Daniel Górski
Korporacyjna Java
Wykład 11: Wybrane formaty danych

Kalendarz

- 8, 15 maja
  - Wykład
  - Nowe zadanie na pracowni
- 22 maja
  - Podsumowanie pracowni
  - Wstępne oddawanie całych projektów
    - Zaczynamy o 8:15 (brak prezentowania przed wykładem)
    - Kilka scenariuszy testowych do przejścia
    - Prezentacja kodu
    - Statyczna prezentacja przepływu sterowania w różnych scenariuszach

Kalendarz...

- 29 maja: zajęcia piątkowe wg kalendarza roku akademickiego
- 5 czerwca
  - Będę dostępny 7:30 – 11:30
  - Możliwość ostatecznej prezentacji projektu
- 12 czerwca: Wykład
  - Kilka osób podłączy się do zdalnej konferencji i opowie o swojej ścieżce edukacyjnej i zawodowej
    - Możliwość dyskusji
    - Będą to osoby z 15 – 20 letnim doświadczeniem w IT

Kalendarz...

- 12 czerwca: Pracownia
  - Będę przeprowadzał testowe rozmowy techniczne
    - Zakres: java, bazy danych, architektura, systemy rozproszone
  - Oceny z pracowni będą już wystawione: na nic to nie będzie rzutować
  - Wyślę dzisiaj email do wszystkich z grupy: jeśli połowa grupy zadeklaruje, że prawie na pewno przyjdzie tego dnia na uczelnię to ja wezmę dzień urlopu i przyjadę do Wrocławia

Wybrane formaty danych

Dane bitowe

- Wszystkie dane są zapisywane i przesyłane jako dane bitowe
- Są mało czytelne
- Inne formaty są serializowane do danych bitowych, a następnie deserializowane

Dane bitowe...

- Przesyłanie czystych danych bitowych pomiędzy dwoma aplikacjami wymaga pełnej zgodności ich interfejsów
  - Jakakolwiek zmiana na różnych poziomach tylko w jednej z nich może spowodować problemy
  - W przypadku desynchronizacji wykrycie faktycznej przyczyny bywa bardzo problematyczne
  - Jest trudno audytowalny
  - Jest to wydajny sposób przenoszenia danych: brak narzutów serializacji / deserializacji

XML

- Uniwersalny język znaczników
- Pierwsza specyfikacja w 1998 roku
- Poprawność dokumentu
  - Poprawność składniowa: dokument jest zgodny z ogólnymi zasadami XML
  - Poprawność strukturalna: dokument jest zgodny z oddzielnie zdefiniowanym opisem struktury
    - XSD
    - DTD
    - Inne mniej popularne

XML...

- XSLT = XSL Transformations
  - Tłumaczenie, konwersja, filtrowanie dokumentów XML na inny dokument XML
- Dokumenty XML generalnie są czytelne dla użytkownika
- Wiele innych języków i protokołów bazuje na XML
  - SOAP
  - XHTML, XHTML5
  - WSDL

XML...

```
<widget>
    <debug>on</debug>
    <window title="Sample Konfabulator Widget">
        <name>main_window</name>
        <width>500</width>
        <height>500</height>
    </window>
    <image src="Images/Sun.png" name="sun1">
        <hOffset>250</hOffset>
        <vOffset>250</vOffset>
        <alignment>center</alignment>
    </image>
    <text data="Click Here" size="36" style="bold">
        <name>text1</name>
        <hOffset>250</hOffset>
        <vOffset>100</vOffset>
        <alignment>center</alignment>
        <onMouseUp>
            sun1.opacity = (sun1.opacity / 100) * 90;
        </onMouseUp>
    </text>
</widget>
```

JSON

- Lekki format danych wywodzący się z JavaScript
- Kwiecień 2001: przesłany pierwszy komunikat
- Tylko poprawność składniowa
- Zdecydowana większość współczesnych języków programowania wspiera / ma biblioteki wspierające ten format

JSON

- Obiekt JSON jest słownikiem
  - Kluczami są wartości tekstowe w cudzysłowiu
  - Wartością mogą być:
    - Tekst
    - Liczba
    - Stała (true, false, null)
    - Kolejny obiekt JSON
    - Tablica wartości
  - Teoretycznie dowolny poziom zagnieżdżenia

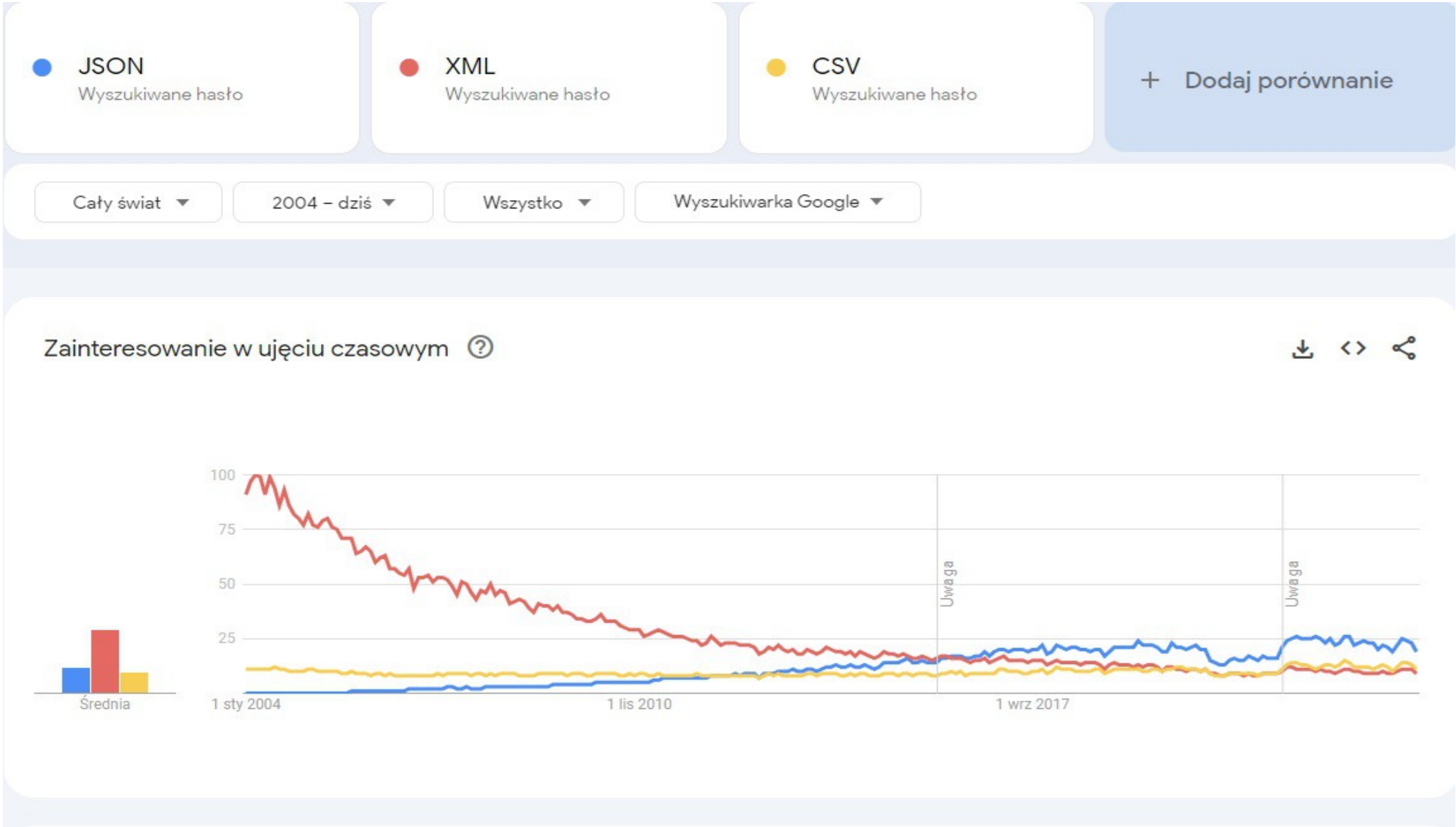
JSON...

```
{
  "widget": {
    "debug": "on",
    "window": {
      "title": "Sample Konfabulator Widget",
      "name": "main_window",
      "width": 500,
      "height": 500
    },
    "image": {
      "src": "Images/Sun.png",
      "name": "sun1",
      "hOffset": 250,
      "vOffset": 250,
      "alignment": "center"
    },
    "text": {
      "data": "Click Here",
      "size": 36,
      "style": "bold",
      "name": "text1",
      "hOffset": 250,
      "vOffset": 100,
      "alignment": "center",
      "onMouseUp": "sun1.opacity = (sun1.opacity / 100) * 90;"
    }
  }
}
```

JSON vs XML

- Dane w formacie JSON zajmują nieco mniej miejsca niż w formacie XML
- Serializacja / deserializacja jest szybsza i mniej złożona dla JSON
  - Nie ma potrzeby trzymania zawsze całego drzewa obiektu / dokumentu
- XML jest bardziej ustrukturyzowany
  - JSON Schema: 2012 – 2020 draft, opublikowany oficjalnie w 2021

JSON vs XML vs CVS



JSON serializacja / deserializacja

- Jest wiele bibliotek wspierających serializację i deserializację obiektów
  - Jackson
  - Gson
  - Wiele innych mniej popularnych

Jak działają te biblioteki

- Korzystają z refleksji
  - Przy serializacji poszukują get'terów
  - Przy deserializacji poszukują konstruktorów / set'terów
- W standardzie ich wyniki serializacji są podobne do wygenerowanej domyślnej metody toString()

Jakie są możliwości stworzenia dedykowanej serializacji / deserializacji

- Dużo rzeczy można zrobić poprzez adnotacje
  - Można zaznaczyć inne metody niż get'tery czy set'tery, które mają być użyte
  - Można stworzyć dedykowane mapowanie enum'ów
  - Można inaczej serializować obiekt w zależności od tego w jakim obiekcie się znajduje
  - Można serializować / deserializować datę w dedykowanych formatach
  - Można używać dedykowanej fabryki do deserializacji obiektu

Czy można stworzyć jeszcze bardziej dedykowaną serializację / deserializację

- Można stworzyć swój dedykowany Serializer / Deserializer
- Generalnie mamy wtedy praktycznie nieograniczone możliwości:
  - Możemy wczytywać dynamicznie konfigurację
  - Możemy mieć bardzo kontekstową serializację z zupełnie inną serializacją obiektu w różnych miejscach
  - Możemy obsłużyć serializację obiektów rekurencyjnych

Czy można stworzyć jeszcze bardziej dedykowaną serializację / deserializację

- Można stworzyć swój dedykowany Serializer / Deserializer
- Generalnie mamy wtedy praktycznie nieograniczone możliwości:
  - Możemy wczytywać dynamicznie konfigurację
  - Możemy mieć bardzo kontekstową serializację z zupełnie inną serializacją obiektu w różnych miejscach
  - Możemy obsłużyć serializację obiektów rekurencyjnych

Pracownia: co wygląda dobrze

- Dużo małych pull-request'ów
  - Znacznie szybciej przegląda się i zatwierdza
  - Mniej potencjalnych konfliktów
  - Łatwiej się wycofuje
- Testy, testy, dużo testów
  - Pokrycie testami będzie składnikiem oceny końcowej

Pracownia: co wygląda na ciężkie w utrzymaniu / niełądnie stylowo

- Wrzutki do repozytorium z pominięciem pull-request
  - Bywają projekty (zwłaszcza w dużych korporacjach), że coś takiego nie jest dostępne dla większości programistów
- Jeden duży pull-request w którym jest wszystko
  - Trudne do przejrzenia i zatwierdzenia
- Zakomentowany kod
- Import z *
  - Korporacyjny styl aby poprawić widoczność
  - Standardem jest wysoki limit klas zastępowany *

Pracownia

- Program ma umożliwić rozwiązywanie podstawowych zadań geometrycznych dla elipsy
- Nowe polecenie: usuń figurę z listy
- Figury mają być zapisywane do pliku w formacie json

Elipsa

- Charakterystyka figury:
  - Półoś wielka
  - Półoś mała
  - Pole powierzchni
- Możliwe wejście: (dowolna dwójka)
  - Półoś wielka
  - Półoś mała
  - Pole powierzchni
- Obwód elipsy liczymy pierwszym przybliżeniem całki eliptycznej