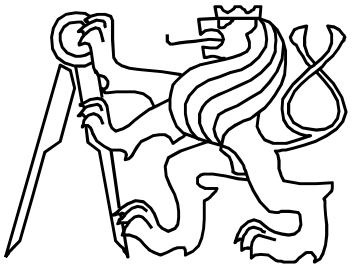
Enterprise Java (BI-EJA)
Technologie programování v jazyku Java (X36TJV)

Ing. Zdeněk Troníček, Ph.D.

Katedra softwarového inženýrství

Fakulta informačních technologií ČVUT v Praze



Letní semestr 2010/2011, přednáška č. 7
https://edux.fit.cvut.cz/courses/BI-EJA
https://edux.feld.cvut.cz/courses/X36TJV

Agenda

- Java API for XML Binding (JAXB)
- Webové služby
- Java API for XML Web Services (JAX-WS)
- Java API for RESTful Web Services (JAX-RS)

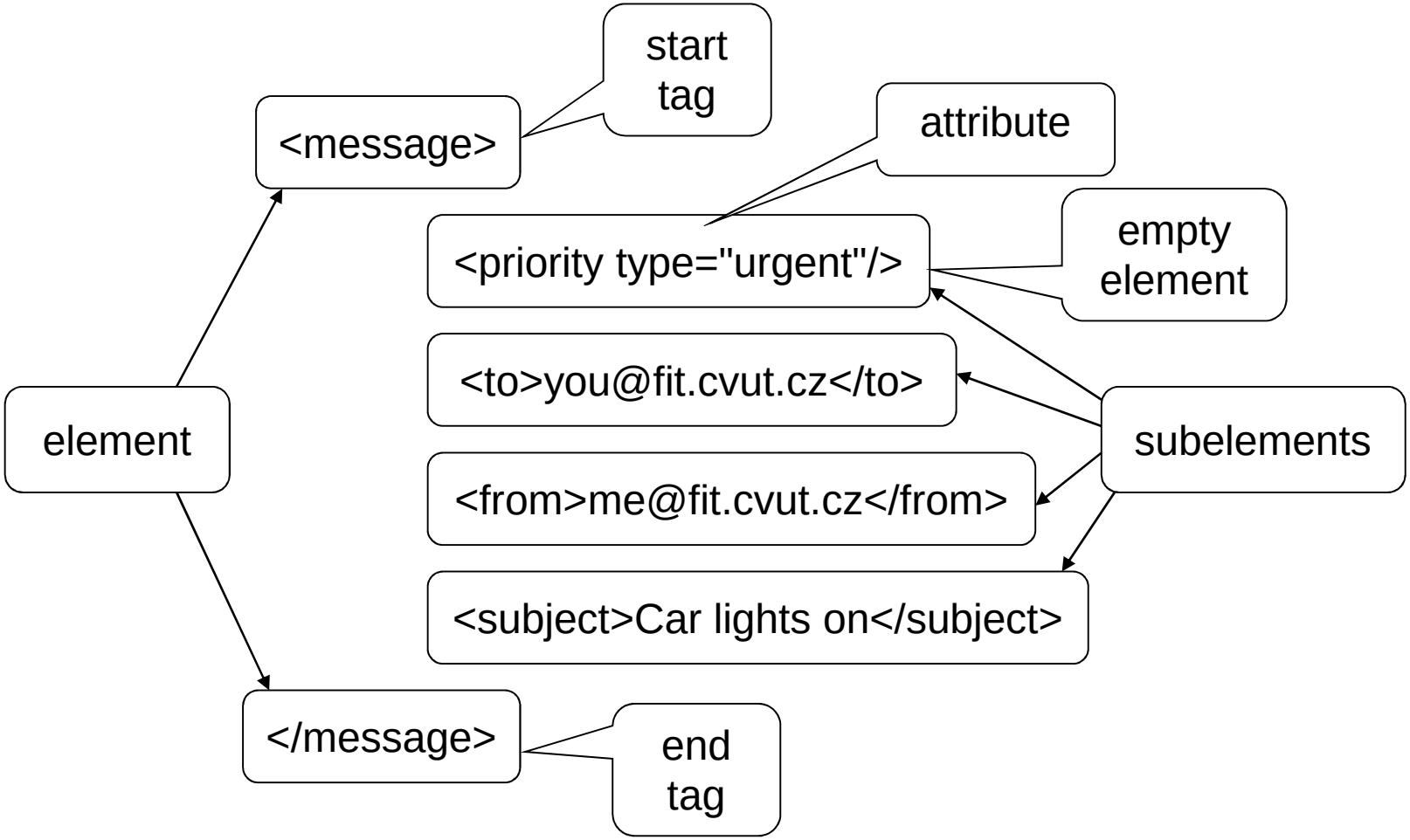
XML (opak.)

- značkovací jazyk stejně jako HTML
- všechny tagy musí být uzavřeny a nesmí se křížit
- hodnoty atributů musí být uzavřeny v uvozovkách nebo apostrofech
- rozlišují se velká a malá písmena
- přípustné značky definuje např. DTD nebo XML schéma

Znakové entity: & ' " < >

Komentáře: <!-- This is a comment -->

Terminologie



Well-formed & Valid XML

Well-formed

splňuje pravidla XML syntaxe

Valid

well-formed a podle připojeného DTD nebo XML schématu

Parser

- validating
- non-validating

Document Type Definition (DTD)

```
<!DOCTYPE articles [
<!ELEMENT articles (article*)>
<!ELEMENT article (articledata+)>
<!ELEMENT articledata (title, author)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
]>
```

XML schéma

- umožňuje přesnější definici struktury než DTD
- obsahuje podporu typů, např. string, date, time, decimal

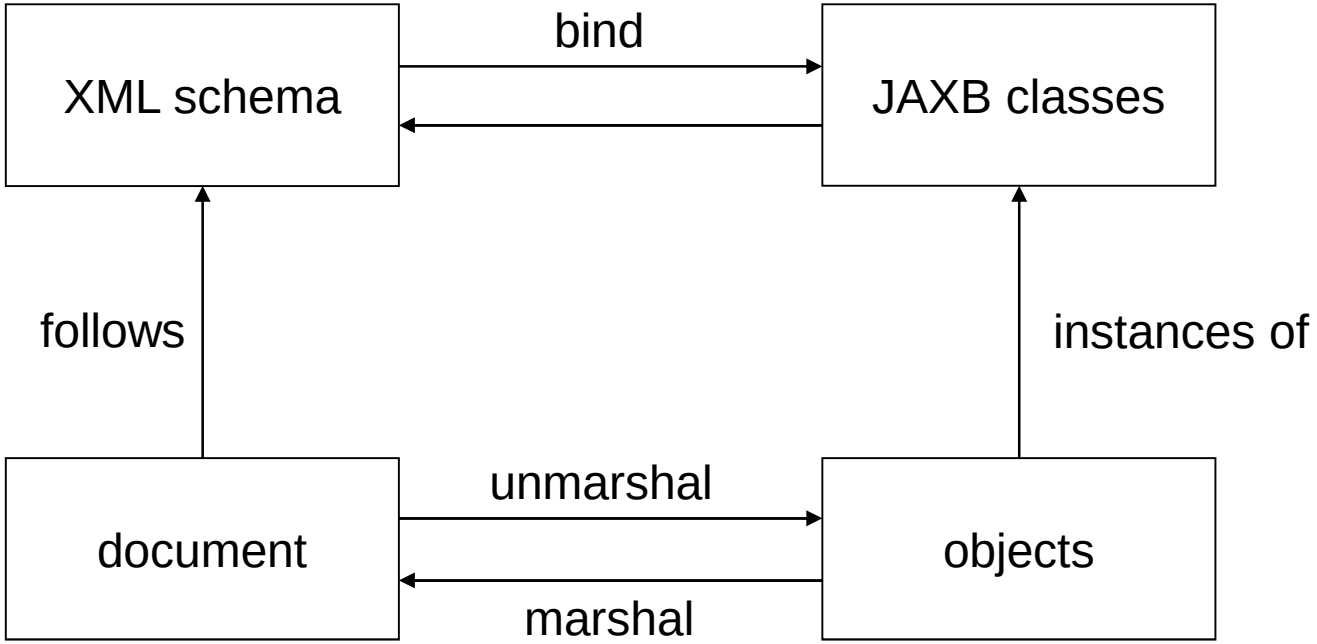
telefonní číslo ve tvaru XX-XXXX

```
<xsd:simpleType name="phone">
   <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d{2}-\d{4}"/>
   </xsd:restriction>
</xsd:simpleType>
```

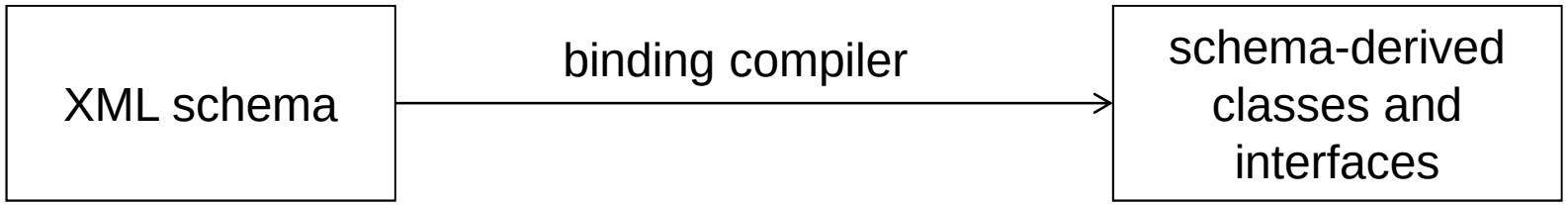
omezení rozsahu na interval <1,100)

```
<xsd:simpleType name="age">
   <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="1"/>
      <xsd:maxExclusive value="100"/>
   </xsd:restriction>
</xsd:simpleType>
```

Java API for XML Binding (JAXB)



Binding Compiler



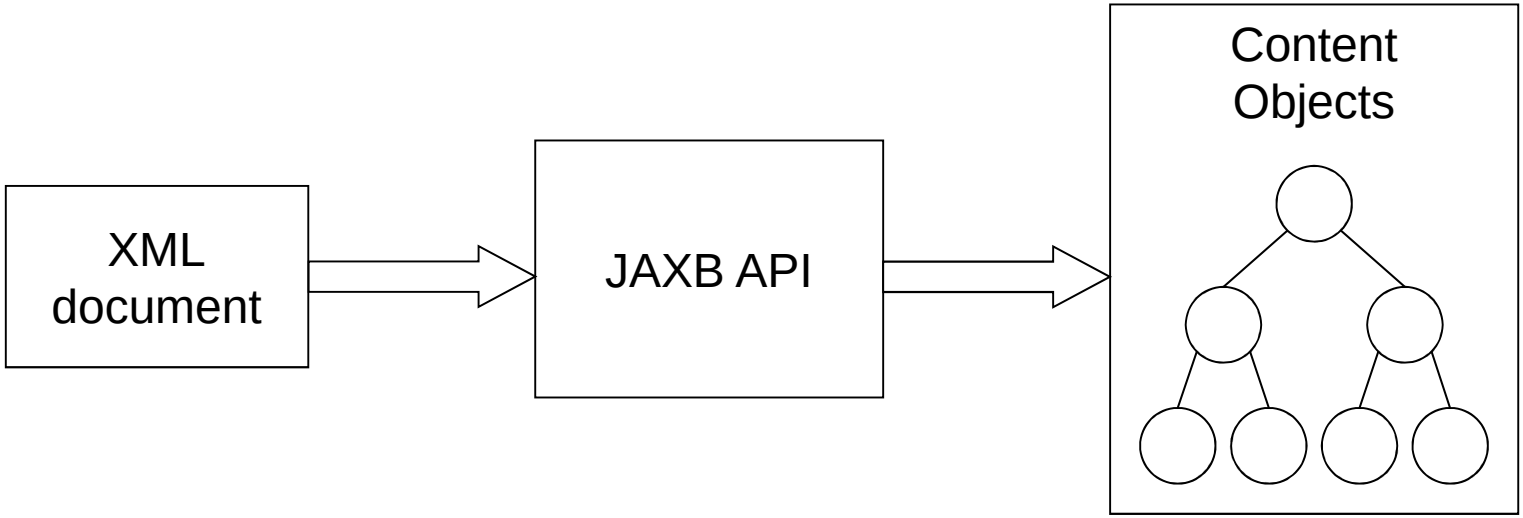

```
<complexType name="ItemType">
   <xsd:sequence>
      <xsd:element name="name"
        type="xsd:string"/>

   ...

</complexType>
```

```
public class ItemType {
    public String getName()
    public void setName(String n)
    ...
}
```

Mapping

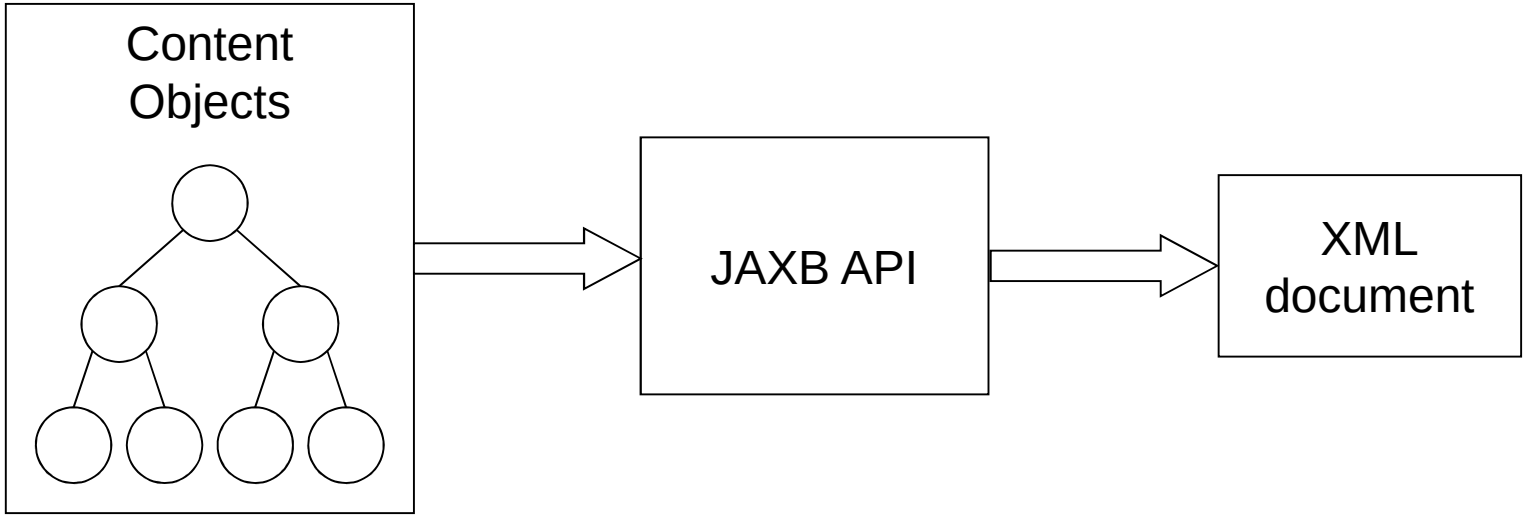
```
@XmlRootElement
@XmlType( propOrder = {"name", "email", "phones"} )
public class Customer {

    @XmlElement
    public String getName() {
        return name;
    }

    @XmlList
    public List<String> getPhones() {
        return phones;
    }
    …
}
```

Unmarshalling




```
JAXBContext ctx = JAXBContext.newInstance( "my.package" );
Unmarshaller u = ctx.createUnmarshaller();
AddressBookType addrBook =
    (AddressBookType) u.unmarshal( new File( "addressbook.xml" ));
```

Mashalling




```
Marshaller m = ctx.createMarshaller();
m.setProperty( Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE );
m.marshal( addrElement, System.out );
```

Webová služba

"A Web Service is a software component that is described via WSDL and is capable of being accessed via standard network protocols such as but not limited to SOAP over HTTP."

- distributed, loosely coupled, self-contained unit
- self-describing
- published, located, and accessed over the Internet
- language and platform neutral
- relies on standard Internet and XML-based protocols

Srovnání s RMI

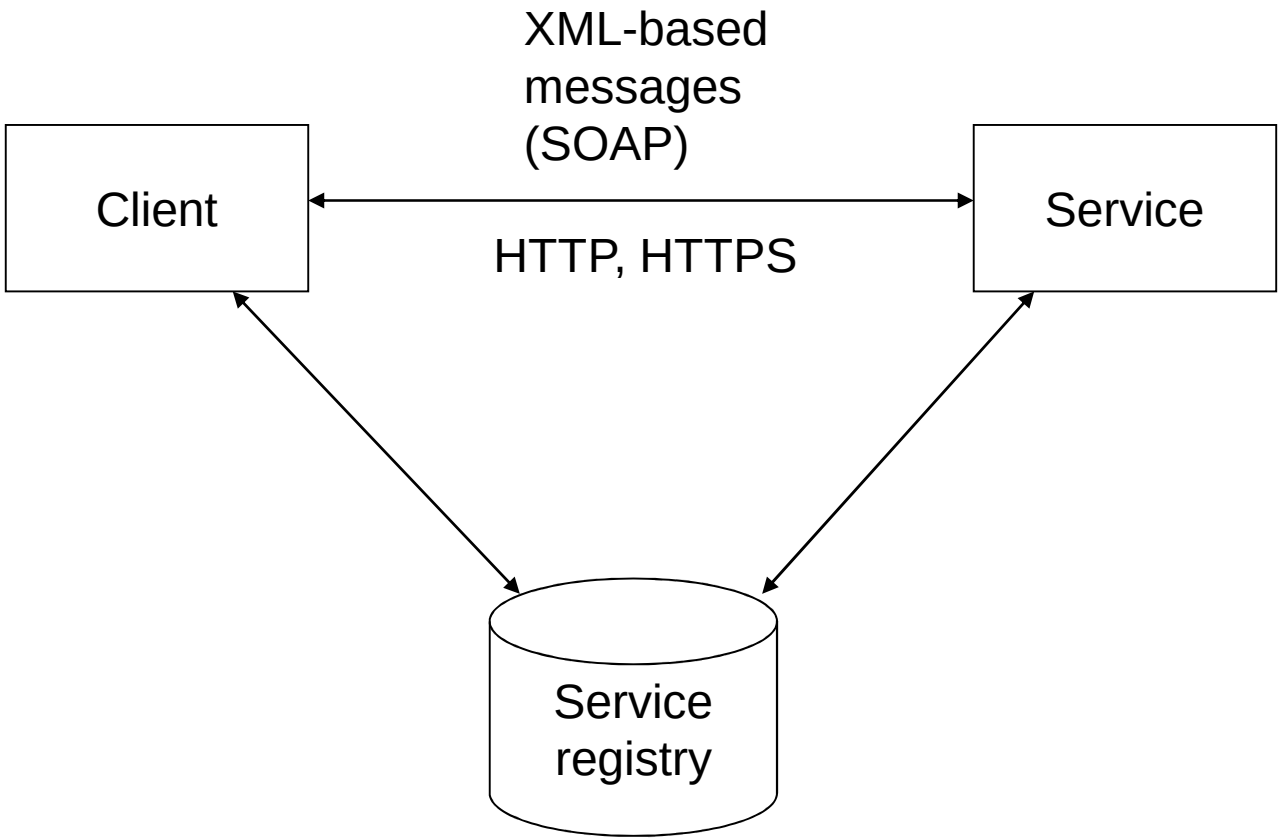
Remote Method Invocation

- within enterprise
- tied to a set of languages
- tightly-coupled
- firewall-unfriendly
  (to some extent)
- efficient processing

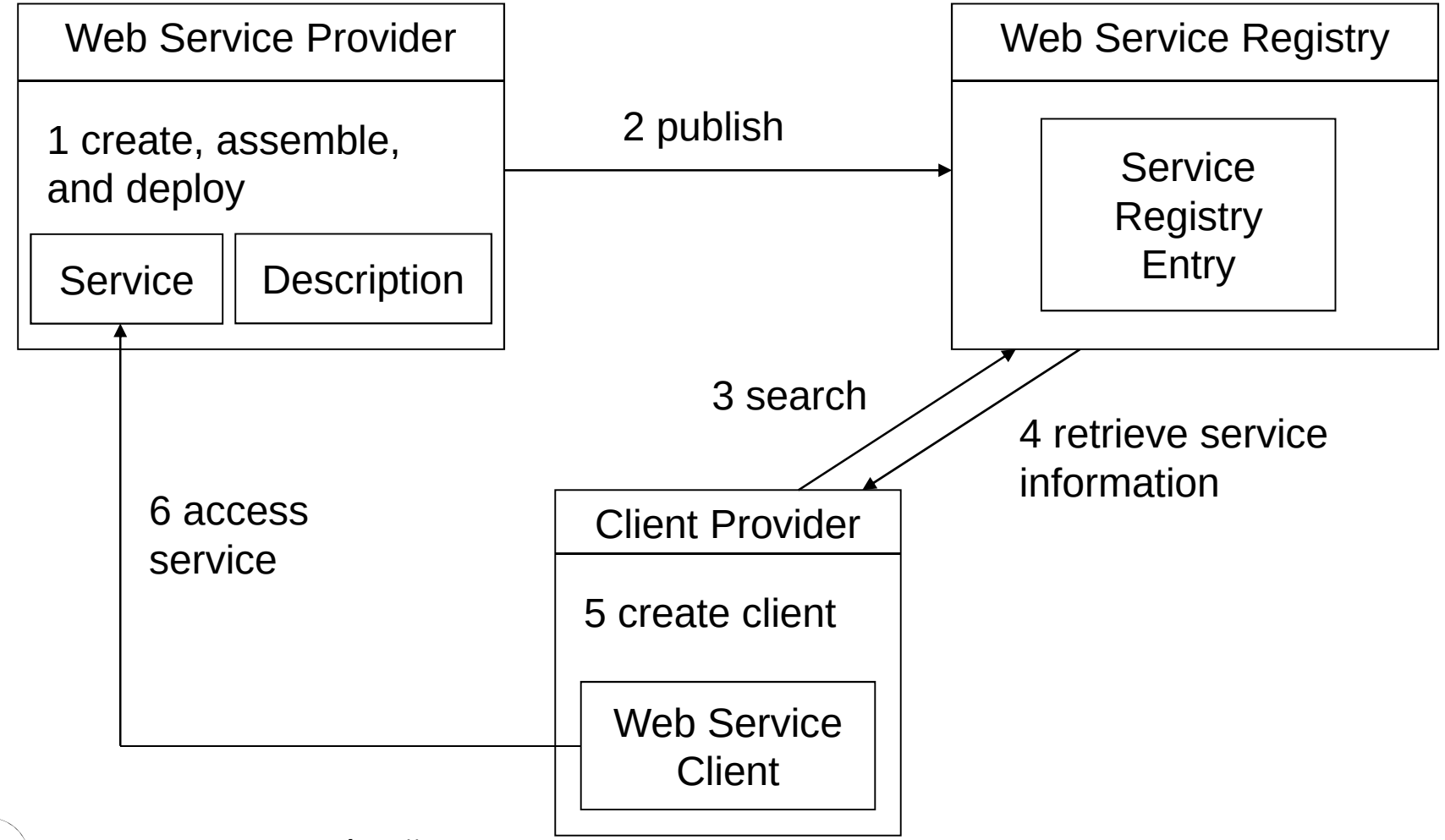
Web services

- between enterprises
- language independent
- loosely-coupled
- firewall-friendly
- inefficient processing

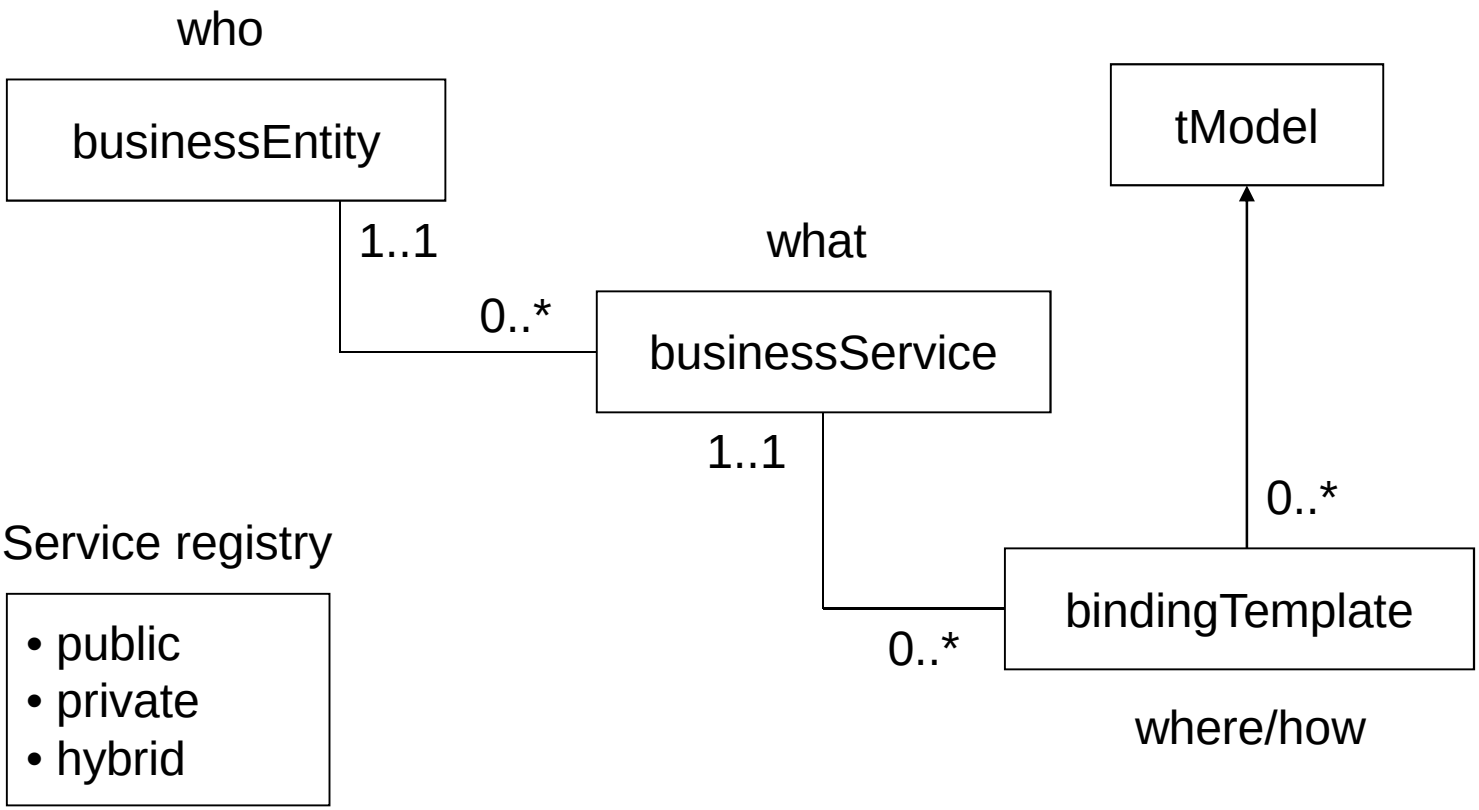
Web Service Elements



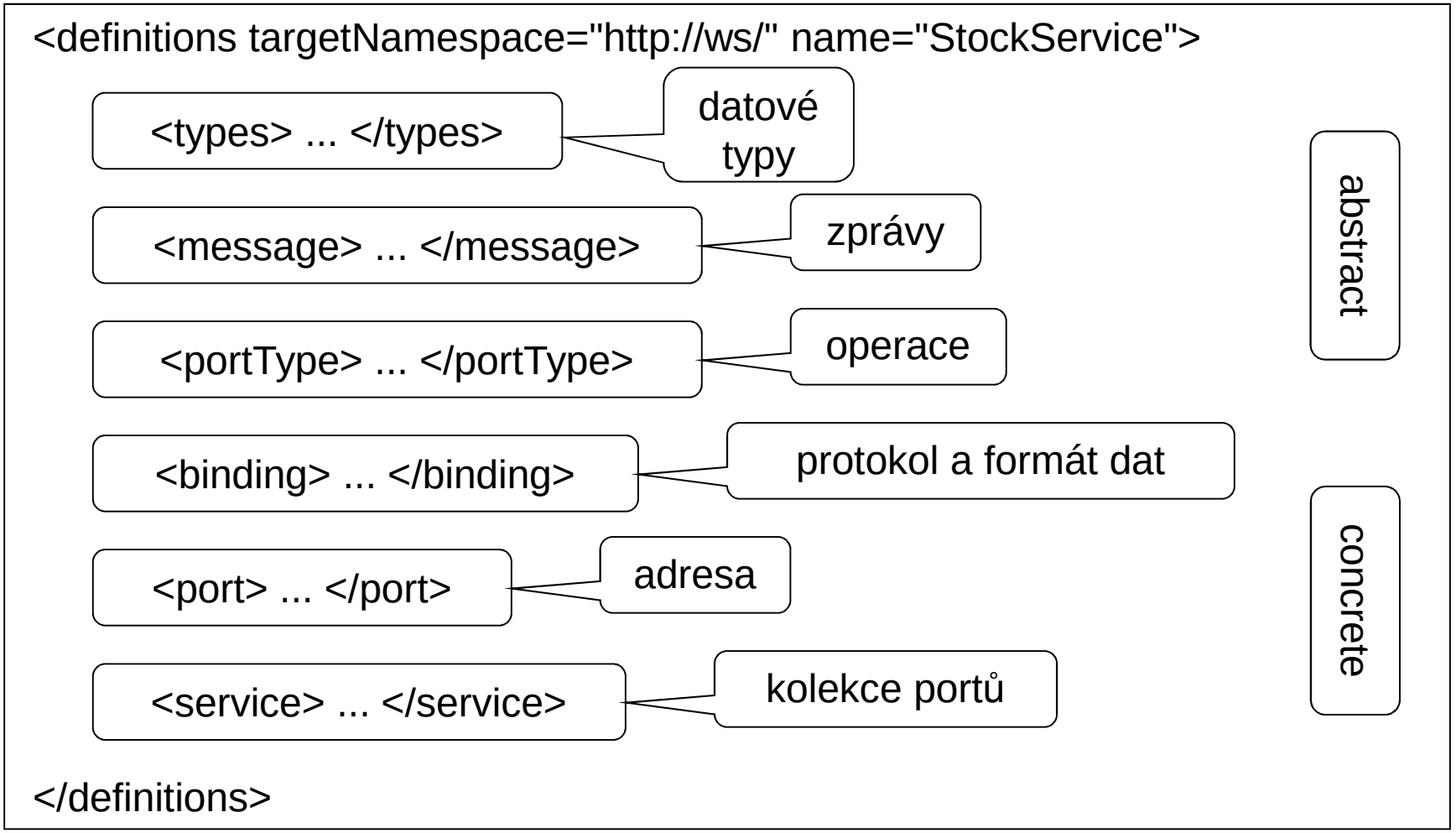
Životní cyklus



UDDI


who
businessEntity
1..1
0..*
what
businessService
1..1
0..*
tModel
0..*
bindingTemplate
where/how
Service registry
• public
• private
• hybrid

Web Service Description Language



WSDL – příklad(1)

```
<xs:complexType name="getQuote">
    <xs:sequence>
        <xs:element name="stock" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="getQuoteResponse">
    <xs:sequence>
        <xs:element name="return" type="xs:int"/>
    </xs:sequence>
</xs:complexType>
```

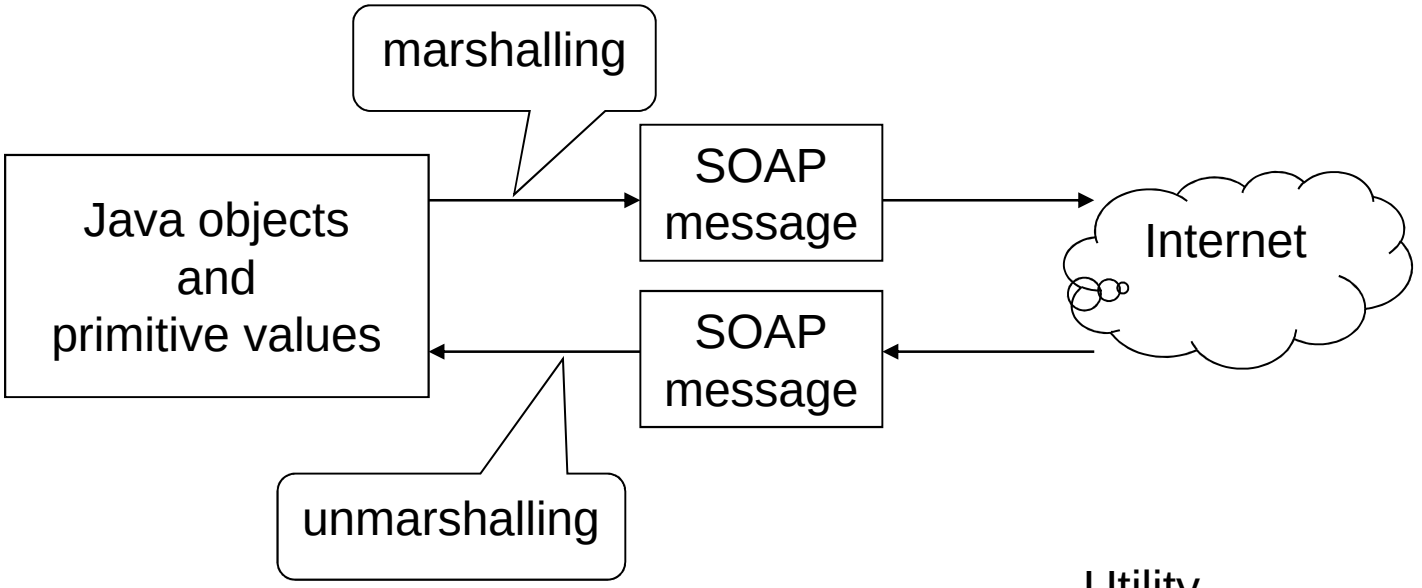
WSDL – příklad(2)

```
<message name="getQuote">
    <part name="parameters" element="tns:getQuote"/>
 </message>
<message name="getQuoteResponse">
    <part name="parameters" element="tns:getQuoteResponse"/>
 </message>
<portType name="StockExchange">
    <operation name="getQuote">
        <input message="tns:getQuote"/>
        <output message="tns:getQuoteResponse"/>
    </operation>
 </portType>
```

WSDL – příklad(3)

```
<binding name="StockExchangePortBinding" type="tns:StockExchange">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
      style="document"/>
    <operation name="getQuote">
        <soap:operation soapAction=""/>
        <input><soap:body use="literal"/></input>
        <output><soap:body use="literal"/></output>
    </operation>
</binding>
<service name="StockExchangeService">
  <port name="StockExchangePort" binding="tns:StockExchangePortBinding">
    <soap:address location="http://localhost:1234/StockWS/StockExchangeService"/>
  </port>
</service>
```

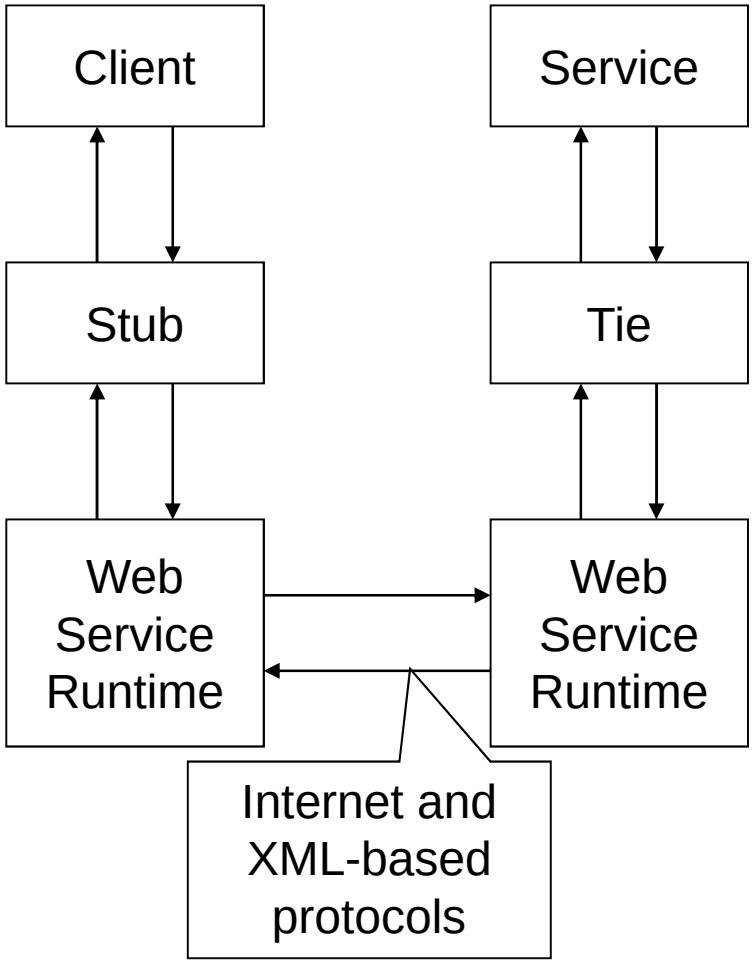
SOAP



Utility

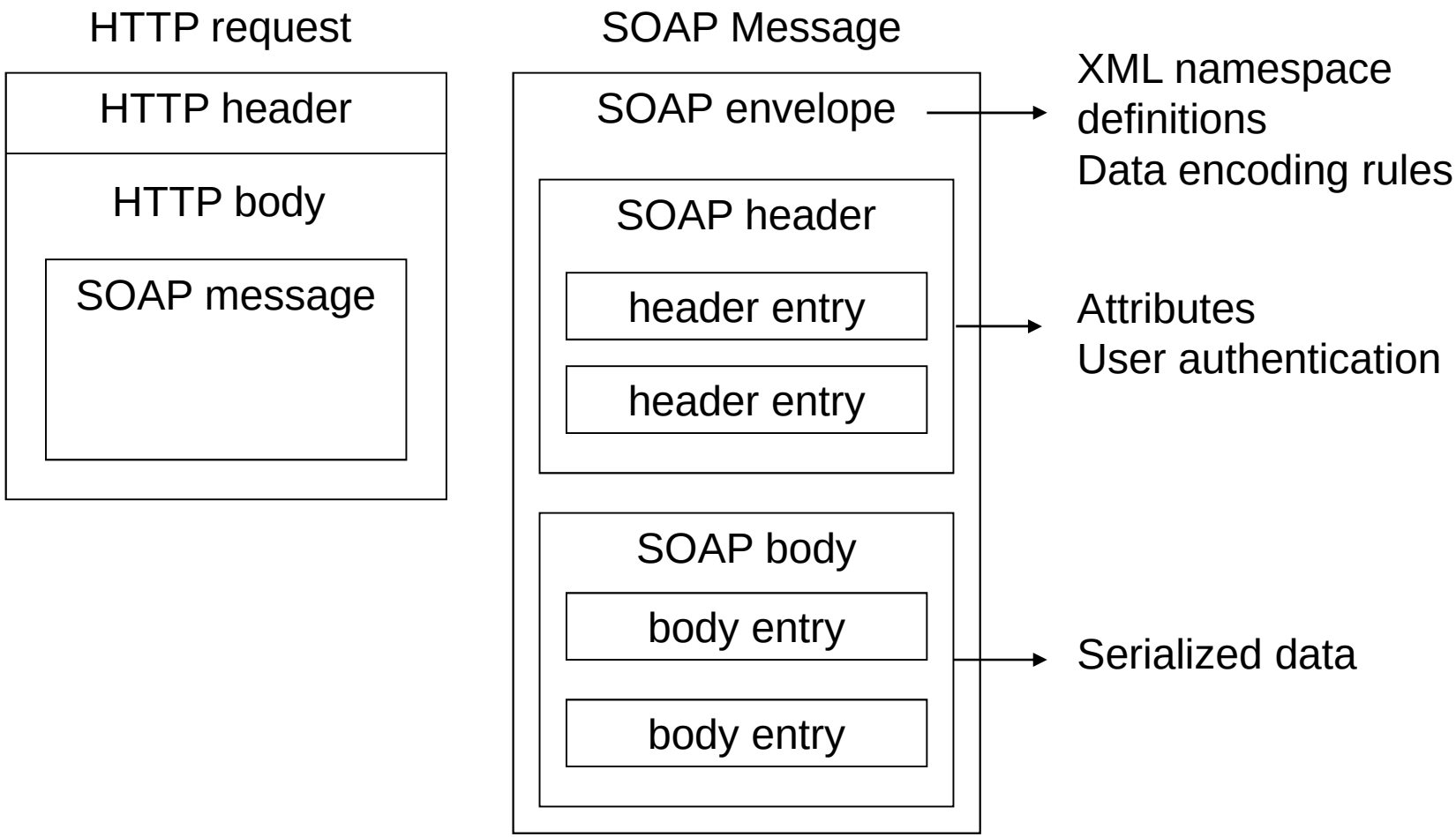
- wsgen
- wsimport

Java API for XML Web Services (JAX-WS)




```
@WebService
public class StockExchange {
    @WebMethod
    public BigDecimal getQuote(
        @WebParam( name = "stock" )
        String stock ) {
        ...
    }
}
```

SOAP over HTTP



Representational State Transfer (REST)

- *architecture style*
- alternativa k SOAPu
- založeno na pojmu *resource*
- každý resource je identifikován jednoznačným URI
- HTTP slouží jako aplikační protokol

HTTP method	action
POST	CREATE
GET	RETRIEVE
PUT	UPDATE
DELETE	DELETE

REST: postup

1. identifikujeme zdroje (resources)
2. každému zdroji přiřadíme jednoznačné URI
3. pro každý zdroj zvolíme příkazy (GET, POST, …)
4. vytvoříme XML reprezentaci zdrojů (zdroje propojujeme odkazy)

Příklad: knihovna

Zdroje: knihy, kategorie, …

Příklad: knihovna (1)

resource	URI	method	representation
kniha	book/[id]	GET PUT DELETE	popis knihy popis knihy
kategorie	category/	GET PUT	seznam kategorií seznam kategorií
knihy v dané kategorii	category/ [kategorie]	GET PUT POST	seznam knih seznam knih seznam knih

Příklad: knihovna (2)

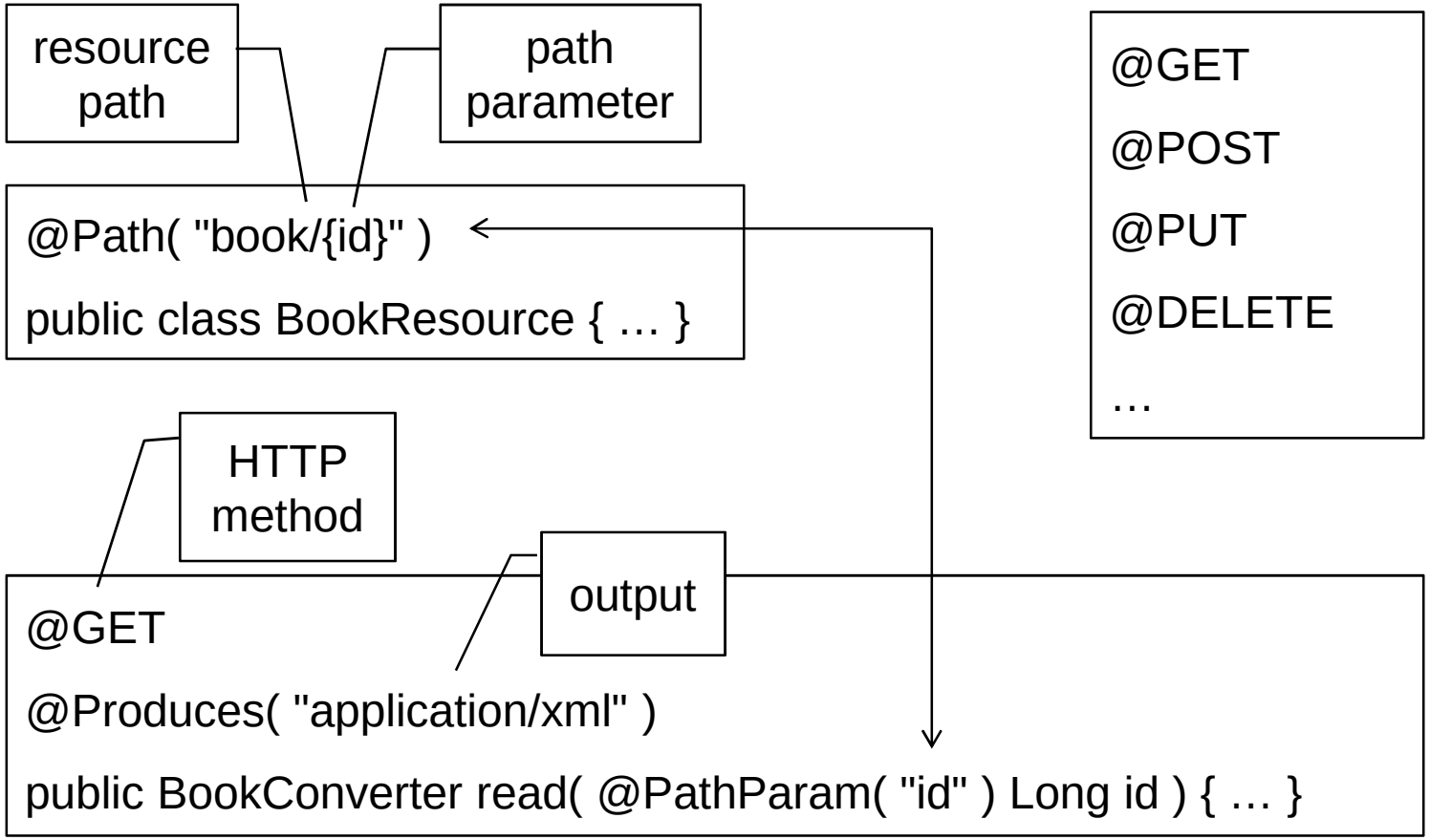
Category

```
<category uri="…">
<name>travel</name>
<book uri="…"/>
<book uri="…"/>
</category>
```

Book

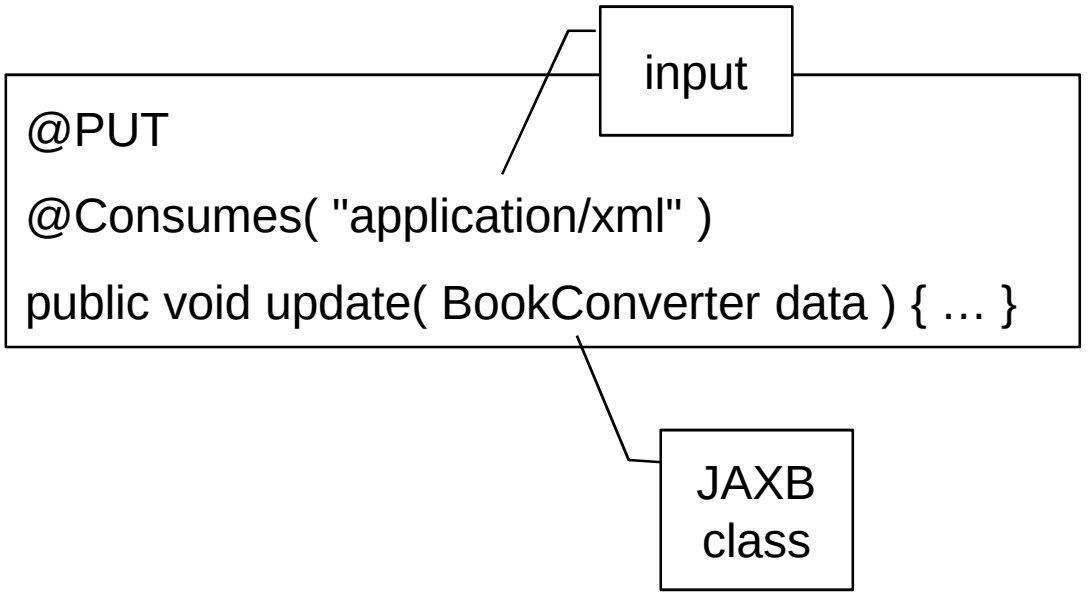
```
<book uri="…">
<id>1</id>
<title>Cesta na sever</title>
<author>Karel Čapek</author>
<pages>292</pages>
</book>
```

Java API for RESTful Services (JAX-RS)



Parameters

```
@CookieParam
@FormParam
@HeaderParam
@MatrixParam
@PathParam
@QueryParam
```



Matrix parameters

```
resources/map;lat=50;long=20;scale=32000
```

Otázky & odpovědi

tronicek@fit.cvut.cz