Testowanie automatyczne

Web Service

HTTP - **H**yper **T**ext **T**ransfer **P**rotocol

Protokół przesyłania dokumentów hipertekstowych (z dodatkowymi znacznikami)

HTTPS - odmiana szyfrowana

HTTP składa się (w dużym uproszczeniu)

- zapytania
  - metody http (GET, POST, HEAD, …)
  - nagłówków
  - parametrów
- odpowiedzi
  - statusu
  - nagłówków
  - body - treści odpowiedzi

HTTP - Kody Odpowiedzi

100 Kody informacyjne	100	Continue
	111	Connection refused
200 Wszystko ok	200	Standard success response
	201	Created
	202	Accepted
	203	Non-authoritative
300 Kody przekierowania	301	Moved permanently
	304	Not modified
	307	Temporary redirect
	310	Too many redirections

HTTP - Kody Odpowiedzi

400 Błąd po stronie klienta	401	Not authorized
	403	Forbidden
	404	Not found
	405	Method not allowed
500 Błąd po stronie serwera	500	Server error
	502	Bad gateway
	504	Gateway timed out

API - Application Programming Interface

Zestaw reguł i ich opisów wg których programy ze sobą powinny się komunikować

Przykłady

- Windows API - interfejs umożliwiający programom Windows wyświetlanie okien, przycisków, interakcję z systemem, itp.
- DirectX - interfejs umożliwiający wyświetlanie grafiki w Windows (gry)
- Selenium - zestaw funkcji WebDriver
- Google Maps API - umieszczenie mapy na własnej stronie internetowej

JSON - Java Script Object Notation

Tekstowy format wymiany danych komputerowych

- Pomimo nazwy jest niezależny od języka programowania

```
{
  "colors": [
    {
      "color": "black",
      "category": "hue",
      "type": "primary",
      "code": {
        "rgba": [255,255,255,1],
        "hex": "#000"
      }
    } ]
}
```

Web Service

Web Service - usługa sieciowa do komunikacji pomiędzy aplikacjami klienckimi i serwerowymi w sieci World Wide Web

Sposoby wymiany informacji

- SOAP - Simple Object Access Protocol
- WSDL - Web Services Description Language
- REST - REpresentational State Transfer

Inne serwisy?

Web serwisy są oparte na protokole HTTP. Oznacza to, że są wspierane przez

Istnieje grupa usług sieciowych opartych na innych protokołach

- CORBA
- DCOM
- ESB
- RPC
- Inne...

Sprawdzają się w bardzo specjalizowanych zastosowaniach (transakcje giełdowe, zbieranie danych z liczników, …), wymagają jednak zaawansowanego zarządzania siecią komputerową i odpowiedniej konfiguracji np. zapór sieciowych - firewall

REST - **RE**presentational **S**tate **T**ransfer

Zestaw instrukcji (API) umożliwiających dostęp do aplikacji webowych.

Inaczej - bezpośrednie wykonanie komend i poleceń bez interakcji z interfejsem użytkownika (stroną html).

Web Service - usługa sieciowa do komunikacji pomiędzy aplikacjami klienckimi i serwerowymi w sieci World Wide Web

RESTful - główne elementy

1. Resources
   Zasób, element
2. Request Verbs
   Akcja
3. Request Headers
   Nagłówki, dodatkowe instrukcje
4. Request Body
   Dane wysyłane z requestem (żądaniem do serwisu)
5. Response Body
   Dane zwracane z odpowiedzią (np. dokument xml lub json)
6. Response Status Codes
   Status Code odpowiedzi

http://katalog.biblioteka.com/authors
http://katalog.biblioteka.com/books/1

RESTful - zasób

- Co chcemy dostać

GET http://trafficlights.com/roads

roads - lista dróg zdefiniowanych w danym systemie

GET http://trafficlights.com/signalizers

signalizers - lista sygnalizatorów (świateł drogowych)

```
roads: [
 {
  "id": "1",
  "name": "N"
 },
 {
  "id": "2",
  "name": "S"
 },
 {
  "id": "3",
  "name": "W"
 },
 {
  "id": "4",
  "name": "E"
 } ]
```

RESTful - zasób

- Co chcemy dostać

GET http://trafficlights.com/roads/1

roads/1 - wybrana droga do wyświetlenia

```
{
  "id": "1",
  "name": "N"
}
```

GET http://trafficlights.com/signalizers/12

signalizers/1 - wybrany sygnalizator

```
{
  "id": "12",
  "name": "Car 1",
  "state": "Red",
  "road": "1"
},
```

RESTful - verb (akcja)

- Co chcemy zrobić

GET - Pobierz dane

POST - Utwórz dane

PUT - Zmodyfikuj istniejący element

DELETE - Usuń wskazany element

- akcje są mapowane na metody http
- parametry są przekazywane jako:
  - element adresu (GET)
  - parametr requestu (POST)
  - body requestu (POST)
  - nagłówki (wszystkie)
- nie wszystkie akcje są dozwolone - kontrola należy do serwisu, z którym się komunikujemy

RESTful - Odpowiedzi

W odpowiedzi otrzymujemy

- status odpowiedzi
- treść odpowiedzi (np. dokument json)
- walidację
- status operacji (np. czy udało nam się zmodyfikować element)

Kody HTTP są ustandaryzowane

Treść odpowiedzi REST Api zależy od implementacji - dokumentacja serwera

RESTful - nagłówki

Nagłówki zawierają meta-informacje o żądaniu

- typ akceptowanej odpowiedzi
- dodatkowe parametry
- dane autoryzacyjne

Content-Type application/json
Last-Modified Tue, 10 May 2019 10:14:26 GMT
Set-Cookie Set-Cookie: UserID=AdamAdamowski; SentAgreement=1

RESTful - co możemy testować?

Wszystko, co otrzymamy w odpowiedzi

- status code
- treść odpowiedzi
- czy zadana akcja została wykonana prawidłowo
- czy otrzymaliśmy właściwe nagłówki
- przekierowanie (powiązane z treścią nagłówka i odpowiednim kodem)
- czy kody błędów mają dodatkową informację w treści
- autoryzację dostępu

Uwierzytelnianie

Basic Auth

Dodajemy nagłówek Authorize, który w treści ma

“Basic base64({user}:{password})”

OAuth2

Posługujemy się tokenem, poświadczonym przez zaufaną stronę

np. logowanie Google, logowanie Facebook

Wersjonowanie - różne wersje API

Poprzez URL

https://gitlab.com/api/v4/projects

+ łatwo rozpoznać po adresie
- trzeba zmienić wszystkie odwołania przy zmianie

Poprzez nagłówek zdefiniowany

https://service.com/api/fancydata

MyHeader v1.1

+ nie ma potrzeby modyfikacji adresów
- niewidoczny na pierwszy rzut oka
- niestandardowy

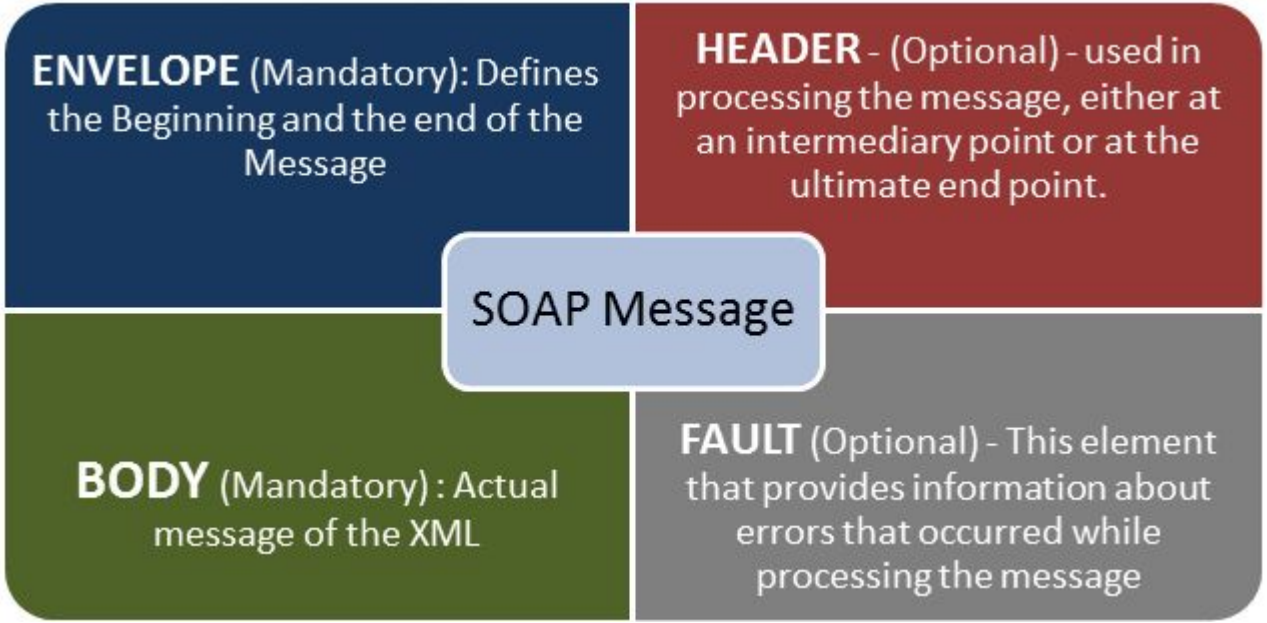
Poprzez nagłówek Accept

https://service.com/api/fancydata

Accept application/service.v1.1+json

+ nie ma potrzeby modyfikacji adresów
- niewidoczny na pierwszy rzut oka

SOAP - Simple Object Access Protocol



Protokół SOAP to o wiele bardziej formalny sposób komunikacji.

Definiuje obowiązkowe elementy

Wymaga zdefiniowania WSDL - dokumentu stanowiącego kontrakt dla Web Service

SOAP vs REST

REST	SOAP
Styl architektoniczny opisujący format	Protokół komunikacji
HTTP	HTTP, UDP, SMTP
JSON, YAML, Tekst, HTML	XML
Bezstanowy	Stanowy, bezstanowy
Ma luźną formę; tylko od twórców aplikacji zależy jakość dokumentacji	Wymaga kontraktu WSDL
Szybszy i bardziej zwięzły	Ze względu na XML - obszerniejszy (o ok. 20%) i wolniejszy w obsłudze
SSL, OAuth	SSL, OAuth, WS-Security, sygnalizuje błędy integracji - poprzez kontrakt

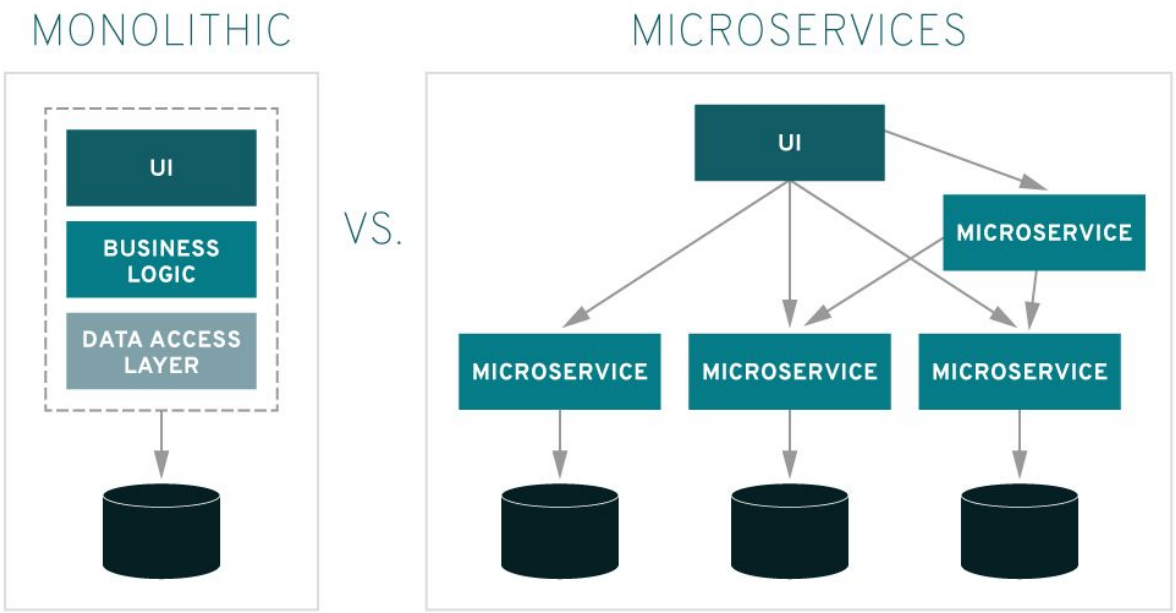
WSDL - kontrakt do wypełnienia

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" … >
 <wsdl:types>
  <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">

    <s:element name="Add">

     <s:complexType>

      <s:sequence>
       <s:element minOccurs="1" maxOccurs="1" name="intA" type="s:int" />
       <s:element minOccurs="1" maxOccurs="1" name="intB" type="s:int" />
      </s:sequence>

     </s:complexType>
    </s:element>

    <s:element name="AddResponse">
     <s:complexType>

      <s:sequence>
       <s:element minOccurs="1" maxOccurs="1" name="AddResult" type="s:int" />
      </s:sequence>
     …
</wsdl:types>
```

WSDL - kontrakt do wypełnienia

```
<wsdl:message name="AddSoapIn">
 <wsdl:part name="parameters" element="tns:Add" /> </wsdl:message>
<wsdl:message name="AddSoapOut">
 <wsdl:part name="parameters" element="tns:AddResponse" /> </wsdl:message>
...
<wsdl:portType name="CalculatorSoap">
 <wsdl:operation name="Add">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Adds two integers. [...] ©DNE Online</wsdl:documentation>
  <wsdl:input message="tns:AddSoapIn" /> <wsdl:output message="tns:AddSoapOut" />
 </wsdl:operation>
...
<wsdl:binding name="CalculatorSoap" type="tns:CalculatorSoap">
 <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
 <wsdl:operation name="Add">
  <soap:operation soapAction="http://tempuri.org/Add" style="document" />
  <wsdl:input> <soap:body use="literal" /> </wsdl:input>
  <wsdl:output> <soap:body use="literal" /></wsdl:output>
 </wsdl:operation>
....
<wsdl:service name="Calculator">
  <wsdl:port name="CalculatorSoap" binding="tns:CalculatorSoap">
   <soap:address location="http://www.dneonline.com/calculator.asmx" />
  </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Micro services



5 zasad microservice'ów:

1. Rozbij większą całość na elementy
2. Każdy element odpowiada za jedną funkcjonalność
3. Komunikacja np. REST Api
4. Każdy z mikroserwisów może być rozwijany oddzielnie
5. Każdy z serwisów działa na odrębnej infrastrukturze

Każdy z mikroserwisów może być testowany jako odrębny produkt

Microservices - przykład architektury