Webové služby

Martin Sochor

Webové služby

- způsob komunikace dvou aplikací přes Web
- binární zprávy (CORBA) blokovány proxy servery a firewally
- masivní využití XML
- protokol SOAP + jazyk pro popis služeb WSDL
- přístup REST

XML Schema

- popis struktury XML dokumentu

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="student">
  <xs:complexType>
     <xs:sequence>
        <xs:element name="jmeno" type="xs:string" />
        <xs:element name="prijmeni" type="xs:string" />
        <xs:element name="absence" type="xs:integer" />
     </xs:sequence>
     <xs:attribute name="uid" type="xs:string" />
  </xs:complexType>
</xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<student xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
         xsi:noNamespaceSchemaLocation="student.xsd"
         uid="sochomar">
  <jmeno>Martin</jmeno>
  <prijmeni>Sochor</prijmeni>
  <absence>5</absence>
</student>
```

SOAP

- Simple Object Access Protocol
- založený na XML
- rozšiřitelný
- nezávislý na transportním protokolu
- nezávislý na použité technologii

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">

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

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

</soap:Envelope>
```

`<soap:Envelope>`

- atributy:
  - povinně `xmlns:soap` odkazující na adresu http://www.w3.org/2001/12/soap-envelope
  - `encodingStyle` definuje použité datové typy

<soap:Header>

- obsahuje informace o zprávě samotné
- atributy:
  - mustUnderstand
  - actor
  - encodingStyle

`<soap:Body>`

- samotná zpráva
- obsahuje `<soap:Fault>`, pokud došlo k chybě

<soap:Fault>

- prvky:
  - <faultcode>
  - <faultstring>
  - <faultactor>
  - <detail>

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 322

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 330

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

WSDL

- Web Service Description Language
- strojově čitelný popis rozhraní webové služby
- lze jej generovat z kódu
- lze z něj generovat kód
- abstraktní část: popis použitých datových typů, popis dostupných operací (rozhraní)
- konkrétní část: vazby rozhraní k protokolům, definice služby a jejích koncových bodů

```
<description>

<types>
  ...
</types>

<interface>
  ...
</interface>

<binding>
  ...
</binding>

<service>
  ...
</service>

</description>
```

<types>

- obsahuje definice použitých datových typů
- definice jsou ve formě XML Schema
- typy určené pro vstup/výstup musí být jednoduché (tj. ne složené)

```
<types>
  <xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://greath.example.com/2004/schemas/res
Svc"
    xmlns="http://greath.example.com/2004/schemas/resSvc">
    <xs:element name="checkAvailability"
type="tCheckAvailability"/>
    <xs:complexType name="tCheckAvailability">
      <xs:sequence>
        <xs:element name="checkInDate" type="xs:date"/>
        <xs:element name="checkOutDate" type="xs:date"/>
        <xs:element name="roomType" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
    <xs:element name="checkAvailabilityResponse"
type="xs:double"/>
    <xs:element name="invalidDataError" type="xs:string"/>
  </xs:schema>
</types>
```

<interface>

- definice samotného rozhraní, dostupných operací, případných chyb
- obsahuje prvky:
  - <operation> definuje dostupné operace a datové typy, které jsou použity jako zprávy
  - <fault> definuje datové typy, které jsou použity jako zprávy o chybě

```
<interface name="reservationInterface">
  <fault name="invalidDataFault"
element="ghns:invalidDataError"/>
  <operation name="opCheckAvailability">
    <input messageLabel="In"
element="ghns:checkAvailability" />
    <output messageLabel="Out"
element="ghns:checkAvailabilityResponse" />
    <outfault ref="tns:invalidDataFault"
messageLabel="Out"/>
  </operation>
</interface>
```

<binding>

- definuje konkrétní formát (kódování) zprávy a protokol použitý k jejímu odeslání
- může použít různé protokoly a formáty pro různé operace v rámci téže vazby a téhož rozhraní

```
<binding name="reservationSOAPBinding"
interface="tns:reservationInterface"
type="http://www.w3.org/ns/wsdl/soap"
wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTT
P/">
  <operation ref="tns:opCheckAvailability"
wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-
response"/>
  <fault ref="tns:invalidDataFault"
wsoap:code="soap:Sender"/>
</binding>
```

<service>

- definuje, kde se na webu služba nachází
- jedna služba má jen jedno rozhraní
- jedna služba může mít více koncových bodů (pro různé vazby)

```
<service name="reservationService"
   interface="tns:reservationInterface">
 <endpoint name="reservationEndpoint"
     binding="tns:reservationSOAPBinding"
     address
="http://greath.example.com/2004/reservation"/>
</service>
```

<documentation>

- WSDL je strojově čitelný formát; stroje ale nejsou jediné čtenáře
- nepovinné, leč nezbytné

„Klasická“ webová služba

- server poskytuje svoje rozhraní jako WSDL
- klientská aplikace je vytvořena z WSDL
- klient si se serverem posílá zprávy ve formátu SOAP, jejichž struktura je popsaná daným WSDL

REST

- Representational State Transfer
- styl architektury distribuovaných aplikací
- inspirován architekturou samotného webu
- není to konkrétní technologie, ale sada principů

Principy RESTu

- architektura klient-server
  - klient se nezajímá o ukládání dat, server se nezajímá o uživatelské rozhraní; spěje k modularitě
- není uchováván stav (na serveru)
  - každý požadavek klienta obsahuje všechny informace potřebné k jeho provedení; stav uchovává pouze klient
- lze ukládat do cache
  - každý požadavek definuje, jestli jej lze uložit do cache; vede ke snížení požadavků na server a urychlení služby

Principy RESTu

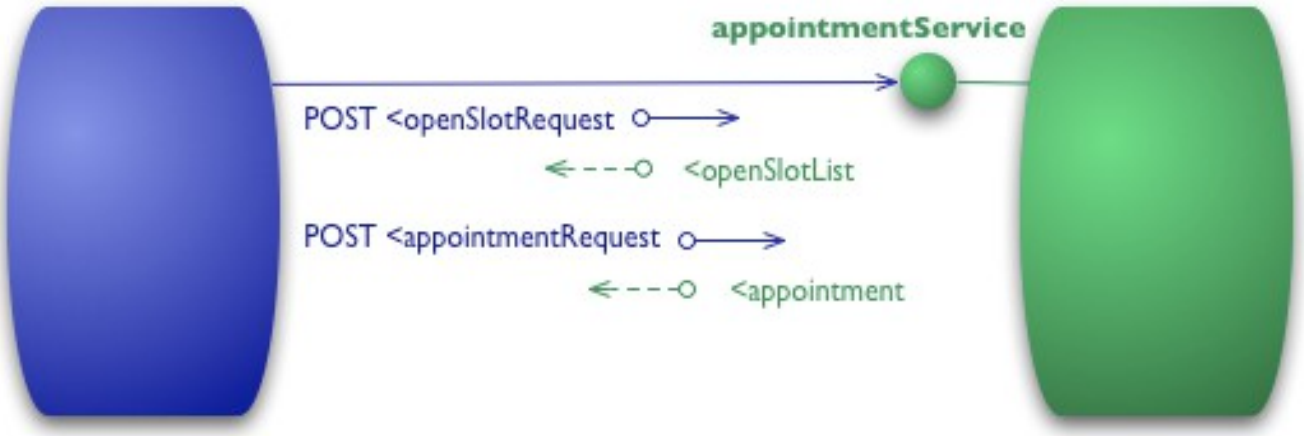
- vrstvený systém
  - klienta nezajímá, zda jeho požadavek míří přímo na koncový server, nebo skrz mezičlánky; snazší implementace sdílených cache a vyvažování zátěže
- jednotné rozhraní
  - používá se co možná nejvíc možností webu – HTTP slovesa (GET, POST) a kódy (200 OK, 404 Not Found)
- (kód na vyžádání)
  - JavaScript, Java applety

Jednotné rozhraní

- identifikace zdrojů (resources)
  - každá zpráva obsahuje informaci o tom, kterého zdroje se týká; odpověď obsahuje reprezentaci zdroje
- samopopisující zprávy
  - každá zpráva obsahuje informaci o tom, jak ji má server interpretovat (Content-Type)
- hypermédia
  - každá zpráva ze serveru obsahuje dynamicky odkazy na další dostupné akce

3 úrovně RESTu

- úroveň 0: Plain Old XML (POX)

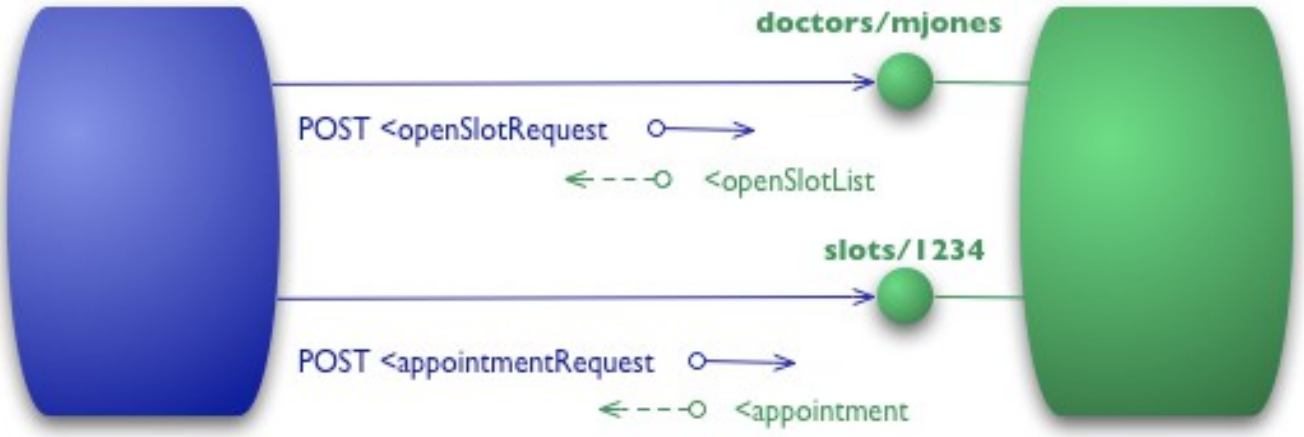


```
POST /appointmentService HTTP/1.1

<openSlotRequest date = "2010-01-04" doctor = "mjones"/>



HTTP/1.1 200 OK

<openSlotList>
  <slot start = "1400" end = "1450">
    <doctor id = "mjones"/>
  </slot>
  <slot start = "1600" end = "1650">
    <doctor id = "mjones"/>
  </slot>
</openSlotList>
```

```
POST /appointmentService HTTP/1.1

<appointmentRequest>
  <slot doctor = "mjones" start = "1400" end = "1450"/>
  <patient id = "jsmith"/>
</appointmentRequest>


HTTP/1.1 200 OK

<appointment>
  <slot doctor = "mjones" start = "1400" end = "1450"/>
  <patient id = "jsmith"/>
</appointment>


HTTP/1.1 200 OK

<appointmentRequestFailure>
  <slot doctor = "mjones" start = "1400" end = "1450"/>
  <patient id = "jsmith"/>
  <reason>Slot not available</reason>
</appointmentRequestFailure>
```

3 úrovně RESTu

- úroveň 1: zdroje

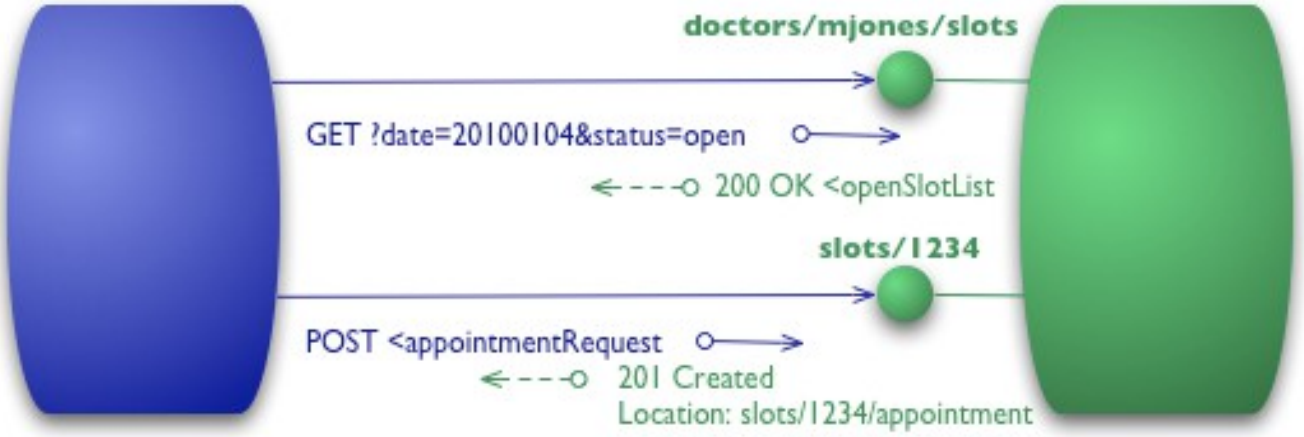


```
POST /doctors/mjones HTTP/1.1

<openSlotRequest date = "2010-01-04"/>


HTTP/1.1 200 OK

<openSlotList>
  <slot id = "1234" doctor = "mjones" start = "1400" end =
"1450"/>
  <slot id = "5678" doctor = "mjones" start = "1600" end =
"1650"/>
</openSlotList>
```

```
POST /slots/1234 HTTP/1.1

<appointmentRequest>
  <patient id = "jsmith"/>
</appointmentRequest>


HTTP/1.1 200 OK

<appointment>
  <slot id = "1234" doctor = "mjones" start = "1400" end =
"1450"/>
  <patient id = "jsmith"/>
</appointment>
```

3 úrovně RESTu

- úroveň 2: HTTP



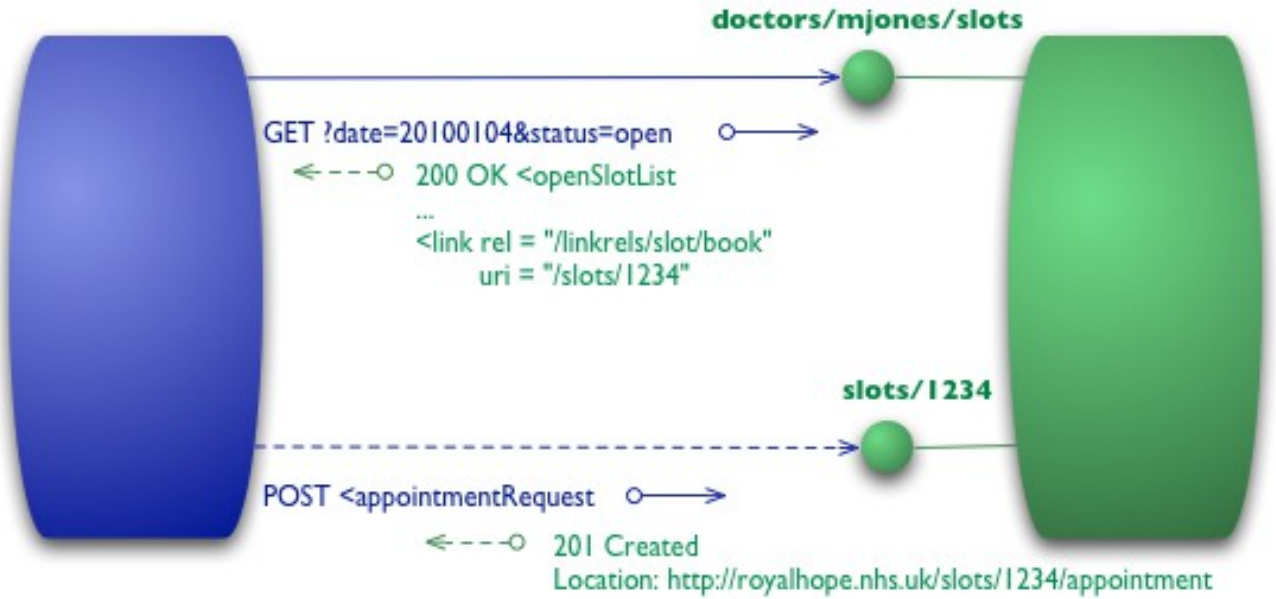
- díky GET lze cachovat
- kódy umožňují např. předání umístění nového zdroje

```
GET /doctors/mjones/slots?date=20100104&status=open
HTTP/1.1
Host: royalhope.nhs.uk


HTTP/1.1 200 OK

<openSlotList>
  <slot id = "1234" doctor = "mjones" start = "1400" end =
"1450"/>
  <slot id = "5678" doctor = "mjones" start = "1600" end =
"1650"/>
</openSlotList>
```

```
POST /slots/1234 HTTP/1.1

<appointmentRequest>
  <patient id = "jsmith"/>
</appointmentRequest>


HTTP/1.1 201 Created
Location: slots/1234/appointment

<appointment>
  <slot id = "1234" doctor = "mjones" start = "1400" end =
"1450"/>
  <patient id = "jsmith"/>
</appointment>


HTTP/1.1 409 Conflict

<openSlotList>
  <slot id = "5678" doctor = "mjones" start = "1600" end =
"1650"/>
</openSlotList>
```

3 úrovně RESTu

- úroveň 3: hypermédia



- obsahuje informace o akcích, které jsou k dispozici
- umožňuje změnit vnitřní strukturu serveru beze změny klientského kódu

```
GET /doctors/mjones/slots?date=20100104&status=open
HTTP/1.1
Host: royalhope.nhs.uk


HTTP/1.1 200 OK

<openSlotList>
  <slot id = "1234" doctor = "mjones" start = "1400" end =
"1450">
     <link rel = "/linkrels/slot/book"
           uri = "/slots/1234"/>
  </slot>
  <slot id = "5678" doctor = "mjones" start = "1600" end =
"1650">
     <link rel = "/linkrels/slot/book"
           uri = "/slots/5678"/>
  </slot>
</openSlotList>
```

```
POST /slots/1234 HTTP/1.1

<appointmentRequest>
  <patient id = "jsmith"/>
</appointmentRequest>

HTTP/1.1 201 Created
Location: http://royalhope.nhs.uk/slots/1234/appointment

<appointment>
  <slot id = "1234" doctor = "mjones" start = "1400" end =
"1450"/>
  <patient id = "jsmith"/>
  <link rel = "/linkrels/appointment/cancel"
        uri = "/slots/1234/appointment"/>
  <link rel = "/linkrels/appointment/addTest"
        uri = "/slots/1234/appointment/tests"/>
  <link rel = "self"
        uri = "/slots/1234/appointment"/>
  <link rel = "/linkrels/appointment/changeTime"
        uri = "/doctors/mjones/slots?
date=20100104@status=open"/>
  <link rel = "/linkrels/appointment/updateContactInfo"
        uri = "/patients/jsmith/contactInfo"/>
  <link rel = "/linkrels/help"
        uri = "/help/appointment"/>
</appointment>
```

...jak to vypadá v praxi?

Díky za pozornost