

# Bazy Danych i Usługi Sieciowe

XML

RPC

AJAX

Paweł Daniluk

Wydział Fizyki

Jesień 2013



# Semistrukuralny model danych

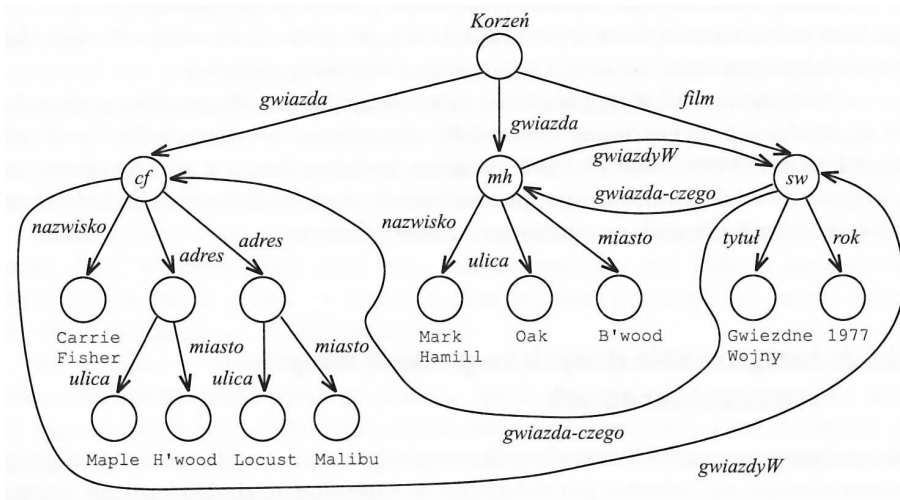
Nie zawsze jest potrzeba albo możliwość posiadania ściśle zdefiniowanego schematu danych.

# Semistrukuralny model danych

Nie zawsze jest potrzeba albo możliwość posiadania ściśle zdefiniowanego schematu danych.

W modelu semistrukuralnym dane określają schemat.

# Semistrukuralny model danych – przykład



# Semistrukuralny model danych c.d.

## Zalety

- Swoboda w rozszerzaniu i adaptowaniu do nowych potrzeb.
- Brak ograniczeń wynikających z modelu danych.

# Semistrukuralny model danych c.d.

## Zalety

- Swoboda w rozszerzaniu i adaptowaniu do nowych potrzeb.
- Brak ograniczeń wynikających z modelu danych.

## Wady

- Trudność w formułowaniu zapytań.
- Brak wydajnych implementacji pozwalających na przechowywanie dużych ilości danych.

# Semistrukuralny model danych c.d.

## Zalety

- Swoboda w rozszerzaniu i adaptowaniu do nowych potrzeb.
- Brak ograniczeń wynikających z modelu danych.

## Wady

- Trudność w formułowaniu zapytań.
- Brak wydajnych implementacji pozwalających na przechowywanie dużych ilości danych.

## Zastosowania

- Integracja danych pochodzących z różnych źródeł.
- Adaptacja “starych” baz danych do nowych potrzeb.
- Opis dokumentów.

## eXtensible Markup Language

### Znaczniki

- określają znaczenie podciągów znaków w dokumencie
- teksty ujęte w nawiasy kątowe <...>
- występują w parach – otwierający <...> i zamykający </...>

### Zastosowania

- Przechowywanie ustrukturyzowanych danych w plikach tekstowych
- Wymiana danych pomiędzy aplikacjami



# Tryby dokumentów XML

## Dobrze sformowany XML

- Podejście semistrukturalne
- Dowolne znaczniki
- Brak ustalonego schematu

## Ustalony typ dokumentu

- Podejście pośrednie pomiędzy schematem semistrukturalnym, a ścisłymi (np. relacyjnym)
- **Document Type Definition**
- Specyfikacja dopuszczalnych znaczników
- Gramatyka zagnieżdżania

# Dobrze sformowany XML

## Przykład

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<etat>
  <etat>
    <nazwa>Profesor</nazwa>
    <placa_od>3000</placa_od>
    <placa_do>6000</placa_do>
  </etat>
  <etat>
    ...
  </etat>
</etat>
```

# Dokument ustalony

## Dokument bez specyfikacji typu

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

## Nagłówek

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>  
<!DOCTYPE typDokumentu SYSTEM "specyfikacja.dtd">
```

# Dokument ustalony c.d.

## Przykład

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<!DOCTYPE Etaty SYSTEM "etaty.dtd">
<etaty>
  <etat>
    <nazwa>Profesor</nazwa>
    <placa_od>3000</placa_od>
    <placa_do>6000</placa_do>
  </etat>
  <etat>
    ...
  </etat>
</etaty>
```

# Dokument ustalony c.d.

- Deklaracja elementu głównego (korzenia)
- Deklaracje elementów
- Deklaracje reguł zagnieżdżania

## etaty.dtd

```
<!DOCTYPE etaty [  
  <!ELEMENT etaty (etat*)>  
  <!ELEMENT etat (nazwa, placa_od, placa_do)>  
  <!ELEMENT nazwa (#PCDATA)>  
  <!ELEMENT placa_od (#PCDATA)>  
  <!ELEMENT placa_do (#PCDATA)>  
>
```

## Dokument ustalony c.d.

- Nazwa typu dokumentu: `<!DOCTYPE [...]>`
- Nazwy dopuszczalnych elementów `<!ELEMENT (...)>`
- Reguły zagnieżdżania dopuszczalnych składowych elementów
- element (podelement1\*, podelement2+, podelement3?,...)
- Operatory:
  - ▶ \*: 0 lub więcej
  - ▶ +: 1 lub więcej
  - ▶ ?: co najwyżej raz
- element (#PCDATA)
- Typ #PCDATA oznacza dowolny tekst
- Nie występują typy elementów

# Przykład

## Model relacyjny

Gwiazdy(nazwisko, adres)

Filmy(tytul, rok, dlugosc)

GwiazdyW(tytul, rok, nazwiskoGwiazdy)

## Dokument DTD

```
<!DOCTYPE Gwiazdy [  
    <!ELEMENT gwiazdy (gwiazda*)>  
    <!ELEMENT gwiazda (nazwisko, adres+, filmy)>  
    <!ELEMENT nazwisko (#PCDATA)>  
    <!ELEMENT adres (#PCDATA)>  
    <!ELEMENT filmy (film*)>  
    <!ELEMENT film (tytul, rok, dlugosc)>  
    <!ELEMENT tytul (#PCDATA)>  
    <!ELEMENT rok (#PCDATA)>  
    <!ELEMENT dlugosc (#PCDATA)>  
]>
```

## Przykładowe dane

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<!DOCTYPE Gwiazdy SYSTEM "gwiazdy.dtd">
<gwiazdy>
  <gwiazda>
    <nazwisko>Carrie Fischer</nazwisko>
    <adres>123 Maple St.</adres>
    <filmy>
      <film>
        <tytul>Gwiezdne Wojny</tytul>
        <rok>1977</rok>
        <dlugosc>93</dlugosc>
      </film>
      <film>
        <tytul>Imperium Kontratakuje</tytul>
        <rok>1980</rok>
        <dlugosc>96</dlugosc>
      </film>
    </filmy>
  </gwiazda>
</gwiazdy>
```



# Przykładowe dane

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<!DOCTYPE Gwiazdy SYSTEM "gwiazdy.dtd">
<gwiazdy>
  <gwiazda>
    <nazwisko>Carrie Fischer</nazwisko>
    <adres>123 Maple St.</adres>
    <filmy>
      <film>
        <tytul>Gwiezdne Wojny</tytul>
        <rok>1977</rok>
        <dlugosc>93</dlugosc>
      </film>
      <film>
        <tytul>Imperium Kontratakuje</tytul>
        <rok>1980</rok>
        <dlugosc>96</dlugosc>
      </film>
      <film>
        <tytul>Powrót Jedi</tytul>
        <rok>1983</rok>
        <dlugosc>94</dlugosc>
      </film>
    </filmy>
  </gwiazda>
  <gwiazda>
    <nazwisko>Mark Hamill</nazwisko>
    <adres>456 Oak Rd.</adres>
    <adres>789 Pine Av.</adres>
    <filmy>
      <film>
        <tytul>Imperium Kontratakuje</tytul>
        <rok>1980</rok>
        <dlugosc>96</dlugosc>
```

## Dokument ustalony c.d.

- Zagnieżdżanie znaczników nie pozwala przedstawić wszystkich informacji
- Atrybuty pozwalają na dodatkowy opis danych
- Zmniejszenie redundancji
- Mogą służyć do powiązania pojedynczej wartości ze znacznikiem
- Alternatywa dla podznaczników, które są zwykłymi tekstami (#PCDATA)

## Składnia

- Po deklaracji elementu `<!ELEMENT (...)>`
- `<!ATTLIST element ...>`
- lista atrybutów
  - ▶ atrybut1
  - ▶ atrybut2 ID
  - ▶ atrybut3 IDREF lub IDREFS

# Atrybuty – przykład

## Dokument DTD

```
<!DOCTYPE Gwiazdy-Filmy [  
  <!ELEMENT gwiazdy-filmy (gwiazda*, film*)>  
  <!ELEMENT gwiazda (nazwisko, adres+)>  
    <!ATTLIST gwiazda  
      gwiazdaId ID  
      wystepujeW IDREFS>  
  <!ELEMENT nazwisko (#PCDATA)>  
  <!ELEMENT adres (ulica, miasto)>  
  <!ELEMENT ulica (#PCDATA)>  
  <!ELEMENT miasto (#PCDATA)>  
  <!ELEMENT film (tytul, rok, dlugosc)>  
    <!ATTLIST film  
      filmId ID  
      gwiazdyW IDREFS>  
  <!ELEMENT tytul (#PCDATA)>  
  <!ELEMENT rok (#PCDATA)>  
  <!ELEMENT dlugosc (#PCDATA)>  
>
```

# Atrybuty – przykład

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<!DOCTYPE Gwiazdy-Filmy SYSTEM "gwiazdy-filmy.dtd">
<gwiazdy-filmy>
  <gwiazda gwiazdaId="cf" wystepujeW="gw, ik, pj">
    <nazwisko>Carrie Fischer</nazwisko>
    <adres>
      <ulica>123 Maple St.</ulica>
      <miasto>Hollywood</miasto>
    </adres>
  </gwiazda>
  <gwiazda gwiazdaId="mh" wystepujeW="ik, pj">
    <nazwisko>Mark Hamill</nazwa>
    <adres>
      <ulica>456 Oak Rd.</ulica>
      <miasto>Malibu</miasto>
    </adres>
    <adres>
      <ulica>123 Pine Av.</ulica>
      <miasto>Brentwood</miasto>
    </adres>
  </gwiazda>
```

# Atrybuty – przykład

```
<film filmId="gw" gwiazdyW="cf">
  <tytul>Gwiezdne Wojny</tytul>
  <rok>1977</rok>
  <dlugosc>93</dlugosc>
</film>
<film filmId="ik" gwiazdyW="cf, mh">
  <tytul>Gwiezdne Wojny</tytul>
  <rok>1980</rok>
  <dlugosc>96</dlugosc>
</film>
<film filmId="pj" gwiazdyW="cf, mh">
  <tytul>Powrót Jedi</tytul>
  <rok>1983</rok>
  <dlugosc>94</dlugosc>
</film>
</gwiazdy-filmy>
```

# Pokrewne standardy i specyfikacje

- XML Namespaces
- XML Base
- The XML Information Set
- xml:id
- XPath
- XSLT
- XSL Formatting Objects
- XQuery
- XML Signature
- XML Encryption

API dla praktycznie wszystkich języków programowania.

- **Remote Procedure Call**
- Protokół zdalnego wywoływania procedur
- Korzenie sięgają 1976
- Pod UNIX implementacja Sun (na bazie NFS)
- Proces w systemie lokalnym wywołuje procedurę w systemie zdalnym
- Tryb żądanie-odpowiedź
- Możliwe wykorzystanie różnych protokołów sieciowych
- Ukrycie operacji sieciowych

- 1 Serwer nasłuchuje na wybranym porcie
- 2 Klient nawiązuje z nim łączność poprzez sieć
- 3 Klient wysyła swoje dane we wcześniej ustalonym przez programistów klienta i serwera formacie
- 4 Serwer realizuje usługę i odsyła potwierdzenie lub kod błędu
- 5 Z punktu widzenia użytkownika w zdalne wywołanie procedury nie różni się od wywołania procedury lokalnej



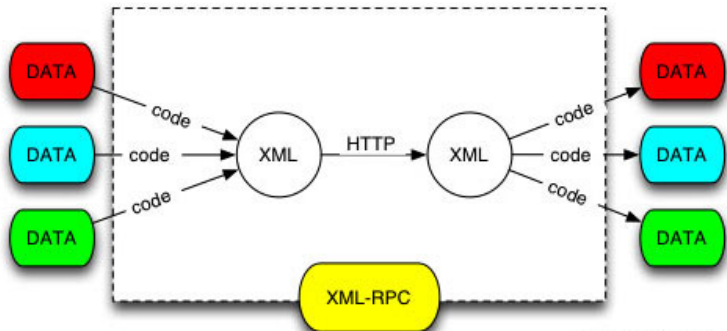
# Schemat działania RPC

- 1 Opakowanie argumentów po stronie klienta (łącznik klienta)
- 2 Przesłanie argumentów zgodnie z używanym protokołem sieciowym
- 3 Rozpakowanie argumentów po stronie serwera (łącznik serwera) do postaci właściwej dla systemu serwera
- 4 Wywołanie procedury lokalnej na serwerze z argumentami
- 5 Przekazanie wyniku do łącznika serwera
- 6 Opakowanie wyniku po stronie serwera (łącznik serwera)
- 7 Przesłanie do łącznika klienta
- 8 Przekształcenie do postaci właściwej dla systemu klienta
- 9 Przekazanie sterowania do miejsca, w którym został wywołany łącznik klienta

- Klient i serwer mają różne środowiska (systemy operacyjne)
- Serializacja danych do ustalonego formatu
- Programy łącznika klienta i łącznika serwera zapewniają komunikację
- W modelu OSI obsługa RPC jest znajduje się między warstwą transportową a procesów użytkownika - w warstwie prezentacji
- Obecnie RPC jest wypierane przez nowsze protokoły
  - ▶ XML-RPC
  - ▶ JSON-RPC

- Format XML nadaje się do automatycznego przetwarzania
- Umożliwia ustandaryzowanie sposobu wymiany danych
- Specyfikacja XML-RPC : Dave Winer, UserLand Software, 1996
  - ▶ Niezależne od oprogramowania (PHP, Python, Java, C++, ...)
  - ▶ Zgodne z protokołem sieciowym (HTTP, HTTPS)
  - ▶ Ścisłe określenie formatu żądania i odpowiedzi

# XML-RPC



Source: JY Stervinou

# XML-RPC

- Format XML nadaje się do automatycznego przetwarzania
- Umożliwia ustandaryzowanie sposobu wymiany danych
- Specyfikacja XML-RPC
- Dave Winer, UserLand Software, 1996
- Odwołanie do udostępnianej przez serwer funkcji za pomocą żądania HTTP POST
- Przekazanie nazwy procedury (metody) i argumentów
- Odebranie wyniku lub komunikatu o błędzie przez HTTP

# Wywołanie XML-RPC

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181
```

```
<?xml version="1.0"?>
<methodCall>
  <methodName>examples.getStateName</methodName>
  <params>
    <param>
      <value> <int>42</int> </value>
    </param>
  </params>
</methodCall>
```

# Parametry wywołania

- Elementy `<param>` w elemencie `<params>`
- Parametry są nienazwane
- Ma znaczenie kolejność parametrów
- Trzy typy parametrów:
  - ▶ Skalary
  - ▶ Tablice
  - ▶ Struktury

# Odpowiedź XML-RPC

## Sukces

HTTP/1.1 200 OK

Connection: close

Content-Length: 158

Content-Type: text/xml

Date: Fri, 17 Jul 1998 19:55:08 GMT

Server: UserLand Frontier/5.1.2-WinNT

```
<?xml version="1.0"?>
  <methodResponse>
    <params>
      <param>
        <value><string>South Dakota</string></value>
      </param>
    </params>
  </methodResponse>
```



# Błąd XML-RPC

## Błąd

```
<fault>
  <value>
    <struct>
      <member>
        <name>faultCode</name>
        <value><int>4</int></value>
      </member>
      <member>
        <name>faultString</name>
        <value><string>Too many parameters.</string></v
      </member>
    </struct>
  </value>
</fault>
```

# Wywołanie XML-RPC

- Serializacja danych do formatu XML-RPC
- Utworzenie nagłówków HTTP i żądania XML-RPC
- Wykonanie żądania
- Odebranie odpowiedzi
- Parsowanie odpowiedzi

# Implementacje XML-RPC

- Perl
- Python
- C, C++
- Java
- PHP
- Ruby

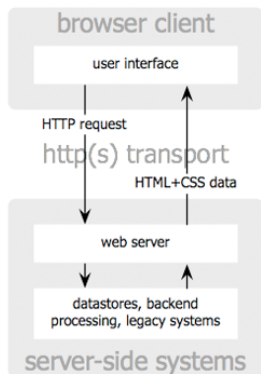
# XML-RPC a SOAP

- SOAP jest również protokołem usług sieciowych
- Wymiana danych w formacie XML
- Daje większe możliwości niż XML-RPC
- Zalecana znajomość dodatkowych standardów:
  - ▶ WSDL (Web Services Description Language) do opisu usług sieciowych
  - ▶ XSD (XML Schema Data) do opisu typów danych

## Asynchronous JavaScript and XML

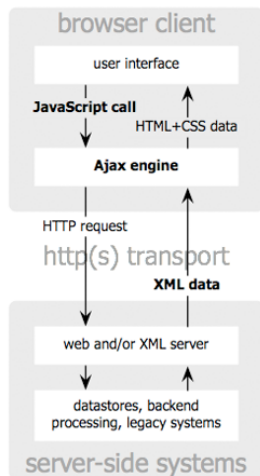
- prezentacja – XHTML i CSS
- dynamiczne modyfikacje – Document Object Model
- wymiana i manipulacja danymi – XML and XSLT
- komunikacja asynchroniczna – XMLHttpRequest
- JavaScript

# AJAX c.d.



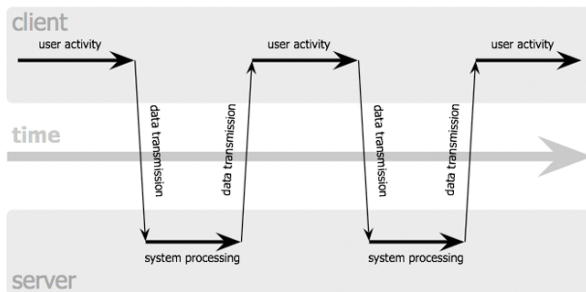
**classic  
web application model**

Jesse James Garrett / adaptivepath.com

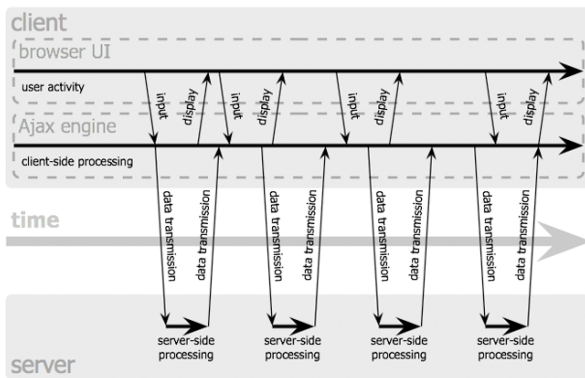


**Ajax  
web application model**

## classic web application model (synchronous)



## Ajax web application model (asynchronous)



Jesse James Garrett / adaptivepath.com



[http://bioexploratorium.pl/wiki/  
Bazy\\_danych\\_i\\_uslugi\\_sieciowe\\_-\\_2013z](http://bioexploratorium.pl/wiki/Bazy_danych_i_uslugi_sieciowe_-_2013z)