Tutoriál Web Services

Martin Kuba makub@ics.muni.cz

Superpočítačové Centrum Brno,
Masarykova Univerzita

Obsah tutoriálu

- definice Web Services
- URI, XML Schema a Namespaces
- protokol SOAP, jazyk WSDL
- první webservice v C i Java
- druhá složitější webservice v C i Java
- cvičení – udělejte si klienta z WSDL
- styly WSDL
- adresace služeb
- bezpečnost

Co jsou Web Services

- technologie pro vzdálené volání procedur (a výměnu XML zpráv)
- podobná RPC, CORBA, Java RMI
- tři části
  - komunikační protokol SOAP (Simple Object Access Protocol)
  - popis služby ve WSDL (Web Service Description Language)
  - vyhledání služby (UDDI,WSIL)

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>
```

klient

SOAP

```
<soap:body>
 <najdiCloveka>
  <rodneCislo>651002/3810</rodneCislo>
 </najdiCloveka>
</soap:body>
```

SOAP

```
<soap:body>
 <najdiClovekaResponse>
 <jmeno>Josef Novák</jmeno>
 <vek>40</vek>
</najdiClovekaResponse>
</soap:body>
```

web service

.najdiCloveka(rc)

.pridejCloveka(Clovek)

.smazCloveka(rc)

Přirozený historický vývoj

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

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

- 2. krok – metoda POST s URL parametry
  - omezení na pouze páry řetězců

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

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

Přirozený historický vývoj (2)

- 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
- r. 2000 přijato W3C SOAP 1.1 jako *Note*
- r. 2003 vydalo W3C SOAP 1.2 jako *Recommendation*
- r. 2003 WS-Interoperability www.ws-i.org vydalo BasicProfile 1.0

Výhody Web Services

- 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

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>
```

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)

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)

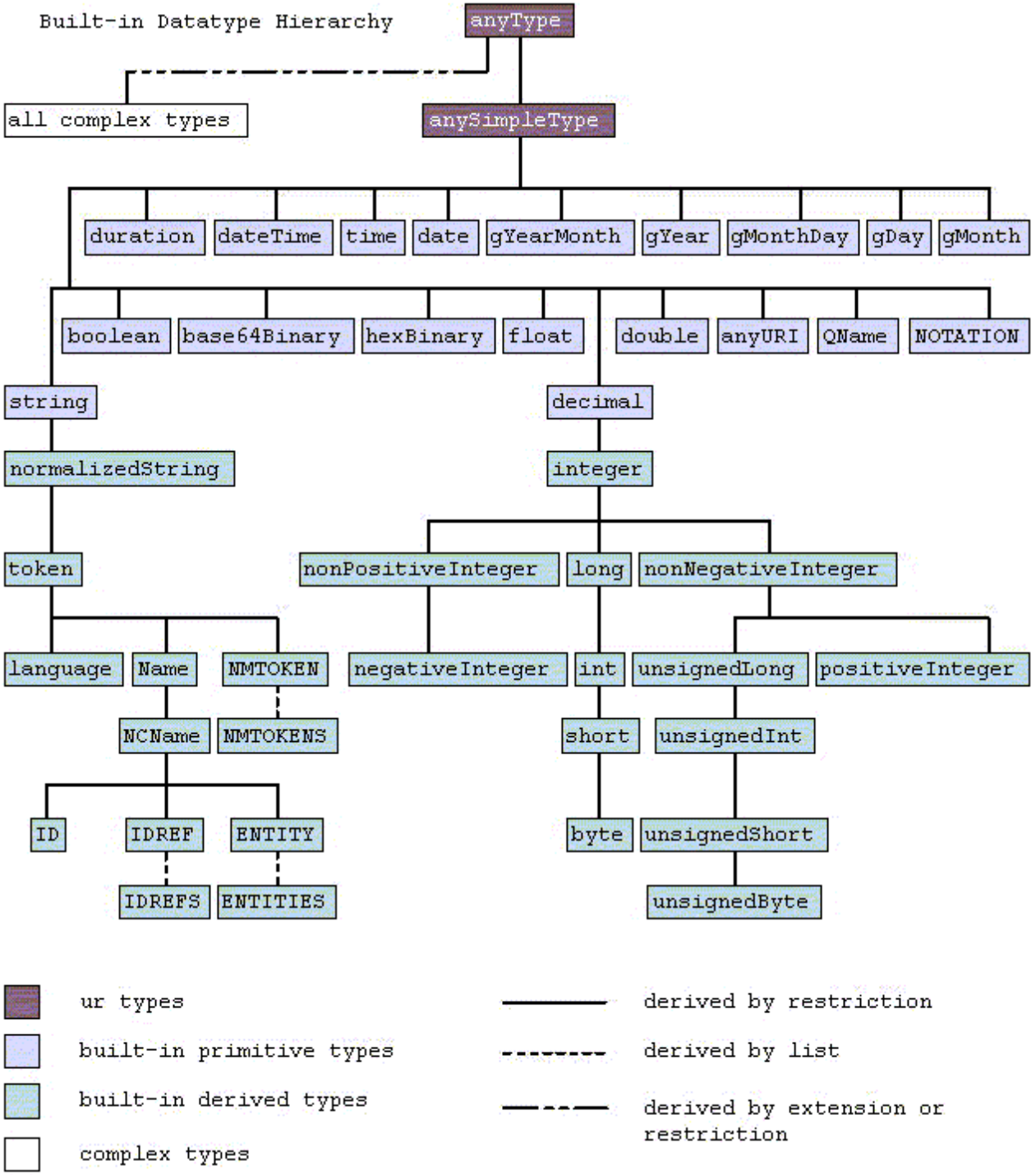
```
<?xml version="1.0" encoding="UTF-8" ?>
<koren xmlns:umělecký="urn:Michelangelo“
      xmlns:pohlavní="http://www.sex.cz/"
      xmlns:divadelní="http://www.divadlo.cz/hra"
      xmlns:pietní="http://www.krematorium.cz/"
      xmlns:u="urn:Michelangelo“
      xmlns="http://urad.cz/akta" >

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" >
<!-- Objekty -->
 <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="prijmeni" type="xsd:string"/>
       <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 v CORBA
- 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 *Recommendation* od roku 2001

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/metody
- 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

Praktická ukázka

- potřebujete
  - gSOAP 2.7.1
  - gcc (g++,bison,flex pro kompilaci gSOAP)
- z gSOAP potřebujete jen
  - soapcpp2 – generátor stubů
  - wsdl2h – parser WSDL
  - stdsoap2.h a stdsoap2.c (přiloženy v příkladech)

Co je gSOAP

- autor Robert van Engelen, Florida State University, Genivia Inc.
- nejrychlejší, nejoblíbenější pro akademické použití, zdarma
- 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++ , a WSDL popis služby

Instalace gSOAP

- z http://www/~makub/ stáhněte gsoap-*
- předkompilováno pro Linux a Windows
- ze src - gsoap_2.7.1.tar.gz
  - *tar xzvf gsoap_2.7.1.tar.gz*
  - *cd gsoap-2.7*
  - *./configure –prefix=$HOME/gsoap-2.7.1*
  - *make; make install*
  - v *$HOME/gsoap-2.7.1/bin* jsou binárky *soapcpp2* a *wsdl2h*

První webservice v C

- kompilace:
  - *tar xzvf prvni_C.tgz*
  - *cd prvni*
  - *vi PrvniSluzba.wsdl klient.c server.c*
  - *#v Makefile zadejte kde je gSOAP*
  - *make*
- z WSDL vznikne prvni.h, z něj knihovny
- spustit ./server a ./klient

Tipy a triky pro gSOAP

- pokud přeložíte *stdsoap2.c* s *–DDEBUG*, v souborech *SENT.log, RECV.log* budou SOAP zprávy
- WSDL lze vygenerovat, viz *prvni_orig.h*
- kolečko .h -> WSDL -> .h produkuje kompatibilní klienty, ale ne kompatibilní názvy funkcí a struktur v C/C++

Apache Axis – Java web services

- projekt z rodiny Apache, zdarma
- součásti
  - knihovny pro komunikaci
  - nástroj WSDL2Java
  - nástroj Java2WSDL
- 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)
- instalace - z http://www/~makub/ stáhněte *axis-bin-1_2RC3.tar.gz* a rozbalte, do *lib/* přidejte *mail.jar* a *activation.jar*

Klient v jazyce Java

- kompilace:
  - *tar xzvf prvni_java.tgz*
  - *cd prvniJava*
  - *vi make.sh #nastavit cesty k Axis a java,javac*
  - *. make.sh*
- podívejte se na vygenerovaný *prvni/Cisilka.java*
- nutný spuštěný server z gSOAP
- spustit:
  - *java VolejSluzbu -3*
- ze SOAP Fault je vytvořena Java vyjímka

Server pomocí Axis/Java

- nutné nainstalovat servlet container (TomCat) a webaplikaci z Axis
- vygenerovat z WSDL server-side stub, implementovat serverovou část
- deploy – umístění služby na server
- viz http://www:8080/axis/
- v jednom kontejneru je více služeb
- klient může použít jiný server se stejnou službou specifikováním jiného URL

Plnotučná služba v C i Java

- služba demonstrující typy (pole, ukazatele, výčet, binární data, čas, složený typ), více výstupních parametrů
- definice typů v *druha/druha.h*
- stejný postup pro *druha_C.tgz* a *druha_Java.tgz*
  - *tar xzvf druha_C.tgz; tar xzvf druha_Java.tgz*
  - *cd druha*
  - *vi Makefile #upravit cestu k gSOAP*
  - *make; ./server & ./klient*
  - *cd ../druhaJava*
  - *vi make.sh #upravit cestu k Axis a javac*
  - *. make.sh*
- v *druha/RECV.log* uvidíte rozdíl mezi gSOAP a Axis

Cvičení

- napište klienta pro službu
- http://www:8080/axis/services/PsaciSluzba?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 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

- výhody doc/lit a obsahuje název 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>
```

Adresace

- v jednom serveru může být více služeb
- v gSOAP pouze jedna
- nalezení služby v Apache Axis
  - podle cesty v URL, např. ***/axis/services/Moje***
  - podle namespace operace, např. ***urn:moje***
  - podle HTTP hlavičky ***SOAPAction:***
  - nějak jinak (custom handler)
- tj. nechytněte z gSOAP zlozvyk adresovat jen strojem a portem, je třeba celý <binding> z WSDL

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

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, WS-Security
  - lze podepisovat
  - lze budovat řetězce zpracovatelů zpráv
  - asi 200x pomalejší
  - ve stadiu vývoje

Konec

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