Zaawansowane programowanie obiektowe - wykład 7

Opracowanie: dr Piotr Jastrzębski

XML, JSON, REST-API

Wstęp

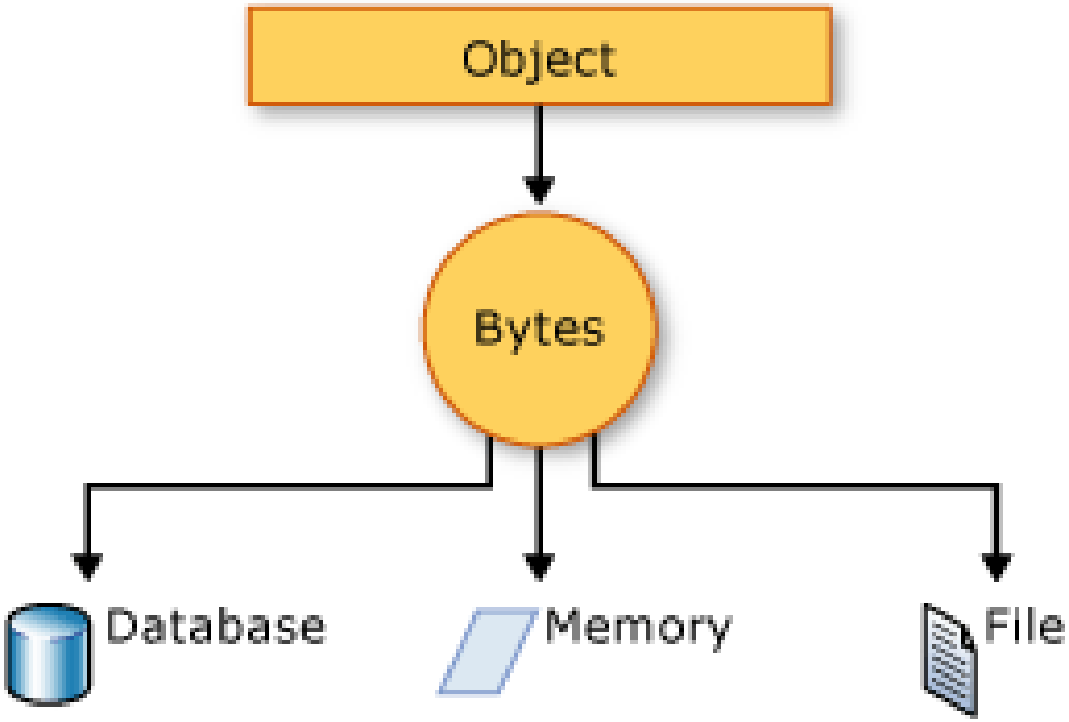
Po co potrzebne jest programowanie?

Jaki kod? - przyjazny dla maszyn czy ludzi?

Serializacja

Serializacja – w programowaniu proces przekształcania obiektów, tj. instancji określonych klas, do postaci szeregowej, czyli w strumień bajtów, z zachowaniem aktualnego stanu obiektu. Serializowany obiekt może zostać utrwalony w pliku dyskowym, przesłany do innego procesu lub innego komputera poprzez sieć. Procesem odwrotnym do serializacji jest deserializacja. Proces ten polega na odczytaniu wcześniej zapisanego strumienia danych i odtworzeniu na tej podstawie obiektu klasy wraz z jego stanem bezpośrednio sprzed serializacji.

Serializacja służy do zapisu stanu obiektu, a później do odtworzenia jego stanu. Mechanizm ten jest używany między innymi na platformach .NET, Java, PHP, Python, Ruby.



Rysunek 1:

XML

XML (ang. Extensible Markup Language, w wolnym tłumaczeniu Rozszerzalny Język Znaczników) – uniwersalny język znaczników przeznaczony do reprezentowania różnych danych w strukturalizowany sposób. Jest niezależny od platformy, co umożliwia łatwą wymianę dokumentów pomiędzy heterogenicznymi (różnymi) systemami i znacząco przyczyniło się do popularności tego języka w dobie Internetu. XML jest standardem rekomendowanym oraz specyfikowanym przez organizację W3C. (Wikipedia)

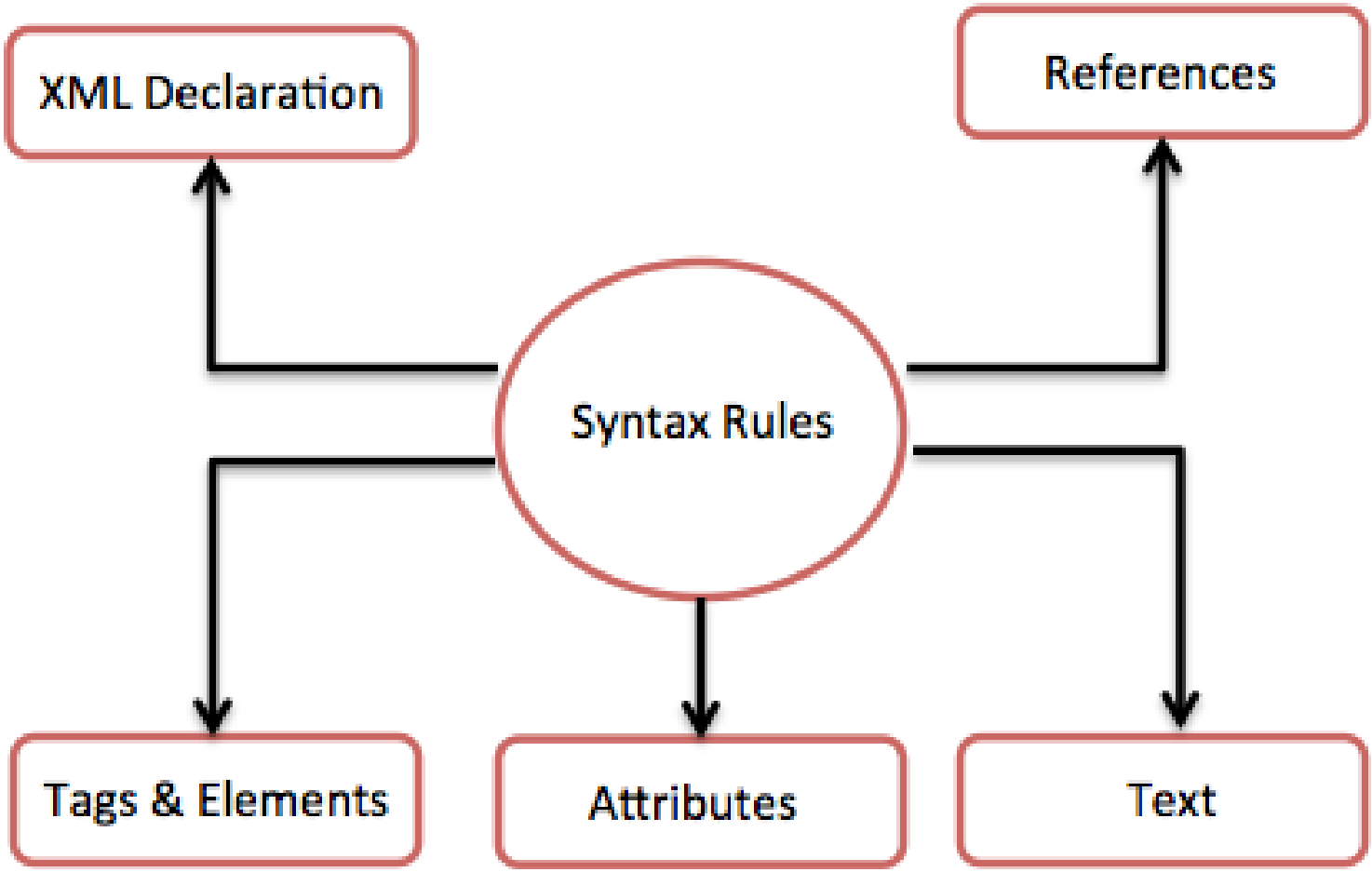
Przykłady:

- https://www.w3schools.com/xml/xml_examples.asp

Zastosowanie:

- http://www.inzynieriawiedzy.pl/reprezentacja-wiedzy/xml

```
<?xml version = "1.0"?>
<contact-info>
   <name>Tanmay Patil</name>
   <company>TutorialsPoint</company>
   <phone>(011) 123-4567</phone>
</contact-info>
```


XML Declaration
References
Syntax Rules
Tags & Elements
Attributes
Text

Rysunek 2:

Węzeł	Opis	Angielska nazwa	Symbol relacji[1]
korzeń	nadrzędny element	root	:root
przodek	element zawierający inny	ancestor	przodek potomek
potomek	element będący w innym	descedant	przodek potomek
rodzic	element bezpośrednio zawierający	parent	rodzic > dziecko rodzic / dziecko
dziecko	element bezpośrednio zawierany	child	rodzic > dziecko rodzic / dziecko
węzeł	każda część dokumentu	node	---

Rysunek 3:

Deklaracja

```
<?xml version = "1.0" encoding = "UTF-8"?>
```

Składnia:

```
<?xml
   version = "version_number"
   encoding = "encoding_declaration"
   standalone = "standalone_status"
?>
```

Elementy

```
<element>....</element>
```

lub

```
<element/>
```

Zagnieżdżanie

```
<?xml version = "1.0"?>
<contact-info>
   <company>TutorialsPoint</company>
<contact-info>
```

Root element

Źle:

```
<x>...</x>
<y>...</y>
```

Dobrze:

```
<root>
   <x>...</x>
   <y>...</y>
</root>
```

Case-sensitive:

<contact-info> jest różne od <Contact-Info>

Atrybuty XML:

```
<a b = "x" c = "y" b = "z">....</a>
```

Komentarze:

```
<!--Your comment-->
```

Entities (znaki specjalne):

Ampersand - `&`

Single quote - `&apos;`

Greater than - `>`

Less than - `<`

Double quote - `&quot;`

XML-DOM

DOM (Document Object Model) to po prostu pewien obiektowy model dokumentu, który jest najpopularniejszym standardem przetwarzania dokumentów XML. Sam w sobie nie ma zbyt wiele wspólnego z dokumentami XML jako takimi, ponieważ ich dotyczą zasady składni. Model danych jest bardziej abstrakcyjną formą opisu i zarówno istnieje w głowach inżynierów, jak i jest implementowany w programach komputerowych.

XML-DOM w C#

Strona dokumentacji - https://docs.microsoft.com/en-us/dotnet/standard/data/xml/reading-an-xml-document-into-the-dom

https://docs.microsoft.com/en-us/dotnet/standard/data/xml/process-xml-data-using-the-dom-model

LINQ to XML

Przestrzeń nazw: `using System.Xml.Linq;`

JSON

JSON, JavaScript Object Notation – lekki format wymiany danych komputerowych. JSON jest formatem tekstowym, bazującym na podzbiorze języka JavaScript. Typ MIME dla formatu JSON to application/json. Format został opisany w dokumencie RFC 4627 .

Pomimo nazwy, JSON jest formatem niezależnym od konkretnego języka. Wiele języków programowania obsługuje ten format danych przez dodatkowe pakiety bądź biblioteki. Wśród nich są ActionScript, C, C++, C#, ColdFusion, E, Java, JavaScript, ML, Objective CAML, Perl, PHP, Python, R, REBOL oraz Ruby. (Wikipedia)

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

```
<employees>
    <employee>
        <firstName>John</firstName>
        <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName>
        <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName>
        <lastName>Jones</lastName>
    </employee>
</employees>
```

REST-API

REST – Representational State Transfer – styl architektury oprogramowania opierający się o zbiór wcześniej określonych reguł opisujących jak definiowane są zasoby, a także umożliwiających dostęp do nich. Został on zaprezentowany przez Roya Fieldinga w 2000 roku.

API – Application Programming Interface – zestaw reguł definiujący komunikację pomiędzy programami komputerowymi.

HTTP – Hypertext Transfer Protocol – protokół, z którego korzystasz codziennie (lub też jego wersji szyfrowanej – HTTPS) podczas przeglądania stron w sieci. Podczas tworzenia REST API do komunikacji z API wykorzystuje się metody HTTP, których łącznie jest 9. Niemniej jednak do zbudowania podstawowego API pozwalającego na odczyt, zapis, aktualizację i usuwanie danych wystarczą tylko 4 metody – GET, POST, PUT i DELETE.

Metody HTTP

GET – służy do pobierania danych. Tutaj wystarczy podać odpowiedni endpoint, ewentualnie zmodyfikować nagłówki(headers) zapytania.

POST – służy tworzeniu i przesłaniu nowych danych. W tym przypadku konieczne jest już stworzenie ciała(body), w którym przekażemy dane do naszego REST API.

PUT – również służy przesyłaniu danych, lecz najczęściej w celu aktualizacji tych danych.

DELETE – metoda służąca do usuwania danych.

Zasady

https://youtu.be/P9b8-BrWdYs

1. Odseparowanie interfejsu użytkownika od operacji na serwerze. Klient poprzez "wydawanie poleceń" nie ma wpływu na to co się dzieje po stronie serwera. Działa to również w drugą stronę – serwer daje klientowi jedynie odpowiedź i nie ma prawa ingerować w UI. Pozwala to na korzystanie z jednego REST API w wielu niezależnych od siebie aplikacjach, a dane pozostaną spójne.
2. Bezstanowość – mówi się że REST jest stateless – oznacza to, że każde zapytanie od klienta musi zawierać komplet informacji oraz, że serwer nie przechowuje stanu o sesji użytkownika po swojej stronie.

3. Cacheability – odpowiedź, którą użytkownik otrzyma z REST API musi jasno definiować, czy ma ona być cacheable czy non-cacheable. Ma to znaczenie przy danych, które bardzo szybko stają się nieaktualne oraz przy danych, które aktualizują się relatywnie rzadko – nie ma sensu na przykład cache'ować współrzędnych geograficznych pędzącego samolotu, natomiast już jego kolor czy nazwę już tak.
4. Endpointy, czyli tak naprawdę adresy, które pozwalają na dostęp do informacji powinny jednoznacznie wskazywać do jakiego zasobu się odwołują. Z ich budowy powinniśmy wiedzieć jaki konkretnie zasób otrzymamy. Co ważne – dane otrzymywane w API powinny być niezależne w żaden sposób od schematu bazy danych w jakiej są przetrzymywane. Oczywiście nie ma przeciwwskazań, aby struktura danych wyglądała identycznie jak schemat bazy danych – niemniej jednak struktura w żaden sposób nie powinna zależeć od tego schematu.

5. Separacja warstw – powinniśmy oddzielić warstwy dostępu do danych, logiki oraz prezentacji. Pozwala to na dowolne operacje na danych – nie wymuszamy na użytkowniku konkretnego działania na nich, ani wyświetlania ich w konkretny sposób. Ponadto pośrednie i zewnętrzne API wykorzystywane przez serwer(!) mogą być wykorzystywane bez wiedzy klienta. Przykładem może być wcześniej wspomniany samolot, gdzie informacja np. o kolorze może pochodzić z zupełnie innego API – klient nie musi o tym wiedzieć.
6. Możliwość udostępniania apletów i skryptów klientowi – jest to opcjonalna reguła, aczkolwiek warto rozważyć jej zastosowanie. Jeśli wiemy, że klienci będą wykonywać konkretne operacje na konkretnych danych możemy im udostępnić gotowe do tego rozwiązania.

Swagger

https://apihandyman.io/writing-openapi-swagger-specification-tutorial-part-1-introduction/

https://idratherbewriting.com/learnapidoc/docapis_introtoapis.html

https://idratherbewriting.com/learnapidoc/docs/rest_api_specifications/openapi_openweathermap.yml

Bibliografia

- Wikipedia, wikibooks na licencji CC.
- Joseph Albahari, Ben Albahari, C# 7.0 w pigułce, wyd. Helion, 2018.
- https://www.w3schools.com/xml/xml_examples.asp, dostęp online 30.03.2019.
- https://www.tutorialspoint.com/linq/linq_xml.htm, dostęp online 30.03.2019.
- https://www.tutorialspoint.com/json, dostęp online 30.03.2019.
- https://devszczepaniak.pl/wstep-do-rest-api/, dostęp online 30.03.2019.
- https://docs.microsoft.com/pl-pl/dotnet/csharp/programming-guide/concepts/serialization/, dostęp online 2.04.2019.