Właściwości specjalne

W klasie SerializationTypeInfo mogą być też zapisane dwie specjalne właściwości typu SerializationTypeInfo:

* ContentProperty,
* TextProperty.

Właściwość ContentProperty określa właściwość, która jest wykorzystywana do pobierania zawartości elementu przy serializacji i do wstawiania zawartości elementu przy deserializacji. Jest ona wykorzystywana zamiast wszystkich innych właściwości serializowanych jako elementy. Właściwość ContentProperty może być przypisana do **typu** przez atrybut [XmlContentProperty]. Atrybut ten został zdefiniowany w zastępstwie atrybutu [ContentProperty], który jest ograniczony do .NET Framework.

Właściwość TextProperty określa właściwość, która podaje tekst do serializacji zamiast wszystkich innych właściwości serializowanych jako elementy XML. Właściwość TextProperty może być przypisana do **właściwości** przez atrybut [XmlText].

### Informacje o właściwościach

Informacja o pojedynczej serializowanej właściwości jest zapisywana w klasie SerializationPropertyInfo.

Tab. . Właściwości klasy SerializationPropertyInfo

|  |  |  |  |
| --- | --- | --- | --- |
| Właściwość | Typ | Znaczenie | Uwagi |
| Order | int | Kolejność wykorzystywana do uporządkowania właściwości przy serializacji atrybutów i elementów. |  |
| Name | QualifiedName | Nazwa atrybutu lub elementu XML używana przy serializacji. |  |
| Property | PropertyInfo | Informacja o właściwości z refleksji typu. |  |
| IsNullable | bool | Czy właściwość dopuszcza wartość null? |  |
| IsReference | bool | Czy właściwość jest serializowana jako referencja do obiektu? |  |
| ValueType | Serialization­TypeInfo | Informacja o serializacji typu wartości właściwości. |  |
| TypeConverter | TypeConverter? | Konwerter do/z postaci tekstowej. |  |
| XmlConverter | XmlConverter? | Konwerter do/z XML. |  |
| HasCheck­Method | bool | Czy ma opcjonalną metodę do sprawdzenia, czy właściwość ma być serializowana. | readonly |
| Check­Method | MethodInfo? | Opcjonalna metoda do sprawdzenia, czy właściwość ma być serializowana. | XmlIgnore |
| IsPolymorfic | bool | Czy właściwość jest polimorficzna? | readonly |
| KnownSubtypes | Known­Types­Dictionary? | Opcjonalna kolekcja informacji o podtypach wartości |  |
| IsCollection | bool | Czy właściwość jest kolekcją (ale nie słownikiem)? | readonly |
| IsDictionary | bool | Czy właściwość jest kolekcją? | readonly |
| CollectionInfo | CollectionInfo? | Opcjonalna informacja o właściwości będącej kolekcją lub słownikiem. |  |

Właściwość Order może być ustalana za pomocą pola Order atrybutu [XmlElement] lub za pomocą dodatkowo zadeklarowanego atrybutu [SerializationOrder]. Ten dodatkowy atrybut uzupełnia informację standardowego atrybutu [XmlAttribute], który nie ma pola Order. Domyślnie właściwości są numerowane od 1, więc podanie atrybutu [SerializationOrder(0)] powinno ustawić właściwość przed innymi. Jeśli klasa dziedziczy właściwości do serializacji po innej klasie, to właściwości odziedziczone są serializowane przed właściwościami danej klasy.

Właściwość Name określa nazwę atrybutu lub elementu, w którym jest zapisywana wartość właściwości. Jest ona ustalana albo przez nazwę właściwości (z uwzględnieniem opcji AttributeNameCase lub ElementNameCase), albo przez któryś z atrybutów: [XmlAttribute], [XmlElement], [XmlArray].

Informacje TypeConverter i XmlConverter oraz CollectionInfo w SerializationPropertyInfo mogą być różne od analogicznych informacji zapisanych w SerializationTypeInfo.

### Właściwości polimorficzne

Jeśli właściwość akceptuje wartości różnych typów, to muszą one pochodzić od wspólnego typu i muszą być znane podczas mapowania typów.

Można to zrobić przypisując atrybut [KnownType] (z przestrzeni nazw System.Runtime.Serialization) do klasy nadrzędnej, np.:

[KnownType(typeof(DictionaryInfo))]

public class CollectionInfo

{

}

albo na poziomie samej właściwości, np.

[KnownType(typeof(DictionaryInfo))]

public CollectionInfo CollectionInfo { get; set; }

wówczas typy pochodne są określone w kolekcji KnownSubtypes. Kolekcja ta może być określona na poziome typu wartości, np.

### Kolekcje i słowniki

Jeśli typ jest kolekcją, to do informacji do SerializationTypeInfo jest dołączany obiekt klasy CollectionInfo (lub pochodnej klasy DictionaryInfo) i ustawiona jest właściwość IsCollection (albo IsDictionary).

Klasa CollectionInfo ma dwie właściwości, a klasa DictionaryInfo rozbudowuje ją o dodatkowe cztery właściwości.

Tab. . Właściwości klasy CollectionInfo i DictionaryInfo

|  |  |  |
| --- | --- | --- |
| Właściwość | Typ | Znaczenie |
| Właściwości wspólne | | |
| StoresReferences | bool | Czy kolekcja serializuje obiekty jako referencje? Jeśli nie, to jako obiekty zagnieżdżone. |
| KnownItemTypes | KnownItem­TypesDictionary | Kolekcja znanych typów składowych. |
| Tylko DictionaryInfo | | |
| KeyTypeInfo | SerializationTypeInfo | Nazwa właściwości typu składowego, która służy za klucz. |
| ValueTypeInfo | SerializationTypeInfo | Nazwa właściwości typu składowego, która służy za klucz. |
| KeyName | string? | Nazwa właściwości typu składowego, która służy za klucz. |

## Rozwiązywanie problemów

### Różne przestrzenie nazw

Standardowy XmlSerializer ma ograniczone możliwości związane z obsługą przestrzeni nazw. Znajdywane w czasie analizy przestrzenie nazw XML są pobierane z atrybutów, takich jak [XmlRoot] i [XmlElement]. Konstruktor serializatora umożliwia podanie domyślnej przestrzeni nazw, która jest przypisywana tym elementom, które nie mają takiego oznaczenia. Te mechanizmy umożliwiają przypisanie przestrzeni nazw XML w postaci URL-i. Prefiksy są przydzielane przez serializator samodzielnie, a programista nie ma nad tym kontroli.

Z kolei metoda Serialize umożliwia podanie parametru typu XmlNamespaces, który zawiera przestrzenie nazw w postaci URL z już przypisanymi prefiksami. Ten drugi mechanizm nie przydziela przestrzeni nazw do elementów, lecz służy jedynie do przypisania prefiksów do przestrzeni nazw XML.

Na tym możliwości standardowego serializatora w zakresie przestrzeni nazw się kończą. Problemy się zaczynają, gdy serializowane klasy należą do różnych przestrzeni nazw.

Rozważmy przykładowe definicje dwóch klas, które powinny być serializowane w różnych przestrzeniach nazw XML:

[XmlRoot(Namespace = "http://rootNamespace")]

public class Class1

{

public Class2 Class2 { get; set; } = new();

}

[XmlRoot(Namespace = "http://specifiedNamespace")]

public class Class2ForSerializeTwoClassesWithNamespaceTest

{

}

oraz kod serializacji:

var testInstance = new Class1ForSerializeTwoClassesWithNamespaceTest();

var serializer = new XmlSerializer(typeof(Class1ForSerializeTwoClassesWithNamespaceTest));

var namespaces = new XmlSerializerNamespaces();

namespaces.Add("root", "http://rootNamespace");

namespaces.Add("spec", "http://specificNamespace");

using (var txtWriter = File.CreateText("SerializeTwoClassesWithNamespace.xml"))

{

using (var xmlWriter = XmlWriter.Create(txtWriter, new XmlWriterSettings { Indent = true }))

{

serializer.Serialize(xmlWriter, testInstance, namespaces);

}

}

Kod ten da wynik w postaci:

<?xml version="1.0" encoding="utf-8"?>

<root:Class1 xmlns:spec="http://specificNamespace" xmlns:root="http://rootNamespace">

<root:Class2 />

</root:Class1>

<?xml version="1.0" encoding="utf-8"?>

Wbrew oczekiwaniom element podklasy zawierający instancję klasy drugiej został zserializowany w przestrzeni nazw klasy pierwszej. Stało się tak dlatego, że atrybut [XmlRoot] jest odczytywany tylko raz – dla klasy głównej podanej w konstruktorze serializatora. Jeśli chcemy, aby instancja klasy drugiej była zserializowana w osobnej przestrzeni nazw, to musimy podać atrybut [XmlElement] dla odpowiedniej właściwości w klasie pierwszej:

[XmlRoot(Namespace = "http://rootNamespace")]

public class Class1

{

[XmlElement(Namespace = "http://specificNamespace")]

public Class2 Class2 { get; set; } = new();

}

public class Class2ForSerializeTwoClassesWithNamespaceTest

{

}

Wówczas otrzymamy:

<?xml version="1.0" encoding="utf-8"?>

<root:Class1 xmlns:spec="http://specificNamespace" xmlns:root="http://rootNamespace">

<spec:Class2 />

</root:Class1>

Ten mechanizm całkowicie ignoruje przestrzenie nazw CLR. Wyobraźmy sobie dwie klasy o tej samej nazwie zadeklarowane w różnych przestrzeniach nazw CLR:

[XmlRoot(Namespace = "http://rootNamespace")]

public class Class1

{

public Specific.Class1 Subclass { get; set; } = new();

}

namespace Specific

{

public class Class1 { }

}

Zauważmy, że właściwość reprezentująca klasę drugą nie może mieć nazwy Class1, bo zabraniają tego reguły języka C#. Jeśli klasa w odrębnej przestrzeni CLR nie ma żadnego atrybutu, to serializator wyrzuci wyjątek, bowiem w jednej przestrzeni XML miałyby się znaleźć dwie różne klasy o tej samej nazwie.

Wyjątku można się pozbyć deklarując inną przestrzeń nazw w atrybucie [XmlRoot] dla drugiej z klas:

[XmlRoot(Namespace = "http://rootNamespace")]

public class Class1

{

public Specific.Class1 Subclass { get; set; } = new();

}

namespace Specific

{

[XmlRoot(Namespace = "http://specificNamespace")]

public class Class1 { }

}

Jednak przy takiej deklaracji w uzyskanym wyniku nie ma rozróżnienia między przestrzeniami nazw:

<?xml version="1.0" encoding="utf-8"?>

<root:Class1 xmlns:spec="http://specificNamespace" xmlns:root="http://rootNamespace">

<root:Subclass />

</root:Class1>

Pamiętając o tym, że XmlSerializer aktywnie stosuje przestrzeń nazw z atrybutu [XmlElement], można umieścić ten atrybut przy deklaracji właściwości:

[XmlRoot(Namespace = "http://rootNamespace")]

public class Class1

{

[XmlElement(Namespace = "http://specificNamespace")]

public Specific.Class1 Subclass { get; set; } = new();

}

namespace Specific

{

public class Class1 { }

}

Wówczas uzyskamy właściwość Subclass w dokumencie wynikowym będzie wskazywała na inną przestrzeń nazw.

<?xml version="1.0" encoding="utf-8"?>

<root:Class1 xmlns:spec="http://specificNamespace" xmlns:root="http://rootNamespace">

<spec:Subclass />

</root:Class1>

Zwróćmy uwagę, że taki atrybut [XmlElement] musi wskazywać na inną przestrzeń nazw XML, bo podanie tej samej przestrzeni (jak poniżej) znowu powoduje wyjątek serializacji:

[XmlRoot(Namespace = "http://rootNamespace")]

public class Class1

{

[XmlElement(Namespace = "http://rootNamespace")]

public Specific.Class1 Subclass { get; set; } = new();

}

namespace Specific

{

public class Class1 { }

}

To oznacza, że chociaż atrybut [XmlElement] jest umieszczany przy właściwości, to tak naprawdę dotyczy jej wartości, czyli w tym wypadku klasy elementu udostępnianego przez tę właściwość. Prowadzi to jednak do ciekawych efektów ubocznych, np. przy deklaracji dwóch różnych przestrzeni nazw przy dwóch różnych właściwościach tego samego typu:

[XmlRoot(Namespace = "http://rootNamespace")]

public class Class1

{

[XmlElement(Namespace = "http://specificNamespace")]

public Specific.Class1 Subclass1 { get; set; } = new();

[XmlElement(Namespace = "http://alternativeNamespace")]

public Specific.Class1 Subclass2 { get; set; } = new();

}

namespace Specific

{

public class Class1 { }

}

ta sama klasa wydaje się być dostępna przez różne przestrzenie nazw:

<?xml version="1.0" encoding="utf-8"?>

<root:Class1 xmlns:spec="http://specificNamespace" xmlns:alt="http://alternativeNamespace" xmlns:root="http://rootNamespace">

<spec:Subclass1 />

<alt:Subclass2 />

</root:Class1>

Przestrzenie nazw w klasie DataContractSerializer

Nieco inaczej ten problem został ujęty w klasie DataContractSerializer. Nie czyta on elementów [XmlRoot] ani [XmlElement], ale atrybuty z przestrzeni nazw System.Runtime.Serialization. Przy braku informacji na temat przestrzeni nazw XML ustala je automatycznie w oparciu o przestrzenie nazw CLR. Przykładowo ostatnia z powyższych deklaracji klas, przy kodzie serializacji:

var testInstance = new Class1();

var serializer = new DataContractSerializer(typeof(Class1));

using (var txtWriter = File.CreateText("SerializeTwoClassesWithNamespace.xml"))

{

using (var xmlWriter = XmlWriter.Create(txtWriter, new XmlWriterSettings { Indent = true }))

{

serializer.WriteObject(xmlWriter, testInstance);

}

}

daje następujący dokument XML:

<?xml version="1.0" encoding="utf-8"?>

<Class1 xmlns="http://schemas.datacontract.org/2004/07/">

<Subclass1 xmlns:d2p1="http://schemas.datacontract.org/2004/07/Specific" />

<Subclass2 xmlns:d2p1="http://schemas.datacontract.org/2004/07/Specific" />

</Class1>

Przestrzeń nazw "http://schemas.datacontract.org/2004/07/" jest domyślną przestrzenią bazową, do której serializator dodał przyrostek „Specific” oznaczający przestrzeń nazw CLR, w której jest zadeklarowana druga z klas. Zauważmy, że pomimo deklaracji xmlns:d2p1="http://schemas.datacontract.org/2004/07/Specific" w każdym z elementów Subclass1, Subclass2, same nazwy tych elementów nie są poprzedzone żadnym prefiksem, więc należą do tej samej domyślnej przestrzeni nazw co Class1.

<?xml version="1.0" encoding="utf-8"?>

<Class1 xmlns:d1p1="http://schemas.datacontract.org/2004/07/" xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://rootNamespace">

<d1p1:Subclass1 xmlns:d2p1="http://schemas.datacontract.org/2004/07/Specific" />

<d1p1:Subclass2 xmlns:d2p1="http://schemas.datacontract.org/2004/07/Specific" />

</Class1>

Dla serializatora kontraktu danych można podać domyślną przestrzeń nazw, chociaż trzeba wówczas podać też nazwę elementu głównego – jest to parametr poprzedzający przestrzeń nazw, który nie może być zastąpiony ani wartością null, ani łańcuchem pustym.

var serializer = new DataContractSerializer(typeof(Class1), "Class1", "http://rootNamespace");

Zadeklarujmy jeszcze jedną klasę w głównej przestrzeni nazw:

public class Class1

{

public Specific.Class1 Subclass1 { get; set; } = new();

public Class2 Subclass2 { get; set; } = new();

}

public class Class2 { }

namespace Specific

{

public class Class1 { }

}

i zobaczmy wynik:

<Class1 xmlns:d1p1="http://schemas.datacontract.org/2004/07/" xmlns="http://rootNamespace">

<d1p1:Subclass1 xmlns:d2p1="http://schemas.datacontract.org/2004/07/Specific" />

<d1p1:Subclass2 />

</Class1>

Przestrzeń nazw "http://rootNamespace" została zastosowana tylko dla elementu głównego. Druga z klas w głównej przestrzeni nazw CLR (Class2) dostępna poprzez właściwość Subclass2 ma przypisaną domyślną przestrzeń nazw XML związaną z prefiksem "d1p1". Klasa zadeklarowana w drugiej przestrzeni nazw CLR (przez właściwość Subclass1) jest w dalszym ciągu związana z tym samym prefiksem (mimo zadeklarowania innego prefiksu i innej przestrzeni nazw XML).

Sposobem na ustalenie przestrzeni nazw dla klas jest użycie atrybutu [DataContract]. Wtedy koniecznie trzeba podać atrybuty [DataMember] dla tych właściwości, które mają być serializowane. Atrybut [DataContract] umożliwia podanie przestrzeni nazw, [DataMember] – nie.

[DataContract(Namespace = "http://rootNamespace")]

public class Class1

{

[DataMember]

public Specific.Class1 Subclass1 { get; set; } = new();

[DataMember]

public Class2 Subclass2 { get; set; } = new();

}

namespace Specific

{

[DataContract(Namespace = "http://specificNamespace")]

public class Class1

{

[DataMember]

public string Name { get; set; } = "Subclass1";

}

}

[DataContract(Namespace = "http://rootNamespace")]

public class Class2

{

[DataMember]

public string Name { get; set; } = "Class2";

}

W wyniku klasa Class2 dostępna przez właściwość Subclass2 jest przypisana do tej samej przestrzeni nazw XML ("http://rootNamespace"), ale niestety klasa Specific.Class1 dostępna przez właściwość Subclass2 również (zamiast do "http://specificNamespace"). Dopiero właściwości wewnętrzne klasy Specific.Class1 należą do przestrzeni nazw "http://specificNamespace".

<?xml version="1.0" encoding="utf-8"?>

<Class1 xmlns="http://rootNamespace">

<Subclass1 xmlns:d2p1="http://specificNamespace">

<d2p1:Name>Subclass1</d2p1:Name>

</Subclass1>

<Subclass2>

<Name>Class2</Name>

</Subclass2>

</Class1>

Ten sam efekt, ale bez konieczności stosowania atrybutów [DataMember] daje zastosowanie globalnego atrybutu [assembly: ContractNamespace]:

[assembly: ContractNamespace("http://rootNamespace")]

[assembly: ContractNamespace("http://specificNamespace", ClrNamespace = "Specific")]

public class Class1

{

public Specific.Class1 Subclass1 { get; set; } = new();

public Class2 Subclass2 { get; set; } = new();

}

namespace Specific

{

public class Class1

{

public string Name { get; set; } = "Subclass1";

}

}

public class Class2

{

public string Name { get; set; } = "Class2";

}

Nie ma sposobu, aby poprzedzić nazwę elementu Subclass1 prefiksem "d2p1" odpowiednim dla przestrzeni nazw "http://specificNamespace". Zresztą wprowadziłoby to pewną nieprawidłowość – właściwość Subclass1 należy do przestrzeni "http://rootNamespace", to klasa, do której się odwołuje, należy do przestrzeni "http://specificNamespace".

Przestrzenie nazw w serializacji XAML

Rozwiązanie problemu znajduje się przy serializacji XAML przez klasę XmlWriter zadeklarowaną w przestrzeni nazw System.Windows.Markup w bibliotece PresentationFramework.dll. Ponieważ ta biblioteka występuje tylko w wersjach .NET Framework dla Windows, więc XmlWriter nie spełnia wymagań przenośności dla serializatora. Może jednak posłużyć za wzór. Przestrzenie nazw XML są w dokumencie wynikowym deklarowane w elemencie głównym na podstawie przestrzeni nazw CLR. Właściwości udostępniające obiekty złożone są serializowane jako elementy XML z nazwą właściwości poprzedzoną nazwą klasy nadrzędnej. Obiekty klasy podrzędnej są serializowane jako elementy składowe. W naszym przykładzie klasa Specific.Class1 ma nazwę elementu przypisaną przez prefiks do przestrzeni nazw XML "clr-namespace:XamlSerializationTest.Specific".

<?xml version="1.0" encoding="utf-8"?>

<Class1 xmlns="clr-namespace:XamlSerializationTest.Root;assembly=XamlSerializationTest"

xmlns:xsts="clr-namespace:XamlSerializationTest.Specific;assembly=XamlSerializationTest">

<Class1.Subclass1>

<xsts:Class1 Name="Subclass1" />

</Class1.Subclass1>

<Class1.Subclass2>

<Class2 Name="Class2" />

</Class1.Subclass2>

</Class1>

Własne rozwiązanie

We serializatorze QXmlSerializer stosowane są rozwiązania mieszane. Mechanizm jest następujący:

* Typy analizowane przy inicjowaniu serializatora są rejestrowane w kolekcji KnownTypes jako indeksowane nazwami kwalifikowanymi QualifiedName, z których każda zawiera nazwę XML i przestrzeń nazw CLR. Nazwa XML jest pobierana z atrybutu [XMLRoot], o ile występuje, albo jest tworzona na podstawie nazwy typu.
* Jeśli typ ma przypisany atrybut [XmlRoot] i ma on określoną przestrzeń nazw XML, to jest ona rejestrowana w kolekcji KnownNamespaces wraz z przestrzenią nazw CLR, do której należy dany typ.
* Przy analizowaniu składowych typu, jeśli typ jeszcze nie był zarejestrowany, to jest rejestrowany i poddawany analizie rekurencyjnej.
* Składowe są rejestrowane w kolekcji KnownMembers przypisanej do danego typu i indeksowane nazwami kwalifikowanymi QualifiedName, z których każda zawiera nazwę XML składowej i przestrzeń nazw CLR, do której należy składowa. Z reguły przestrzenią nazw CLR jest przestrzeń nazw klasy, w której składowa jest zadeklarowana. Nazwa XML składowej jest pobierana z atrybutu [XMLAttribute], [XmlElement] lub [XmlArray], o ile taki występuje, albo jest tworzona na podstawie nazwy składowej z uwzględnieniem opcji.
* Jeśli składowa ma przypisany atrybut [XmlAttribute,] [XmlElement] lub [XmlArray] i ma on określoną przestrzeń nazw XML, to jest ona rejestrowana w kolekcji KnownNamespaces wraz z przestrzenią nazw CLR, do której należy dana składowa.
* Jeśli asemblat, do którego należy typ ma określony atrybut [ContractNamespace] i opcja AcceptContractNamespaces jest włączona, to para przestrzeni nazw XML i CLR z tego atrybutu jest rejestrowana w kolekcji KnownNamespaces.
* W kolekcji KnownNamespaces rejestrowane są tylko unikatowe pary przestrzeni nazw XML-CLR. Jednej przestrzeni nazw XML może być przypisanych wiele przestrzeni CLR, ale jedna przestrzeń CLR może mieć tylko jedną przestrzeń XML.

Po zakończeniu analizy przestrzenie nazw z kolekcji KnownTypes, KnownMembers i KnownNamespaces są rozwiązywane metodą Resolve(string defaultNamespace) zdefiniowaną w klasie XmlSerializationInfoMapper. Najpierw sprawdzane jest, czy każda przestrzeń nazw CLR z nazw kwalifikowanych QualifiedName ma określoną przestrzeń nazw XML. Jeśli nie, to możliwe są dwa przypadki:

* Dana przestrzeń nazw CLR może mieć nadrzędną przestrzeń nazw, która ma określoną przestrzeń nazw XML.
* Dana przestrzeń nazw CLR nie ma nadrzędnej przestrzeń nazw, która miałaby określoną przestrzeń nazw XML.

W pierwszym przypadku sprawdza się, czy w danej przestrzeni nazw nie ma przypadkiem typów do serializacji, które wchodziłyby w konflikt nazw z typami z przestrzeni nadrzędnej. Jeśli nie, to danej przestrzeni nazw przydzielana jest taka sama przestrzeń nazw XML, jak przestrzeni nadrzędnej. Jeśli konflikt występuje, to danej przestrzeni nazw jest przypisywana nowa nazwa przestrzeni XML utworzona przez dodanie składowej nazwy podrzędnej CLR do nadrzędnej nazwy XML.

W drugim przypadku najwyższej nadrzędnej przestrzeni nazw CLR przydzielana jest nowa przestrzeń nazw XML z przedrostkiem "clr-namespace:", a z podrzędnymi podstępuje się tak, jak w pierwszym przypadku.

Każdej osobnej przestrzeni nazw XML przydzielany jest prefiks. Przestrzeni przekazanej jako parametr jest przypisywany prefiks pusty, a pozostałym prefiksy „n\_”, gdzie w miejsce „\_” pisywane są kolejne numery 1, 2 itd.

## Serializacja

W czasie serializacji tworzone jest drzewo dokumentu XML począwszy od elementu przekazanego jako parametr do metody Serialize. Do serializacji wykorzystywany jest element klasy XmlWriter z przestrzeni nazw System.Xml.Serialize.

### Nagłówek XML

Nagłówek XML to napis

<?xml version="1.0" encoding="utf-8"?>

umieszczany na początku dokumentu. Można pominąć nagłówek poprzez opcję XmlWriterSettings.OmitXmlDeclaration podawaną przy tworzeniu instancji klasy XmlWriter.

### Przestrzenie nazw