Web Services

Martin Kuba makub@ics.muni.cz

Superpočítačové Centrum Brno,
Masarykova Univerzita

Obsah

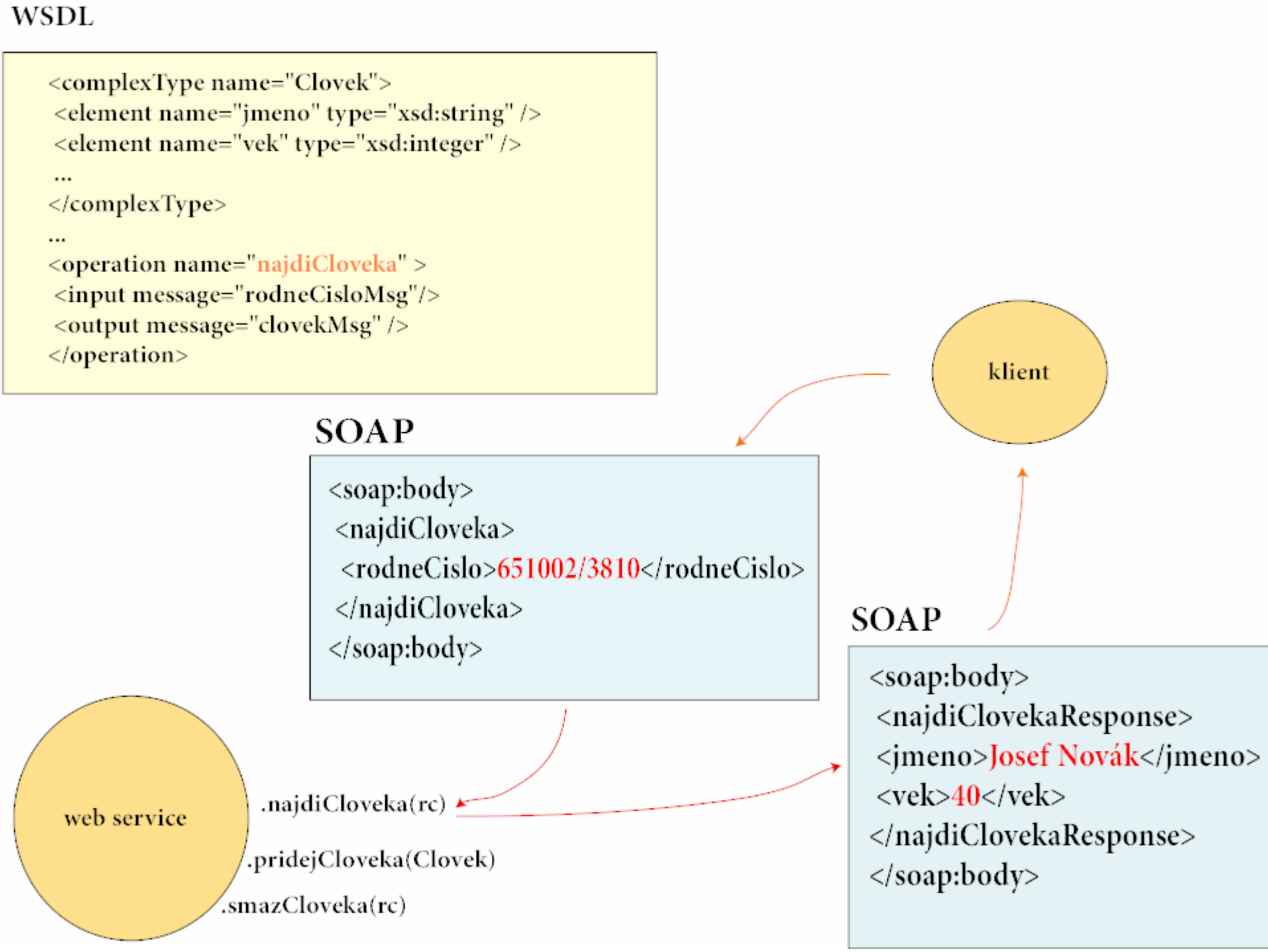
- definice webových služeb
- historický vývoj ze strany WWW
- pozadí distribuovaných systémů – RPC, RMI, SOA
- SOAP webové služby
  - XML, URI, XML Namespaces, XML Schema
  - protokol SOAP
  - jazyk WSDL, styly WSDL
  - nástroje
- vyhledávání služeb - UDDI, WSIL
- džungle specifikací WS-*, REST
- Grid, OGSA
- bezpečnost u webových služeb
- ESB, WSBPEL
- sémantické webové služby

Co je web service

- W3C WS-ARCH: *„Webová služba je softwarový systém zkonstruovaný k podpoře interakce mezi stroji přes síť. Má rozhraní popsané ve strojově zpracovatelném formátu (specificky WSDL). Ostatní systémy interagují s webovou službou způsobem předepsaným jejím popisem za pomoci SOAP zpráv, typicky dopravovaných použitím HTTP s XML serializací v součinnosti s ostatními webovými standardy.“*

WSDL

```
<complexType name="Clovek">
 <element name="jmeno" type="xsd:string" />
 <element name="vek" type="xsd:integer" />
 ...
</complexType>
...
<operation name="najdiCloveka" >
 <input message="rodneCisloMsg"/>
 <output message="clovekMsg" />
</operation>
```



Výhody a nevýhody Web Services

- výhody vyplývají z XML
- žádné problémy s češtinou nebo i18n, díky XML – vše v UTF-8 nebo UTF-16
- nezávislost na programovacím jazyku, objektové orientovanosti, platformě
- nízká vstupní bariéra
- vhodné pro loosely coupled systémy – klient a server o sobě téměř nic nepředpokládají
- nevýhoda – parsování XML je drahé/pomalé

Anatomie URL

- web je tvořen dokumenty adresovanými URL (Uniform Resource Locator)
- *protocol://user@host:port/path?query#fragment*
- query má mít tvar (podle HTML4)
  - *kytka=pampeli%C5%A1ka&kytka=zvonek*
  - nebo oddělený středníky či libovolný
- kódování českých znaků nelze určit, tj. nelze české znaky spolehlivě předávat
- informace pro aplikaci na straně serveru lze mít v path i query
- délka URL je omezena na cca 4 kB

Historický vývoj ze strany WWW

- 1. krok – HTTP metoda GET s URL parametry
  - omezení na 4kb

```
GET /prog?a=1&b=Pepa+Novak HTTP/1.0
```

- 2. krok – metoda POST s URL parametry
  - omezení na pouze páry řetězců
  - stále problémy s kódováním znaků

```
POST /prog HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 16

a=1&b=Pepa+Novak
```

Historický vývoj SOAP

- 3. krok – metoda POST s XML v těle
- strukturovaná data, typovaná data
- stejný nápad mělo více lidí a firem, vznikly XML-RPC, WDDX, XMI a další
- od r. 1998 Microsoft a IBM tvořili SOAP (Simple Object Access Protocol)
- SOAP byl myšlen jako RPC pomocí XML a HTTP
- r. 2000 přijato W3C SOAP 1.1 jako *Note*
- r. 2003 vydalo W3C SOAP 1.2 jako *Recommendation*
- r. 2004 WS-Interoperability www.ws-i.org vydalo BasicProfile 1.0

Pozadí - RPC

- distribuované systémy komunikují zasíláním zpráv
- vhodná abstrakce – RPC - Remote Procedure Calls
- např. DCE RPC, SUN RPC, ...
- synchronní volání požadavek-odpověď
  - volaná procedura, parametry předávané hodnotou
  - návratové hodnoty
- IDL – Interface Definition Language
- klient a server stubs
  - volané jako běžné funkce v daném prog. jazyce
  - zajišťují marshalling/serializaci, komunikaci, unmarshaling/deserializaci

Komunikace v distribuovaných systémech

- z hlediska synchronnosti
  - synchronní – volající strana zastaví a čeká, dokud nedostane odpověď
  - asynchronní – volající strana pokračuje v práci, na příchod odpovědi je upozorněna
- z hlediska zajištění doručení zprávy
  - transientní (pomíjivá)
  - persistentní (vytrvalá)
- webové služby obvykle používají transientní synchronní komunikaci, ale lze použít všechny kombinace

Pozadí - RMI

- distribuované objektově-orientované systémy potřebují předávat parametry odkazem
- distribuovaný objekt má stav a metody – **interface**, a **implementaci**
- Remote Method Invocation
- např. CORBA, Java RMI, DCOM
- binární protokoly, Object Request Broker
- Java RMI umí předat objekt – stav i implementaci metod

Pozadí - SOA

- RMI funguje jen v systémech pod centrální správou, neškáluje na Internet-size
  - synchronní komunikace neškáluje
  - tight coupling, verzování a evoluce jsou obtížné
  - distribuovanost nelze schovat (partial failure)
- SOA – Service Oriented Architecture
  - služby mají definovaný interface
  - interface je popsán zprávami, ne operacemi na datových typech
  - služby lze nalézt (např. v adresáři)

SOA (2)

- rozdíl mezi OO a SOA
  - přehrávač CD poskytuje službu přehrání CD
  - různá kvalita služby ve walkmanovi nebo HiFi věži
  - v objektově-orientovaném přístupu by každé CD bylo dodáno s vlastním přehrávačem, ze kterého by nešlo vyjmout
- SOA patrně lépe odpovídá způsobu, jímž jsou organizovány lidské aktivity

Co jsou SOAP Web Services

- technologie pro vzdálené volání procedur pomocí výměny XML zpráv
- tři části
  - komunikační protokol SOAP (Simple Object Access Protocol)
  - popis služby ve WSDL (Web Service Description Language)
  - vyhledání služby (UDDI,WSIL)

Vývoj SOAP

- SOAP - komunikační protokol webových služeb
- začal jako XML-based RMI – Simple Object Access Protocol
- definoval vlastní typový systém – zdroj problémů s kompatibilitou
- až později se začal používat podle zásad SOA a s typovým systémem XML Schema
- tj. lze ho použít dobře i špatně

URI

- URI (Universal Resource Identifier)
  - URL (Universal Resource Locator)
  - URN (Universal Resource Name)
- URL určuje zdroj jeho *umístěním*, např. http://www.nekde.cz/cesta/soubor.html
- URN určuje zdroj jeho *jménem*, např. urn:isbn:628361298, ed2k:0b366c8e95b43
- URI jsou celosvětově jedinečná - nekolidují
- !! URL použitá jako URI nemusí odkazovat na existující zdroj !!!
- IRI (Internationalized Resource Identifier) smí obsahovat libovolné UNICODE znaky, URI jen ASCII znaky, je definován převod IRI na URI (RFC3987)

Rychlokurz XML

- XML je značkovací jazyk se stromovou strukturou, právě jeden kořen, tagy, atributy, texty

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- komentář -->
<kořen atribut="hodnota atributu" další="1&lt;1" >
  <vnořený_tag id="tady" />
  nebezpečné znaky: &lt; &gt; &amp; &quot; &apos;
        <![CDATA[  <  >  &  "  '  ]]>
</kořen>
```

XML Namespaces

- zabraňují kolizím stejných jmen pro různé věci
- týká se jmen tagů i atributů
- *qualified names* – prefix:localpart
- prefix je mapován na URI
- významné je URI, ne prefix – dva prefixy mapované na stejné URI definují stejný jmenný prostor
- mapování provedeno atributem xmlns:prefix="<*URI*>"
- *default namespace* atributem xmlns="<*URI*>"

XML Namespaces (2)

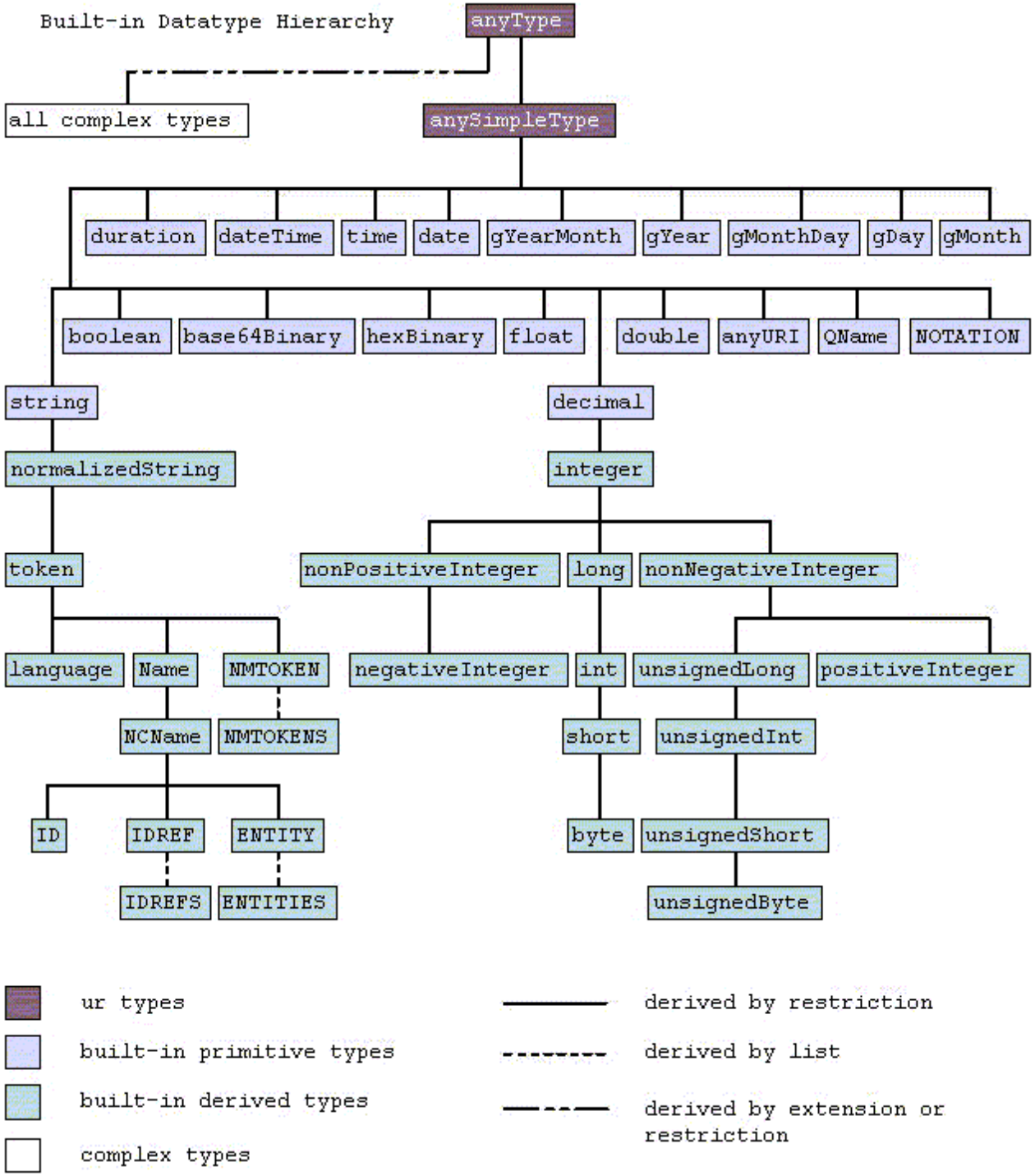
```
<?xml version="1.0" encoding="UTF-8" ?>
<koren xmlns:umělecký="urn:Michelangelo“
      xmlns:pohlavní="http://www.sex.cz/"
      xmlns:divadelní="http://www.divadlo.cz/hra"
      xmlns:pietní="http://www.krematorium.cz/"
      xmlns:u="urn:Michelangelo“
      xmlns="http://urad.cz/akta" >

Tohle jsou různé tagy:
 <umělecký:akt> obraz </umělecký:akt>
 <pohlavní:akt> styk </pohlavní:akt>
 <divadelní:akt> dějství </divadelní:akt>
 <pietní:akt> pohřeb </pietní:akt>
 <akt> úřední papír </akt>
Tohle jsou stejné tagy:
 <umělecký:akt> obraz </umělecký:akt>
 <u:akt> zase obraz </u:akt>

</koren>
```

XML Schema

- norma pro definici datových struktur a datových typů v XML
- typy:
  - simple types
    - atomic types (string, byte, integer, long, double, boolean, base64Binary, date, duration, ...)
    - list types (pole)
    - union types (variantní typy)
    - derived types (vzniklé omezením, např. číselný interval)
  - complex types (složené typy)
- parsery XML umí kontrolovat vůči konkrétnímu Schema při načítání dokumentu


Built-in Datatype Hierarchy
anyType
all complex types
anySimpleType
duration
dateTime
time
date
gYearMonth
gYear
gMonthDay
gDay
gMonth
boolean
base64Binary
hexBinary
float
double
anyURI
QName
NOTATION
string
decimal
normalizedString
integer
token
nonPositiveInteger
long
nonNegativeInteger
language
Name
NMTOKEN
negativeInteger
int
unsignedLong
positiveInteger
NCName
NMTOKENS
short
unsignedInt
ID
IDREF
ENTITY
byte
unsignedShort
IDREFS
ENTITIES
unsignedByte
ur types
built-in primitive types
built-in derived types
complex types
derived by restriction
derived by list
derived by extension or restriction

XML Schema (2)

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:impl="urn:mojedata"
        targetNamespace="urn:mojedata" >
<!– složené typy -->
 <complexType name="Adresa">
    <sequence>
       <element name="ulice" type="xsd:string"/>
       <element name="cislo" type="xsd:long"/>
       <element name="mesto" type="xsd:string"/>
    </sequence>
 </complexType>
 <complexType name="Osoba">
    <sequence>
       <element name="jmeno" type="xsd:string"/>
       <element name=„narozeni" type="xsd:dateTime"/>
       <element name="adresa" type="impl:Adresa"/>
    </sequence>
</complexType>
```

XML Schema (3)

```
<!-- číselný interval - integer omezený na 10000 až 99999 -->
 <simpleType name="myInteger">
   <restriction base="xsd:integer">
      <minInclusive value="10000"/>
      <maxInclusive value="99999"/>
   </restriction>
 </simpleType>

<!-- výčet - string omezený na vyjmenované hodnoty -->
 <simpleType name="ovoce">
    <restriction base="xsd:string">
      <enumeration value="jablka"/>
      <enumeration value="hrušky"/>
      <enumeration value="banány"/>
   </restriction>
 </simpleType>
</schema>
```

SOAP – Simple Object Access Protocol

- není simple ☺
- umožňuje vzdáleně volat funkce
- HTTP protokol přenese XML zprávu
- zpráva popisuje volanou funkci a její parametry
- jako odpověď přenese HTTP zpět opět XML zprávu reprezentující výsledná data
- teoreticky nemusí být HTTP (ale SMTP, FTP, JMS, MQSeries, ...), může být jen jednosměrný přenos

SOAP (2)

- příklad – mějme funkci (operaci)

  boolean jePrvocislo(long cislo)
- typy boolean a long jsou z XML Schema
- operace musí být světově jedinečně pojmenovaná, proto je v namespace, třeba urn:mojeURI

SOAP request

```
POST / HTTP/1.1
Content-Type: text/xml; charset=utf-8
Content-Length: 411
Connection: close
SOAPAction: ""

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns1="urn:mojeURI">
 <SOAP-ENV:Body>
    <ns1:jePrvocislo>
       <cislo>1987</cislo>
    </ns1:jePrvocislo>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 433
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:ns1="urn:mojeURI">
 <SOAP-ENV:Body>
    <ns1:jePrvocisloResponse>
       <vysledek>true</vysledek>
    </ns1:jePrvocisloResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP (3)

- název hlavního tagu ve volání = název operace
- název hlavního tagu v odpovědi = název operace + *Response*
- názvy vnořených tagů = názvy vstupních popř. výstupních parametrů operace
- existují tzv. Faults, obdoba vyjímek v Javě a C++

SOAP fault

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
            xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
            xmlns:ns1="urn:mojeURI">
 <SOAP-ENV:Body>
   <SOAP-ENV:Fault>
    <faultcode>SOAP-ENV:Client</faultcode>
    <faultstring>Neplatny vstup</faultstring>
    <detail>
     <ns1:Vyjimka>
      <duvod>cislo musi byt &gt;= 2</duvod>
      <cislo>-3</cislo>
     </ns1:Vyjimka>
    </detail>
   </SOAP-ENV:Fault>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

WSDL – Web Service Description Language

- popisuje rozhraní služby
- jména operací, jména a typy parametrů a návratových hodnot
- kde a jak službu volat – HTTP/HTTPS, port, stroj, URL
- WSDL je jako *.h v Céčku, interface v Javě, nebo IDL
- nepopisuje sémantiku, pouze syntaxi
- zcela stačí pro volání služby, automatizované nástroje z něj umí vygenerovat ***stub*** – zástupný kód pro volání ve zvoleném prog. jazyku
- WSDL 1.1 je *W3C Note* od roku 2001
- WSDL 2.0 je *W3C Candidate Recommendation* od března 2006

struktura WSDL popisu

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="PrvniSluzba”  targetNamespace="urn:mojeURI”
  xmlns="http://schemas.xmlsoap.org/wsdl/“ ...  >
<types>
  ... definice datových typů ....
</types>

<message>
 ... definice komunikačních zpráv pomocí typů ...
</message>

<portType>
 ... definice operací pomocí komunikačních zpráv ...
</portType>

<binding> ... že se volá přes HTTP ... </binding>
<service> ... na jakém URL (stroji, portu) se volá ... </service>

</definitions>
```

WSDL - ukázka typů

```
<!-- fault element -->
 <element name="Vyjimka">
  <complexType>
   <sequence>
    <element name="duvod" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
    <element name="cislo" type="xsd:long" minOccurs="1" maxOccurs="1"/>
   </sequence>
  </complexType>
 </element>
 <!-- operation request element -->
 <element name="jePrvocislo">
  <complexType>
   <sequence>
    <element name="cislo" type="xsd:long" minOccurs="1" maxOccurs="1"/>
   </sequence>
  </complexType>
 </element>
 <!-- operation response element -->
 <element name="jePrvocisloResponse">
  <complexType>
   <sequence>
    <element name="vysledek" type="xsd:boolean" minOccurs="1" maxOccurs="1"/>
   </sequence>
  </complexType>
 </element>
```

WSDL - zprávy

```
<message name="jePrvocisloRequest">
 <part name="parameters" element="ns1:jePrvocislo"/>
</message>

<message name="jePrvocisloResponse">
 <part name="parameters" element="ns1:jePrvocisloResponse"/>
</message>

<message name="VyjimkaFault">
 <part name="fault" element="ns1:Vyjimka"/>
</message>
```

WSDL - operace

```
<portType name="Cisilka">
 <operation name="jePrvocislo">
  <documentation>Spocita, zda cislo je prvocislo</documentation>
  <input message="tns:jePrvocisloRequest"/>
  <output message="tns:jePrvocisloResponse"/>
  <fault name="Vyjimka" message="tns:VyjimkaFault"/>
 </operation>
</portType>
```

- jméno portType je v Javě použito pro název interface
- jméno operation je použito pro název funkce (v C) či metody (v Javě)
- input, output a fault odkazují na zprávy definující vstup, výstup, popř. vyjímky
- všechny tři jsou nepovinné
- v tagu documentation je popis lidskou řečí, lze použít jako komentář ve vygenerovaném stubu

Použití WSDL

- typový systém SOAP způsobuje problémy – nepoužívat
- datové typy v různých jazycích jsou nekompatibilní, kdežto XML zprávy jsou vždy stejné
- typový systém XML Schema je společná půda pro všechny programovací jazyky
- je dobré používat contract-first přístup – vždy začínat od WSDL

Styly WSDL

- historicky 4 styly WSDL
  - RPC/encoded
  - RPC/literal
  - document/literal
  - document/literal wrapped
- dnes se prosazuje doc/lit wrapped
- WS-I Basic Profile zakazuje RPC/encoded
- MS .NET podporuje pouze doc/lit wrapped

RPC/encoded

- určen pro volání operací, vznikl před dokončením XML Schema a WSDL
- typová informace v SOAP zprávách
- zprávu nelze snadno validovat vůči XML Schema
- zdroj nekompatibilit
- umožňuje cyklické odkazy mezi přenášenými objekty a polymorfismus (j:Jablko místo j:Ovoce)

WSDL:

```
<message name="myMethodRequest">
 <part name="x" type="xsd:int"/>
</message>
```

SOAP:

```
<soap:body>
  <myMethod>
    <x xsi:type="xsd:int">5</x>
  </myMethod>
</soap:body>
```

RPC/literal

- určen pro volání operací
- ušetří typovou informaci ve zprávách
- stále nelze snadno validovat zprávy
- neumožňuje cyklické odkazy a polymorfismus

WSDL:

```
<message name="myMethodRequest">
 <part name="x" type="xsd:int"/>
</message>
```

SOAP:

```
<soap:body>
  <myMethod>
    <x>5</x>
  </myMethod>
</soap:body>
```

document/literal

- určen pro přenos libovolného XML, včetně atributů
- ušetří typovou informaci ve zprávách
- obsah zpráv lze snadno validovat
- zmizel název operace !

WSDL:

```
<types>
 <schema>
  <element name="x" type="xsd:int"/>
 </schema>
</types>
<message name="myMethodRequest">
 <part name=„params" element="x"/>
</message>
```

SOAP:

```
<soap:body>
    <x>5</x>
</soap:body>
```

document/literal wrapped

- document/literal s názvem operace

WSDL:

```
<types>
 <schema>
  <element name="myMethod"/>
   <complexType>
    <sequence>
     <element name="x" type="xsd:int"/>
    </sequence>
   </complexType>
  </element>
 </schema>
</types>
<message name="myMethodRequest">
 <part name="parameters" element="myMethod"/>
</message>
```

SOAP:

```
<soap:body>
 <myMethod>
  <x>5</x>
 </myMethod>
</soap:body>
```

Změny WSDL

- změna WSDL je změnou rozhraní služby
- pokud je zpětně kompatibilní (přidání operací), lze ponechat namespace
- při změně stávajících operací je doporučováno změnit namespace
- pak je možno provozovat i více verzí stejné služby na stejném URL, tj. obsluhovat staré i nové klienty

Pomněnka pro WSDL

- používejte jen wrapped document/literal !
- začínejte vždy od WSDL, nikdy od funkce/metody v nějakém programovacím jazyce ! (Java2WSDL je špatnost)
- proč ? protože
  - datové typy nejsou přenositelné mezi různými programovacími jazyky
  - XML je přenositelné mezi různými prog. jazyky

WS-I

- specifikace SOAP 1.x a WSDL 1.1 jsou místy vágní, potíže s interoperabilitou
- vznikla organizace WS-I Web Services Interoperability Organization www.ws-i.org
- BasicProfile 1.0 (2004) a 1.1 (2006) vyjasňuje sporná místa a zakazuje některé rysy
- např. zakazuje SOAP encoding, DTD nebo Processing Instructions v XML, vyžaduje WSDL 1.1 a SOAP 1.1
- Basic Security Profile je ve vývoji

Nástroj gSOAP

- autor Robert van Engelen, Florida State University, Genivia Inc.
- nejrychlejší, nejoblíbenější pro akademické použití, zdarma
- optimalizováni na výkon – např. zásobník syntaktických analyzátorů pro konkrétní zprávy, minimalizace kopírování v paměti, atd.
- generátor zdrojových kódů pro C/C++
- program *wsdl2h* z WSDL popisu služby vygeneruje speciální .h soubor
- program *soapcpp2* z .h vygeneruje stub v C nebo C++
- lze z .h vygenerovat WSDL popis služby

Nástroj Apache Axis

- projekt z rodiny Apache, zdarma, Java
- součásti
  - knihovny pro komunikaci
  - nástroj WSDL2Java
  - nástroj Java2WSDL
  - servletová aplikace pro umístění serverové části služby
- umožňuje sestavit SOAP volání dynamicky, ale je 9-12x (IBM 1.4) resp. 13-15x (SUN 1.5) pomalejší než gSOAP
- (243 resp. 163 versus 2060 msg/s na Pentium4 2.5GHz)
- (310 resp. 230 versus 3600 msg/s na AMD FX-53)

Vyhledávání služeb

- umístění služby je popsáno ve WSDL
- jak však nalezneme WSDL ?
- můžeme ho od někoho dostat
- můžeme ho vyhledat
- UDDI – Universal Description, Discovery and Integration
- WSIL – Web Service Inspection Language

UDDI

- iniciativa z roku 2000 publikovaná na uddi.org
- centralizovaný seznam služeb (white pages, yellow pages) s vyhledáváním i podle oborů
- verze 2 v roce 2001, verze 3 v roce 2002
- verze 3.0.2 jako OASIS standard v roce 2005
- veřejné UDDI rejstříky (IBM, Microsoft, SAP) vypnuty v lednu 2006
- většina záznamů veřejného UDDI byla špatných
- velmi obecné (tModels), jako seznam webových služeb nepříliš praktické
- nejdřív výběr služby, pak obchodního partnera

WSIL

- jednoduchý XML formát pro seznam webových služeb dané instituce
- IBM a Microsoft, listopad 2001
- soubor http://nekde.com/inspection.wsil
- nejdřív výběr obchodního partnera, pak služby

WS-* džungle

- existuje přes 30 dalších a soupeřících dodatečných specifikací
- WS-Addressing, WS-Eventing, WS-Transfer, WS-Policy, WS-ReliableMessaging, ...
- mají smysl ve složitých enterprise systémech
- pro jednoduché Internetové aplikace mnohdy stačí XML-over-HTTP nebo REST

WS-*

- WS-Addressing
  - přidává do SOAP hlavičky informace obdobné e-mailovým From:, To:
- WS-Policy a WS-SecurityPolicy
  - rámce pro popisy vlastností, které nejde popsat ve WSDL
- WS-Transfer
  - obdoba protokolu HTTP nad SOAP, umožňuje operace Get, Put, Create
- WS-Eventing (MS), WS-Notification (IBM)
  - zasílání událostí, budou spojeny do nové specifikace WS-EventNotification

Plain Old XML

- amazon.com má dvě rozhraní ke službám
  - SOAP – 20% provozu
  - jednoduché XML dotazy – 80% provozu
- XML+HTTP poskytují všechna prostředí
- SOAP+WS-* potřebují spec. nástroje
- webovým aplikacím stačí XML+HTTP
- podnikové aplikace potřebují SOAP+WS-*

REST

- REpresentational State Transfer
- styl architektury webových aplikací
- veškerá informace v požadavku, bezestavovost
- Roy Fielding, disertace, rok 2000
- webový zdroj je XML adresované přes URI
- HTTP metody PUT,GET,POST,DELETE
- zdroje jsou podstatná jména, metody slovesa
- škáluje velmi dobře

REST versus SOAP

- SOAP služba pro zjištění teploty
  - operace getTemperature
  - parametr „city“
- REST služba generující odpovědi pro různá URL obsahující města
  - http://pocasi.cz/teplota/Brno
  - http://pocasi.cz/teplota/Praha
- WSDL 2.0 umožňuje popsat i REST služby s parametry v URL
- zdroje orientované na činnost se lépe vyjadřují jako webové služby, zdroje poskytující data se lépe vyjadřují pomocí REST

Grid, OGSA

- Grid je infrastruktura pro sdílení zdrojů nepodléhajících centralizované správě
- problémy s heterogenitou prostředí
- OGSA – Open Grid Services Architecture
  - grid založený na webových službách
  - potřebuje správu životního cyklu a stavu
  - potřebuje notifikace
- 1. verze – OGSI – webové služby jako objekty
- 2. verze – WSRF – webové služby + WS-Resources
- služby jsou bezestavové, zdroje mají stav a cyklus
- 3. verze – sloučení s WS-Transfer/WS-Eventing

Bezpečnost

- SOAP a WSDL ji neřeší
- na transportní vrstvě – HTTP nad SSL
  - rychlé, funkční, odzkoušené
  - nelze zpětně prokazovat, co kdo zaslal
  - pouze dva komunikující body
- na úrovni zpráv – XML Encryption, XML Signature
  - lze podepisovat
  - lze budovat řetězce zpracovatelů zpráv
- WS-Security
  - integrace s PKI, Kerberos, hesly, ...
  - zatím neusazená technologie
  - nelze popsat ve WSDL, je třeba WS-Policy, WS-SecurityPolicy

Co jsou tedy webové služby ?

- z pohledu webu a jeho aplikací
  - důležité jsou HTTP,XML, adresovatelnost URI
  - SOAP je nadbytečnou komplikací
- z pohledu enterprise aplikací
  - důležité je SOAP pro jeho hlavičky a nezávislost na transportním protokolu
  - HTTP není nutné
- shoda – strojová interakce pomocí XML

ESB

- Enterprise Service Bus
- pojem zaveden v prosinci 2002 v předpovědi firmy Gartner na rok 2003
- pokud všechny enterprise aplikace jsou služby, lze je propojit sběrnicí pro zasílání zpráv
- obchodní procesy jsou dány a měněny nastavením ESB, ne změnami služeb
- větší flexibilita propojení služeb

WSBPEL

- Web Services Business Process Execution Language
- nástupce BPEL4WS (2002) od IBM, MS, BEA
- jazyk (založený na XML) pro zápis procesů volajících webové služby
- orchestrace webových služeb
- proměnné, programovací struktury (if-then-elseif-else, while, paralelismus), fault-handlers, event-handlers, compensation-handlers
- pro vykonání potřebuje BPEL engine

Semantic Web Services

- ***„Sémantický web*** *je rozšíření současného webu, ve kterém je informacím přiřazen dobře definovaný význam, což umožní lidem a počítačům lépe spolupracovat“*
- Tim Berners-Lee (vynálezce WWW) et al, Scientific American, 2001
- všechna data na webu budou sémanticky označkována a tudíž přístupna strojovému zpracování
- postupy sem. webu lze aplikovat i u webových služeb

Ontologie

- pojem z oboru umělé inteligence
- **výslovný formální zápis konceptualizace**
- konceptualizace je systém pojmů modelující určitou část světa
- jazyk W3C OWL umožňuje zapisovat ontologie jako soubory tříd, individuí a vlastností
- sémantický web je spíš „logický web“
- při omezení na logiku DL (Description Logic) lze automatizovaně odvozovat

Semantic Web Services (2)

- současné (konkurující si) aktivity
  - WSMO – Web Services Modeling Ontology
  - WSDL-S – WSDL s možností přidávat sémantické odkazy
  - OWL-S (OWL for Services)
- může značně vylepšit automatizované vyhledávání webových služeb
- ve fázi výzkumu

Konec

- Dokumentace viz Google, hesla gSOAP, Apache Axis, WSDL, SOAP atd.
- Děkuji za pozornost