Webové služby

KAMILLA SABIRZYANOVA, KRISTÝNA ŠEDOVÁ,
VERONIKA HENDRYCHOVÁ

Plán:

- HTTP protokol
- SOAP
- Technologie a nástroje pro webové služby

HTTP PROTOKOL

HYPERTEXT TRANSFER PROTOCOL

Základy HTTP protokolu

Co je HTTP?

HTTP je aplikační protokol používaný pro výměnu informací na internetu. Podporuje model klient-server.

Hlavní charakteristiky

- Jednoduchost
- Rozšiřitelnost pomocí hlaviček
- Bezstavovost (každý požadavek je nezávislý).

Využití

HTTP je základní stavební jednotkou World Wide Webu, umožnuje přenos dat, jako jsou texty, obrázky a videa.

Vývoj verzí HTTP

- **HTTP-0.9** (1991): první verze zaměřená pouze na přenos jednoduchého textu, bez hlaviček nebo dalších metadat.
- **HTTP/1.1** (1997): první standardizovaná verze. Přidává persistentní připojení a další vylepšení.
- **HTTP/2** (2015): binární protokol, komprese hlaviček, paralelní dotazy.
- **HTTP/3** (nové): využívá QUIC místo TCP, zlepšuje latenci a bezpečnost.

HTTP Komunikace
Model Klient Server

- **Požadavek klienta** (request) **:** Klient odesílá HTTP zprávu s metodou (např. GET) a URL zdroje.
- **Odpověd servetu** (response)**:** Server reaguje zprávou obsahující stavový kod (např. 200 OK) a data.
- **Bezstavovost:** Každý požadavek je nezávislý, bez sdílení stavu mezi transakcemi.

Požadavek klienta

1. **Metoda** (GET, POST, apod.).
2. **Adresa** požadovaného zdroje (URL).
3. **Verze** HTTP protokolu.
4. **Hlavičky** s dalšími informacemi.
5. **Tělo** (volitelné, obsahuje data).

GET /index.html HTTP/1.1 Host: www.example.com User-Agent: Mozilla/5.0

Metody HTTP

Hlavní metody:

- **GET**: získání dat.
- **POST**: odesílání dat.
- **PUT**: aktualizace nebo vytvoření zdroje.
- **DELETE**: smazání zdroje.

Speciální metody:

- **OPTIONS**: zjistí podporované metody.
- **PATCH**: částečná aktualizace.

Vlastnosti HTTP metod

- **Bezpečné** metody
- **Idempotentní** metody
- **"Cacheable"** metody

Odpověď serveru

1. **Stavový řádek** (HTTP verze, kód, popis).
2. **Hlavičky** (informace o odpovědi).
3. **Tělo** (volitelný obsah dat).

HTTP/1.1 200 OK Content-Type: text/html Content-Length: 123

Stavové kódy

Kategorie stavových kódů:

- **1xx**: Informace (např. 100 Continue).
- **2xx**: Úspěch (např. 200 OK).
- **3xx**: Přesměrování (např. 301 Moved Permanently).
- **4xx**: Chyba klienta (např. 404 Not Found).
- **5xx**: Chyba serveru (např. 500 Internal Server Error).

Kód	Název	Popis
200	OK	Nejběžnější odpověď, signalizuje úspěšné provedení požadavku.
201	Created	Značí úspěšné vytvoření nového zdroje na serveru.
202	Accepted	Požadavek byl přijat, ale zatím nebyl celý zpracován.
204	No content	Požadavek se úspěšně provedl, ale server nevrací žádná data.
301	Moved Permanently	Zdroj byl trvale přesunut na URL definovanou v hlavičce *Location*.
302	Found	Zdroj byl dočasně přesunut na URL definovanou v hlavičce *Location*.
304	Not Modified	Signalizuje, že není třeba přenášet zdroj jelikož nedošlo od minulého požadavku k jeho změně.
400	Bad Request	Server není schopen zpracovat požadavek, kvůli chybě na straně klienta (např. požadavek má špatnou syntaxi).
401	Unauthorized	Pro úspěšné provedení požadavku se musí klient autentizovat. Používá se s hlavičkou *WWW-Authenticate*.
403	Forbidden	Kliet nemá potřebná práva na provedení požadavku.
404	Not Found	Server nebyl schopen nalézt požadovaný zdroj.
405	Method Not Allowed	Daná metoda není povolena pro dotazovaný zdroj. Odpověď by měla obsahovat hlavičku *Allow* s podporovanými metodami.
408	Request Timeout	Server neobdržel celý požadavek po dobu, kterou byl připraven čekat a ukončuje spojení.
415	Unsupported Media Type	Data v těle požadavku jsou zaslána v nepodporovaném formátu.
500	Internal Server Error	Na serveru vznikla neočekávána chyba při zpracování požadavku. Jedná se o obecný chybový stav.
503	Service Unavailable	Signalizuje neschopnost zpracovat požadavek z důvodu přetížení či údržby serveru.
504	Gateway Timeout	Server fungující jako brána nebo proxy nedostal včas od cílového serveru odpověď a proto nemůže dokončit požadavek.

Popis vybraných stavových HTTP kódů

HTTP hlavičky

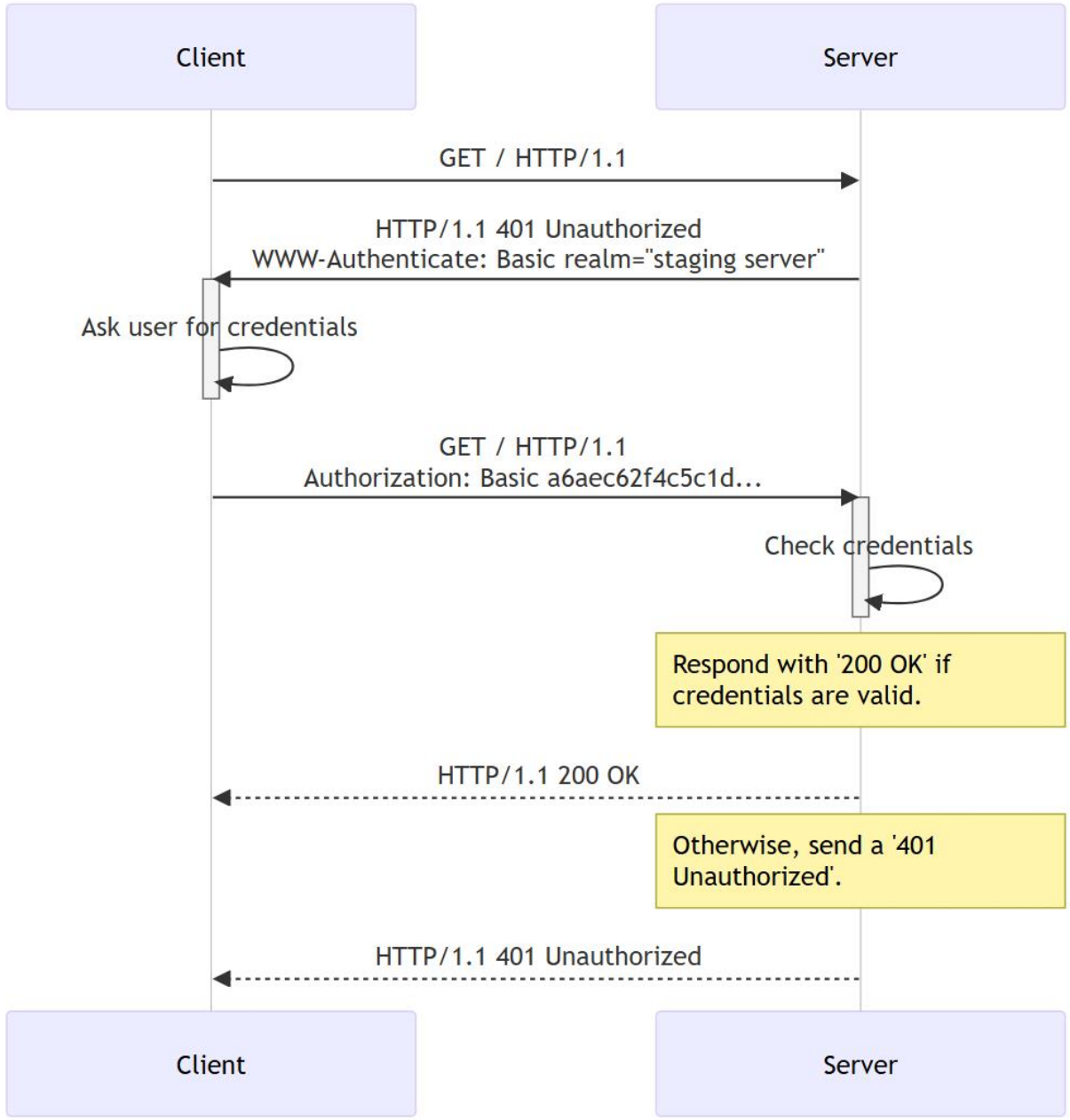
Typy hlaviček:

1. **Obecné** (Date, Connection).
2. **Požadavku** (Accept, User-Agent).
3. **Odpovědi** (Content-Type, Server).
4. **Popisující zdroje** (Content-Length, Content-Encoding).

Bezpečnost v HTTP

- **HTTPS**: šifrovaná varianta HTTP pomocí TLS.
- **Autentizace**:
    - Basic: jednoduchá, kóduje uživatelské jméno a heslo.
    - Bearer: tokeny, často využívá OAuth 2.0.

HTTP autentizace



Shrnutí

- HTTP je základní protokol pro webovou komunikaci.
- Klíčové koncepty: požadavky, odpovědi, metody, hlavičky, stavové kódy.
- Význam bezpečnosti (HTTPS).

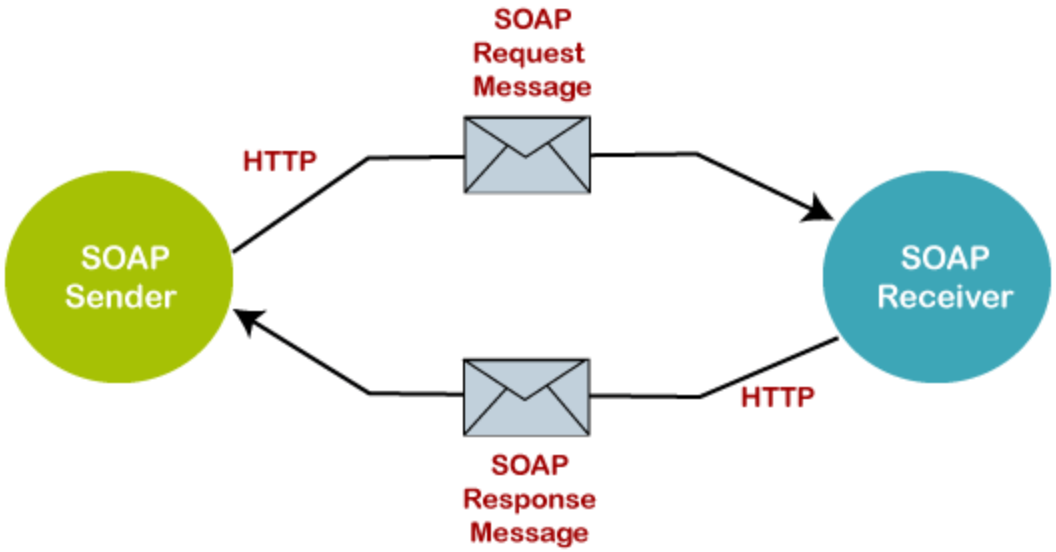
SOAP

SIMPLE OBJECT ACCESS PROTOCOL

SOAP

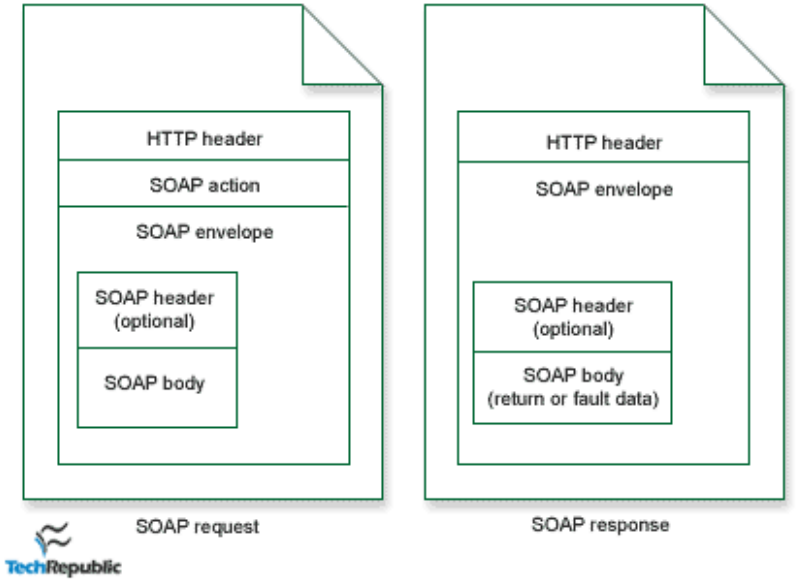
- Definuje podobu zpráv, které si aplikace zasílají
- Zprávy jsou zapisovány ve formátu XML
- Nejčastěji používá protokol HTTP
- Nezávislý na OS a platformě
- První verze, která získala doporučení od W3C,

je verze 1.2 vydaná v roce 2003



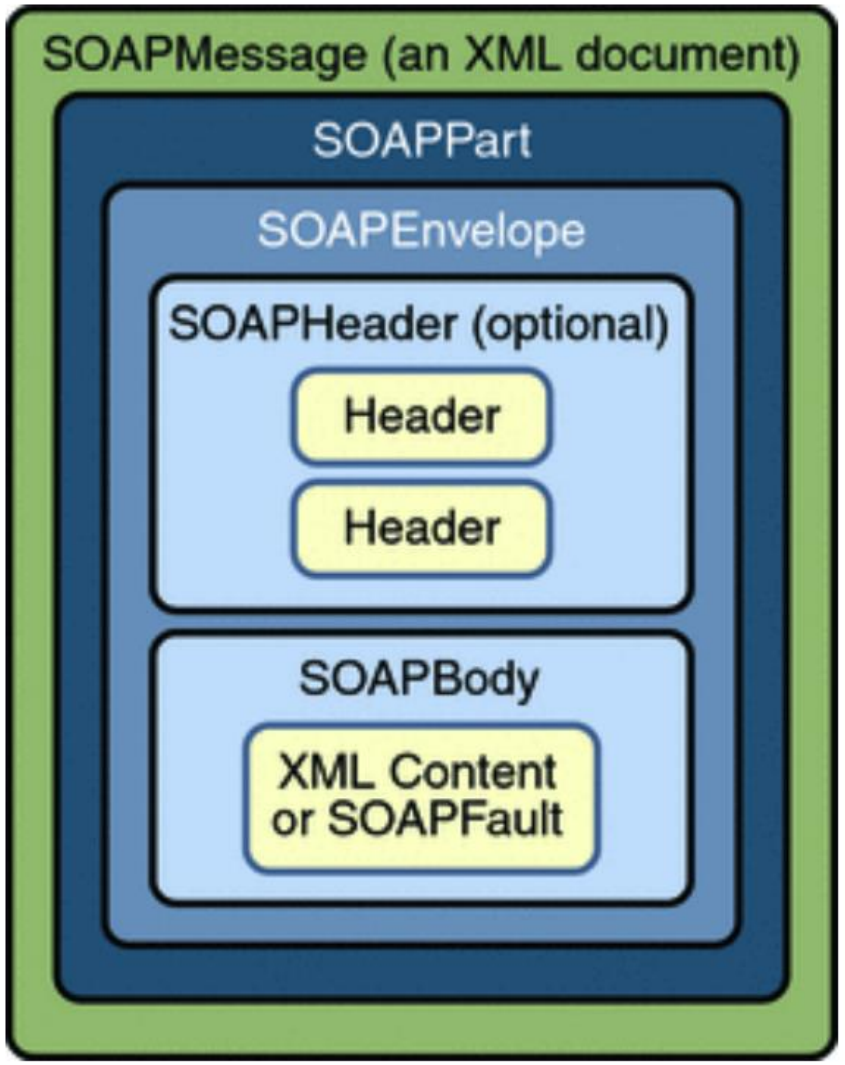
- World Wide Web Consortium
- Mezinárodní organizace, která se zabývá vývojem a standardizací technologií pro web.
- Byla založena v roce 1994
- Jejím hlavním cílem je zajistit, aby byl web přístupný, efektivní, bezpečný pro všechny.

SOAP a HTTP



- HTTP je nejběžněji používaným přenosovým protokolem pro SOAP (SOAP využívá pouze malou část HTTP)
- Z pohledu SOAP lze o HTTP mluvit jako o přenosové vrstvě
- Požadavek obsahuje povinou hlavničku Host a SOAPAction (identifikuje SOAP požadavek)

Formát zpráv



- XML zprávy jsou textové
- Elementy ve jmenném prostoru: obálka (envelope), hlavička (header), tělo (body), chyba (fault)

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

Hlavička

- Nepovinný element
- Musí být uveden jako první
- Skládá se z jednotlivých záznamů

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Header>
  <m:Trans xmlns:m="https://www.w3schools.com/transaction/"
  soap:mustUnderstand="1">234
  </m:Trans>
</soap:Header>
...
...
</soap:Envelope>
```

Tělo

- Povinný element
- Díky němu může být provedena příslušná operace

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body>
  <m:GetPrice xmlns:m="https://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>

</soap:Envelope>
```

Chyba (Fault)

- Vyskytuje se pouze v odpovědích
- Signalizuje chybu, která nastala při zpracování požadavku
- Klient – chyba na straně klienta (zpráva špatně formátovaná nebo neobsahovala úplné informace)
- Server – chyba na straně serveru (nebyl schopen zprávu zpracovat)

WSDL

- Jazyk na bázi XML
- Navržený pro popis webových služeb
- Webová služba běžící na serveru současně publikuje svůj popis ve WSDL formátu a klient zná funkcionalitu služby
- První zveřejněná verze byla vydaná v roce 2001
- Definuje jména operací, jména a typy vstupních parametrů a návratové hodnoty
- Díky standardizaci mohou různé systémy (například psané v různých programovacích jazycích) snadno komunikovat

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>

<binding type="glossaryTerms" name="b1">
   <soap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http" />
   <operation>
     <soap:operation soapAction="http://example.com/getTerm"/>
     <input><soap:body use="literal"/></input>
     <output><soap:body use="literal"/></output>
  </operation>
</binding>
```

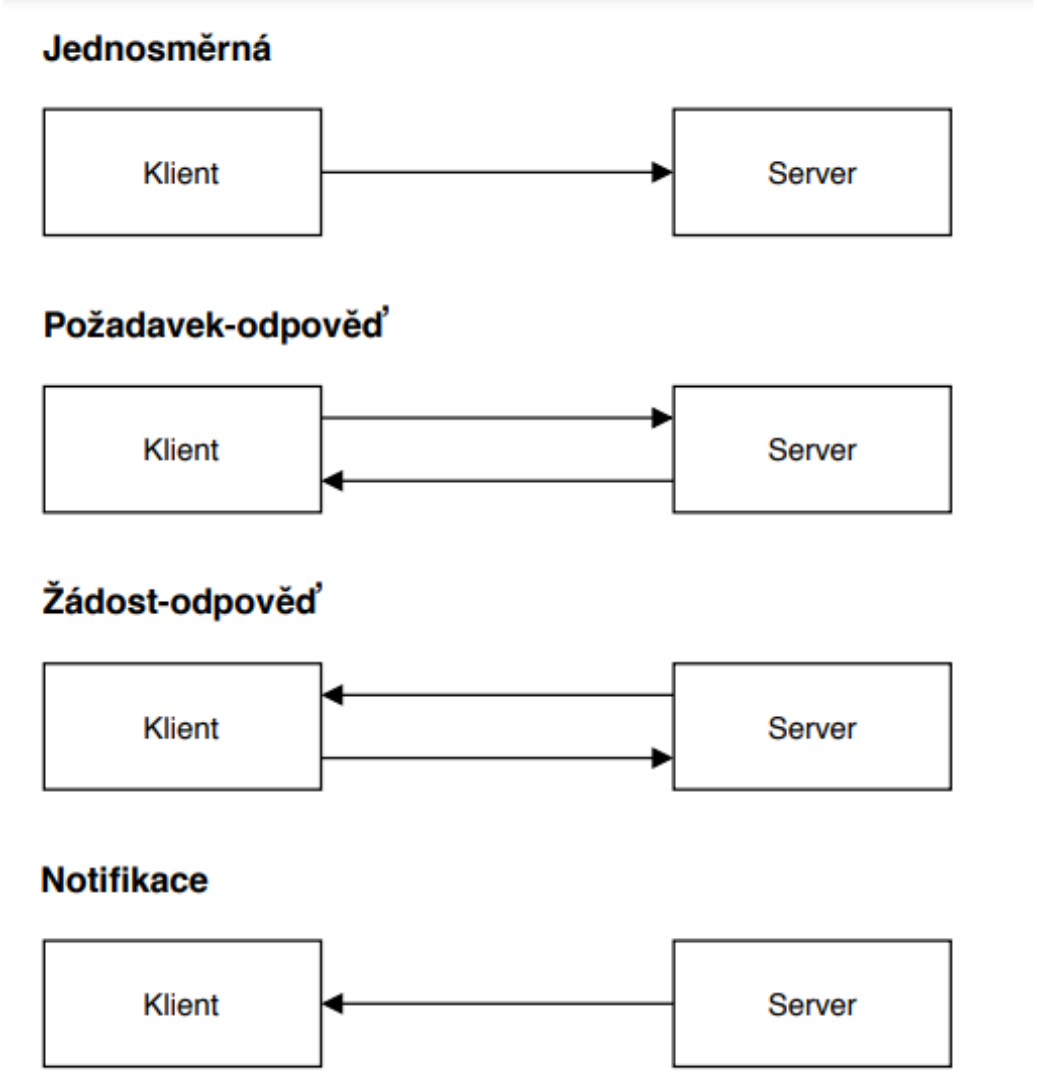
WSDL

- Zpráva (Message) - definuje formát zprávy (lze si představit jako parametry u funkcí)
- Operace (Operation) – definuje vzdálené volání funkcí
- Vazba (Binding) - definuje, jaký protokol se používá
- Služba (Service) - definuje konkrétní adresu (URL), na které je služba dostupná

```
<service name="WeatherService">
    <port name="WeatherServicePort" binding="tns:WeatherServiceSoapBinding">
        <soap:address location="http://example.com/weather/service"/>
    </port>
</service>
```

WSDL

- Zpráva (Message) - definuje formát zprávy (lze si představit jako parametry u funkcí)
- Operace (Operation) – definuje vzdálené volání funkcí
- Vazba (Binding) - definuje, jaký protokol se používá
- Služba (Service) - definuje konkrétní adresu (URL), na které je služba dostupná



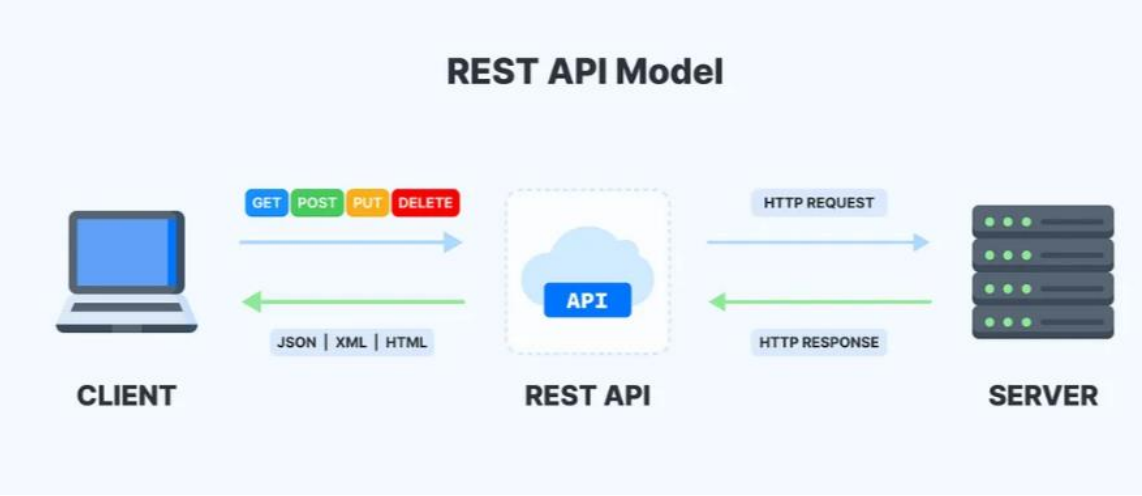
Typy operací

- Operace - volání funkcí
- Nejběžněji se používá typ: Požadavek-Odpověď (klient pošle zprávu a server mu odpoví zprávou)

REST

REPRESENTATIONAL STATE TRANSFER

REST



- Architektonický styl navržený pro distribuovaná prostředí pracující s hypermédii
- Definuje způsob práce se zdroji
- Byl navržen v roce 2000

REST - Ukázka

```yaml
---
openapi: 3.0.1
info:
  title: Price API
  version: '2.10'
paths:
  "/prices/apple":
    get:
      summary: Get current apple price
      operationId: applyPrice
      responses:
        '200':
          description: Successful response
          content:
            application/json:
              schema:
                type: object
                properties:
                  value:
                    type: number
```

Srovnání SOAP a REST

	SOAP	**REST**
Typ	Standardizovaný protokol.	Architektonický styl.
Komunikace	Prostřednictvím služeb s definovanými operacemi.	Prostřednictvím tzv. zdrojů.
Stavovost	Podporuje stavovou i bezstavovou komunikaci.	Komunikace je z principu bezstavová.
Mezipaměť	Nemůže používat mezipaměť (cache).	Může používat mezipaměť.
Bezpečnost	Definuje vlastní standardy, např. WS-security.	Neřeší, ponechává na transportní vrstvě (HTTPS).
Formát zpráv	Pouze XML.	Libovolný – HTML, XML, JSON, YAML apod.
Výkon	Vyšší nároky na objem přenášených dat i výpočtu (zpracování XML).	Nižší nároky na objem přenášených dat. Nemusí zpracovávat přenášená data pro zjištění účelu zprávy.
Výhody	Standardizovanost, rozsáhlé možnosti zabezpečení, rozšiřitelnost.	Škálovatelnost, výkonnost, jednoduchost a flexibilita.
Nevýhody	Horší výkon, složitost (velké množství standardů – chaotičnost), menší flexibilita.	Bezpečnost, do určité míry absence standardů.

Technologie a nástroje pro webové služby

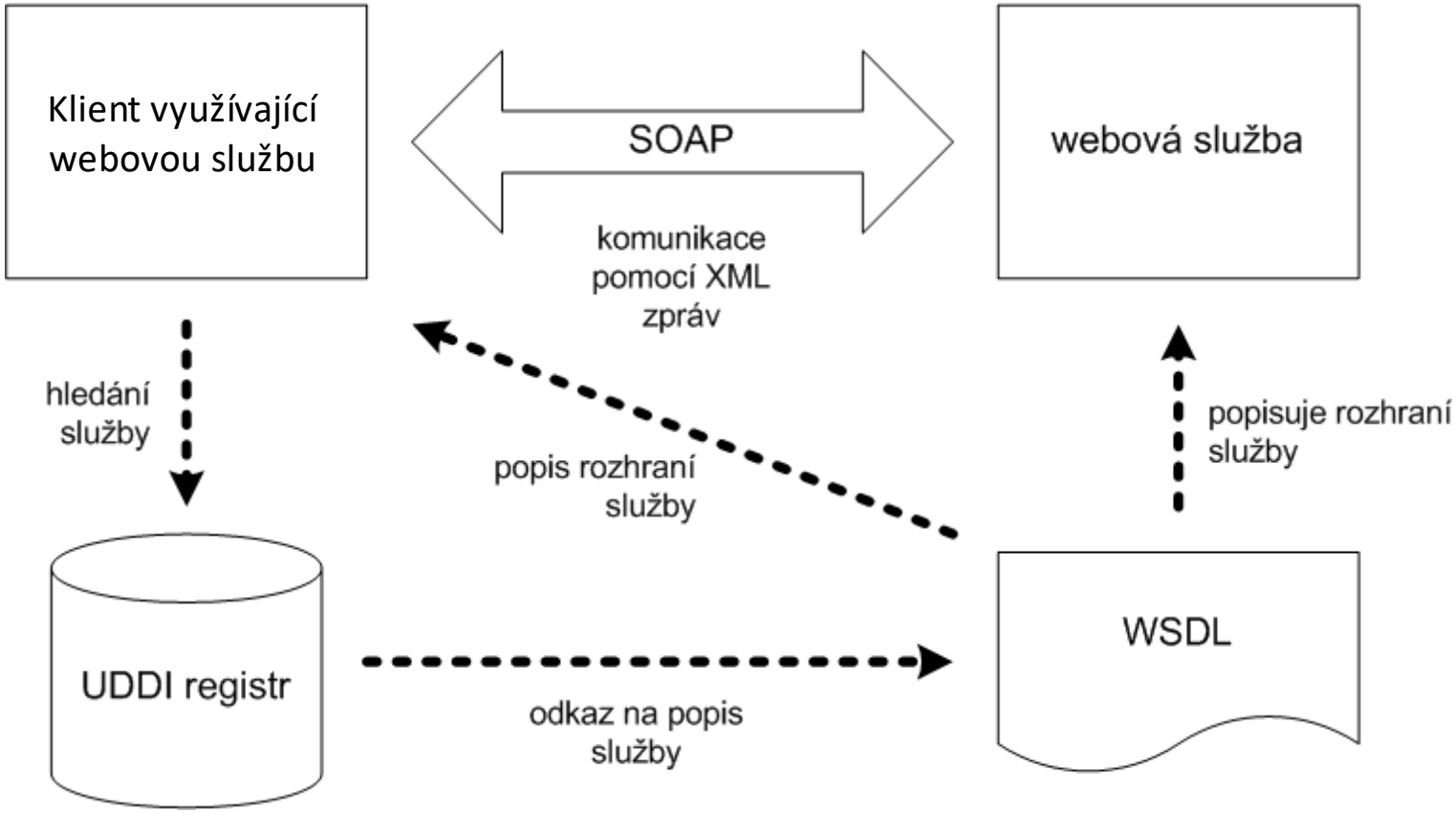
Ekosystém SOAP

- Mnoho specifikací (standardů) na řešení problémů s vývojem pomocí protokolu
- Většina vytvořena firmami IBM nebo Microsoft
- Standardizaci zajišťuje OASIS
  - Organization for the Advancement of Structured Information Standards
  - Vývoj, schvalování a propagace otevřených standardů v IT
- Patří sem již zmíněné WSDL
  - Definuje dostupnost služeb, jejich parametry, typy odpovědí, …
  - Automatizace, standardizace, spolupráce v různých programovacích jazycích

UDDI
Universal Description, Discovery and Integration

- Standardizovaný registr pro vyhledávání a registraci webových služeb popsaných pomocí WSDL
- Založené na XML, komunikuje pomocí SOAP
- Prvky v registru UDDI:
  - Podnikatelské entity (business entities)
    - Info o firmě nebo organizaci; název, popis činnosti, kontaktní údaje
  - Služby (business services)
    - Popis konkrétní nabízené služby; seznam šablon vazeb (binding templates)
  - Šablony vazeb (binding templates)
    - Definují, jak s danou službou komunikovat; odkaz na WSDL soubor
  - Typy služeb (service types)
    - Abstraktní definice služeb; více firem může nabízet služby se stejným typem

Vztah mezi SOAP, WSDL a UDDI



[https://www.kosek.cz/diplomka/html/websluzby.html]

WS specifikace

- Rozšíření funkcionality protokolu SOAP
- Označovány jako WS-* (výjimkou je WSDL), standardizovány OASIS
- Řeší specifické potřeby jako bezpečnost, spolehlivost, směrování zpráv,…

WS-Security

- Použití zejména v případě jiného transportního protokolu než HTTP
- Zajišťuje integritu, šifrování či autentizaci zpráv
- Umožňuje do SOAPových zpráv přidat tokeny (např. jméno a heslo), certifikát nebo info pro Kerberos
- Nezajišťuje sama bezpečnostní řešení, je základem pro další technologie a protokoly
  - Např. využívá XML šifrování, určuje jak vložit hash nebo šifrovaná data do SOAP zprávy

WS-ReliableMessaging

- Zajišťuje spolehlivé doručování zpráv
- Zasílání zpráv skrze prostředníky, kteří potvrzují příjem
- 4 typy možných záruk:
  - *ExactlyOnce* - doručení přesně jednou bez duplicit; při nedoručitelnosti vznikne chyba
  - *AtMostOnce* - doručení maximálně jednou bez duplicit; nemusí být doručena vůbec
  - *AtLeastOnce* - doručení minimálně jednou, při nedoručitelnosti chyba
  - *InOrder* - přijetí zpráv v pořadí zaslání odesilatelem; možnost spojit s některou z předchozích

WS-Policy

- Popisuje požadavky a omezení služby (např. bezpečnostní pravidla, autentizace)
- Může určovat, že komunikace musí být šifrovaná nebo použít specifické záruky

---

- Mnoho dalších WS-* specifikací: WS-Transaction, WS-Notification, WS-Addressing,…

Ekosystém REST

- Převážně modelovací jazyky pro formální definici API a práci s ním, nástroje pro testování

OpenAPI (dříve Swagger specifikace)

- Human-friendly specifikace API, aktuálně nejpoužívanější
- Specifikuje jazyky YAML a JSON
- Dělí prvky API na komponenty (objekty dat, odpovědí, parametry požadavků, apod.)

API Blueprint

- Podobná funkcionalita jako OpenAPI, syntaxe založená na Markdownu
- Umožňuje rychlý návrh a testování

RAML (RESTful API Modeling Language)

- Podobný OpenAPI, stavěný na YAML

```yaml
openapi: 3.0.0
info:
  title: User endpoint.
  version: ’1.0’

paths:
  /users:
    get:
      responses:
        200:
          description: User was succesfully deleted from server.
          content:
            application/json:
              schema:
                type: array
                items:
                  type: object
                  properties:
                    name:
                      type: string
                      example: "Váleč Koulivý"
                    age:
                      type: integer
                      example: 12
```

OpenAPI – definice REST API ve formátu YAML

Swagger

- Kolekce nástrojů pro vývoj a dokumentaci v OpenAPI
- Návrh dokumentace, testování, vývoj, nasazování,...
- Open source i licencované nástroje
  - *Swagger Codegen* – pro podporované jazyky generuje serverové i klientské aplikace v OpenAPI specifikaci
  - *Swagger Editor* – WYSIWYG editor pro vývoj designu API a jeho dokumentace v OpenAPI
  - *Swagger Inspector* – REST API klient pro funkční testování
  - *Swagger Hub* – integrace všech předešlých nástrojů pro týmové využití, umí vytvořit mock server

Apiary

- Nástroj pro zjednodušení vývoj API, podobný Swagger Hub
- Původně vyvinutý českým start-upem, odkoupil ho Oracle
- Využívá API Blueprint

Mock Server

- Simulace chování jiného serveru
- Poskytuje náhodně generovaná statická nebo dynamická data
- Užitečný zejména při vývoji aplikací typu klient-server
  - Vývoj serverové části aplikace obvykle náročnější a delší
- Použitelný i k testování
  - Snadné nastavení poskytovaných dat bez nutnosti uložení do databáze apod.
- Některé nástroje vytvoří mock server samy, ale dá se naprogramovat i vlastní

Apache JMeter

- Nástroj k měření výkonnosti webových aplikací, open-source v jazyce Java
- Využívá HTTP, lze použít i k testování SQL nebo služeb na jiných protokolech
  - Díky HTTP umí testovat SOAP i REST, pokud taky využívají HTTP
- Simulace reálné zátěže pomocí tzv. test plánů
  - Lze uvést formu a počet požadavků, počet vláken simulujících uživatele, různé typy zpoždění...
- Nástroje pro data v HTML, JSONu, XML nebo TXT
- Uživatelsky definovatelné pluginy ve skriptovacím jazyce Groovy

Děkujeme za pozornost