Webové služby v Java EE (JAX-WS)

Marek Rychlý

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů

Přednáška pro PDI
29. listopadu 2007

Obsah

Obsah

Architektura orientovaná na služby

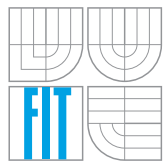
SOA: architektura orientovaná na služby (Service-Oriented Architecture), obecný koncept dvou komunikujících komponent,

WS: webové služby (Web Services), jedna z implementací SOA, spravuje W3C skupina Web Services Architectures.

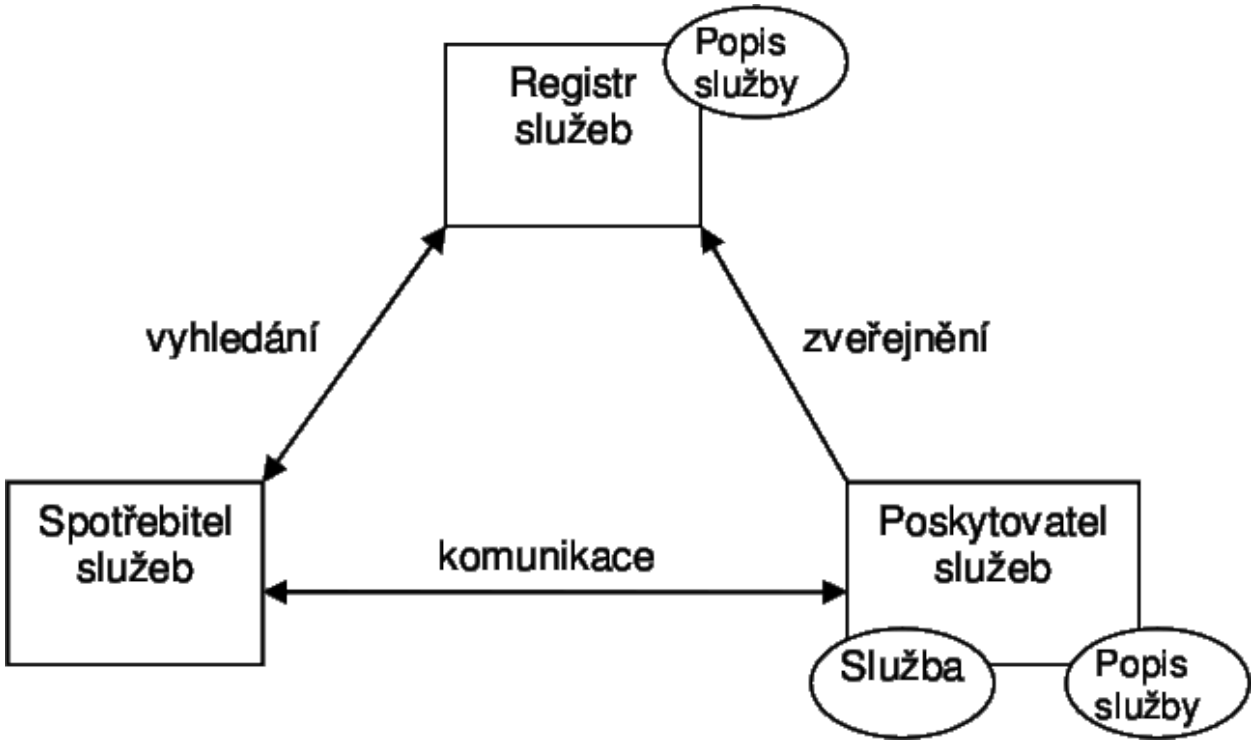
Role komunikujících stran:

poskytovatel služeb implementuje a nabízí služby (service provider), služba je specifikovaná svým popisem,

spotřebitel služeb na základě popisu vyhledá službu v registru služeb a použije ji (service consumer).



Konceptuální model SOA



model interakce mezi poskytovatelem služeb a spotřebitelem služeb

Struktura SOA

SOA je částečně vrstevnatá architektura (vrstva = úroveň abstrakce)

1. **vrstva byznys procesů** – BP je posloupnost kroků respektující byznys pravidla a vedoucí k zisku (hmotnému i nehmotnému), reprezentován sekvencí provedení několika služeb (choreografie služeb),
2. **vrstva služeb** – rozhraní jednotlivých komponent sjednocena do služeb, služba za běhu sestavuje komponenty a přeposílá jim požadavky, služba na rozhraní zpřístupňuje své funkce (popis služby),
3. **vrstva komponent** – základní stavební kameny služeb, realizace funkčnosti služeb a za zajištění požadované kvality služeb (QoS), komponenty jsou černé skříňky a jejich funkce jsou přístupné pouze přes rozhraní.

Spolupráce mezi službami v SOA

- služby **poskytují své prostředky** buď přímo cílovému spotřebiteli anebo jiným službám,
- služby mezi sebou **spolupracují** (komunikují) zasíláním zpráv.

Spolupráce mezi službami:

kooperace: jedna služba využívá prostředky jiné služby pro realizaci nabízených funkcí,

agregace: nová služba sestavená ze dvou (nebo více) služeb nabízí kombinaci funkcí dílčích služeb,

choreografie: služby potom spolupracují za účelem provedení byznys procesu.

Základní vlastnosti SOA

volné spojení – vztahy mezi službami jsou navazovány účelově,

nezávislost – služby jsou autonomní sebe-řídící jednotky,

abstrakce – služby zapouzdřují svoji logiku a okolnímu světu jsou přístupné pouze přes rozhraní,

znovupoužitel. – žádoucí vedlejší efekt návrhu a implementace, danou službu využívá co nejvíce aplikací,

bezstavovost – služby se pro vnějšího pozorovatele během své činnosti nenacházejí v žádném stavu,

multiplatformní – služby jsou nezávislé na implementačním jazyce i na operačním systému.

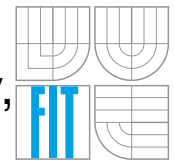
Porovnání SOA s jinými architekturami a principy

SOA **vs.** klient-server architektura

- klient–server = rozhraní a apl. logika $\times$ apl. logika, stav a sdílené zdroje,
- spotřebitel–poskytovatel = potřebuje $\times$ nabízí funkčnost,
- SOA je jemnější dekompozice (např. menší nároky na zdroje),
- SOA je více distribuovaná (rozmístění výpočetní logiky),
- služby se snaží být bezstavové z vnějšího pohledu.

SOA **vs.** objektově orientovaný přístup

- SOA přístup preferuje volném provázání entit (služeb) $\times$ OO přístup přesně vztahy mezi třídami, těsnější vazby entit (objektů),
- základní vlastností OO přístupu je dědičnost $\times$ SOA přístup s dědičností nepočítá, preferuje delegaci,
- základní vlastností SOA přístupu je bezstavovost entit $\times$ zapouzdření dat do objektů v OO přístupu,
- aktivita služeb v SOA přístupu je vyvolána až příchodem nějaké zprávy,
- podobný pohled na abstrakci entit (rozhraní).



Webové služby (Web Services)

Webové služby jsou neznámější a nejpoužívanější reálnou implementací SOA. Postaveny na následujících technologiích:

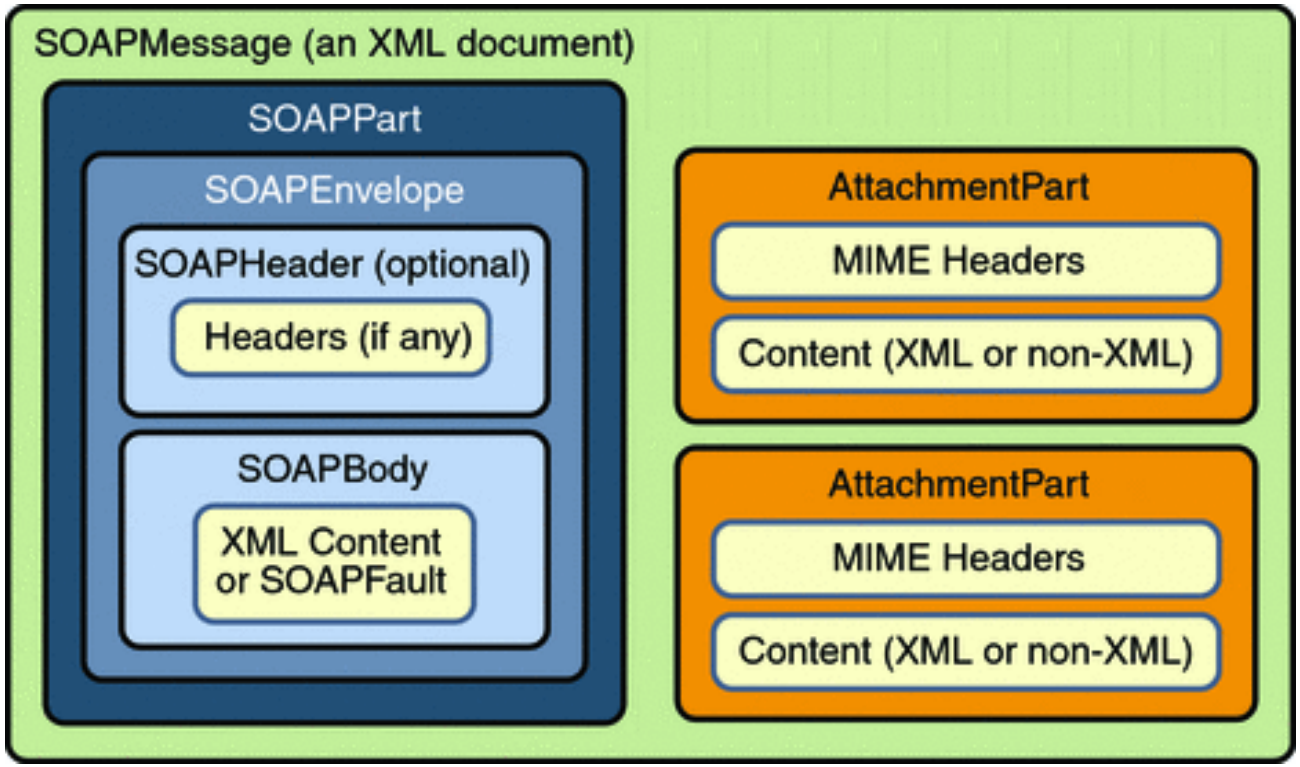
- Simple Object Access Protocol (SOAP) a HTTP protokol,
  (komunikační spojení, obálka, adresace, volání konkrétních služeb)
- eXtensible Markup Language (XML),
  (strukturování informací během přenosu a pro popisu)
- Universal Description, Discovery and Integration (UDDI),
  (mechanismus registrů pro vyhledávání webových služeb)
- Web Services Description Language (WSDL).
  (popis funkcí a umístění služeb a způsobu komunikace)

Simple Object Access Protocol (SOAP)

Protokol SOAP:

- základní vrstva WS technologie, výměna XML zpráv,
- patří do aplikační vrstvy pětivrstvého TCP/IP modelu,
- bezstavový protokol, nezávislé na protokolu a implementaci,
  (jedním z protokolů komunikace je HTTP/HTTPS protokol)
- podporuje několik typů volání funkcí služeb,
  (kde klient posílá XML zprávu na server, nejznámější je implementované Remote Procedure Call (RPC), SOAP vychází ze staršího XML-RPC)
- definuje strukturu zprávy (obálka kolem hlavičky a těla).
  (pravděpodobně vychází ze staršího Web Distributed Data eXchange (WDDX))

Struktura SOAP zprávy podle [SUN, 2007]



zpráva =
obálka s hlavičkami (nepovinné) a tělem + přílohy (nepovinné)

Ukázka struktury SOAP zprávy v HTTP protokolu I

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary; type=text/xml;
    start="<part1.xml@example.net>"
Content-Description: This is the optional message description.

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: <part1.xml@example.net>

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns0:setMemory xmlns:ns0="http://cz.vutbr.fit.rychly.wsDemo.service/">
      <newValue>999.999</newValue>
    </ns0:setMemory>
    ...
```

Ukázka struktury SOAP zprávy v HTTP protokolu II

```
    ...
    <myAttachment href="cid:part2.jpeg@example.net/>
    ...
  </soap:Body>
</soap:Envelope>

--MIME_boundary
Content-Type: image/jpeg
Content-Transfer-Encoding: binary
Content-ID: <part2.jpeg@example.net>

...binary JEPG image...
--MIME_boundary--
```

Přílohu lze odkazovat:

- pomocí HTTP atributu hlavičky `Content-ID`,
  (viz příklad nahoře)
- pomocí HTTP atributu hlavičky `Content-Location`.
  (MIME entita může mít uvedeny oba dva atributy)

Druhy volání služeb v SOAP protokolu

- **notification**
  – volání bez parametrů, jen vrací hodnotu,
- **one way**
  – volání s parametry a nevrací žádnou hodnotu,
- **request-response**
  – volání s parametry a návratovou hodnotou.

Web Services Description Language (WSDL)

W3C zavedla WSDL jako standard pro XML popis webových služeb.

- Jaké funkce poskytuje daná služba?
- Kde je daná služba uložena?
- Jak může být s danou službou navázána komunikace?

Každá služba jako množina koncových bodů (service endpoints).

- v těchto bodech komunikuje s okolím pomocí zasílání zpráv,
  (pro jednoduchost si lze koncový bod představit jako rozhraní služby)
- WSDL poskytuje formální definici koncových bodů:
  1. abstraktní popis koncového bodu,
  2. konkrétního popis koncového bodu.

Abstraktní a konkrétní popis ve WSDL

Abstraktní popis koncového bodu:

- popis rozhraní služby bez ohledu na konkrétní technologie a protokoly,
- tři základní oddíly:

interface:	rozhraní služby, tj. poskytované operace,
operation:	popis operací, jejich vstupní a výstupní parametry,
message:	popis zpráv, které představují operace a jejich parametry.

Konkrétní popis koncového bodu:

- navázání abstraktního popisu na reálnou implementaci a komunikace na konkrétní protokol,
- tří základních oddíly:

binding:	požadavky pro navázání konkrétního spojení (např. v SOAP), pro jednotlivé operation nebo celé interface,
service:	popisuje službu, seskupuje prvky enpoint,
endpoint:	určuje fyzickou adresu, na které je služba přístupná.

Ukázka WSDL popisu služby I

Definujeme jmenné prostory pro jednotlivé části dokumentu:

- soap: WSDL popisuje službu pro protokol SOAP,
- tns: názvy používané v naší aplikaci,
- xsd: popis protokolu volání služby jako XML dokumentu,

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="mathService"
 targetNamespace="http://service.wsDemo.rychly.fit.vutbr.cz/"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:tns="http://service.wsDemo.rychly.fit.vutbr.cz/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns="http://schemas.xmlsoap.org/wsdl/">

<types>
  <xsd:schema>
    <xsd:import namespace="http://service.wsDemo.rychly.fit.vutbr.cz/"
     schemaLocation="http://localhost:8080/ws-demo/math?xsd=1" />
  </xsd:schema>
</types>
```

Ukázka WSDL popisu služby II

Definujeme zprávy přijímané a navracené danou službou:

- message: definice vlastní zprávy přenášené při komunikaci,
- operation: přiřazení zpráv k operacím posytovaným službou.

```
<message name="addToMemory">
  <part name="parameters" element="tns:addToMemory"></part>
</message>
<message name="addToMemoryResponse">
  <part name="parameters" element="tns:addToMemoryResponse"></part>
</message>

<portType name="math">
  <operation name="addToMemory">
    <input message="tns:addToMemory" />
    <output message="tns:addToMemoryResponse" />
  </operation>
</portType>
```

Ukázka WSDL popisu služby III

Popis formátu volání operací pro SOAP a umístění služby:

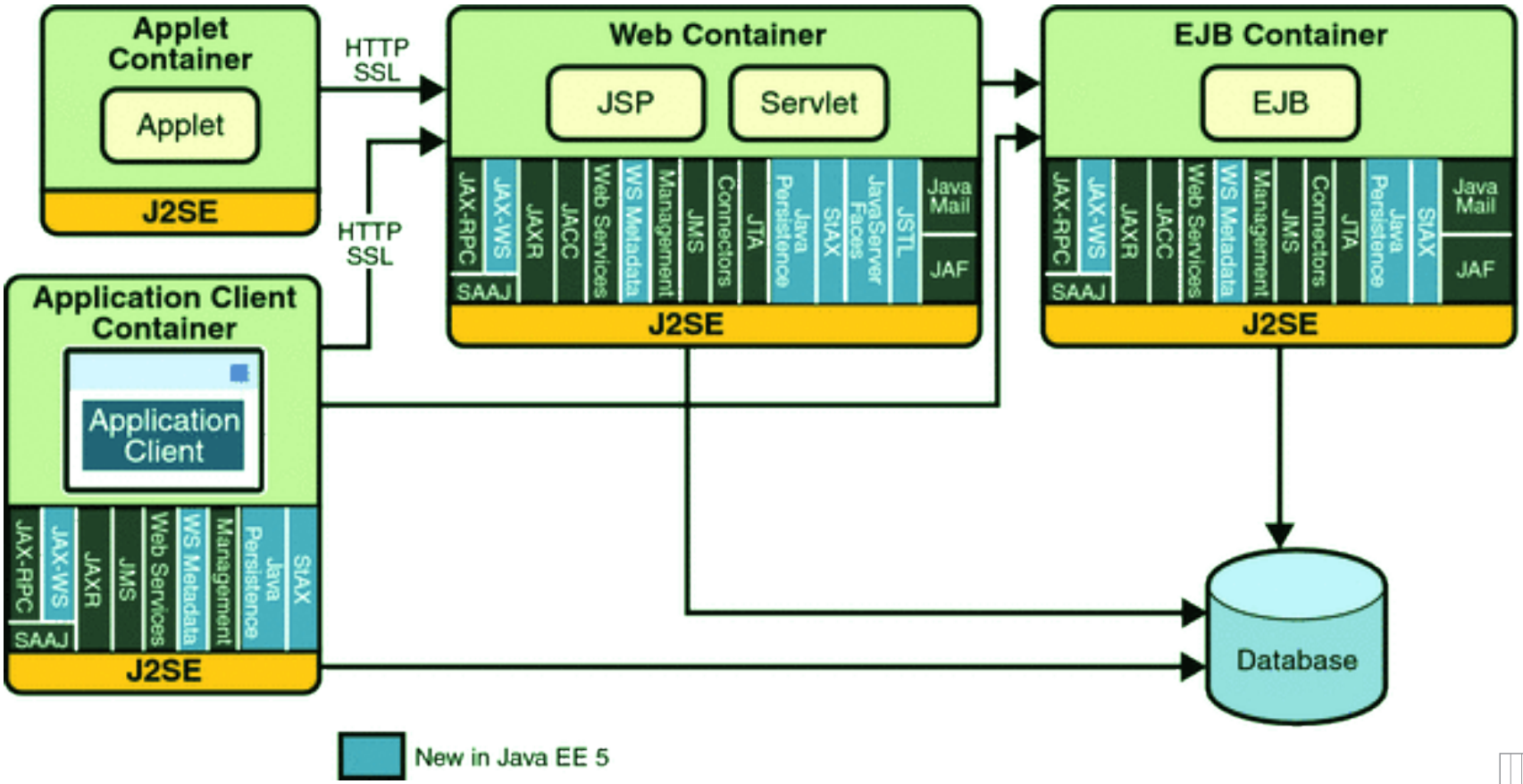
```
<binding name="mathPortBinding" type="tns:math">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
   style="document" />
  <operation name="addToMemory">
    <soap:operation soapAction="" />
    <input>   <soap:body use="literal" />  </input>
    <output>  <soap:body use="literal" />  </output>
  </operation>
</binding>

<service name="mathService">
  <port name="mathPort" binding="tns:mathPortBinding">
    <soap:address location="http://localhost:8080/ws-demo/math" />
  </port>
</service>

</definitions>
```

Obsah

Java API for XML Web Services (JAX-WS)

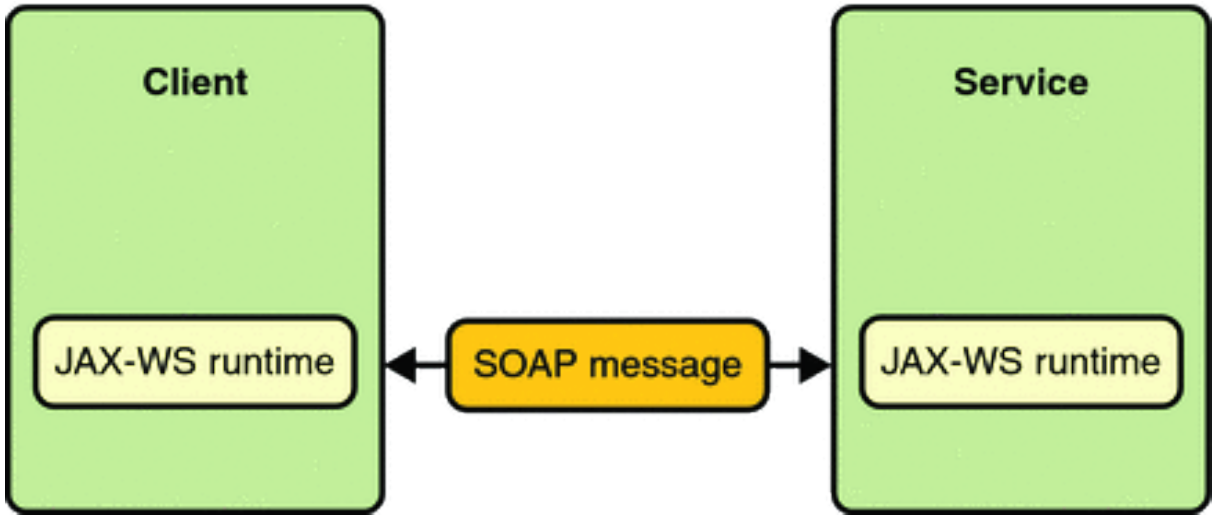
- pro realizaci XML webových služeb a jejich klientů v Java EE,
- používá přenos XML zpráv SOAP protokolem přes HTTP,
- podporuje „message-oriented“ a „RPC-oriented“ webové služby,
- webové služby jsou definovány jako Java třídy s poskytovanými metodami označenými pomocí anotací,
  (automatický převod definice rozhraní třídy do popisu pomocí WSDL)
- spotřebitel vytvoří pomocí JAX-WS lokální proxy pro vzdálenou webovou službu a tu transparentně používá,
  (proxy má stejné rozhraní jako třída implementující webovou službu u poskytovatele, převod volání metod proxy na SOAP zprávy je automatický),
- respektuje nezávislost na platformě – lze komunikovat s libovolně implementovanými službami.
  (tzn. se službami neimplementovanými v Javě, pomocí JAX-WS, apod.)

Technologie JAX-WS v Java EE podle [SUN, 2007]



tenký/tlustý klient – Java EE aplikační server – databáze

Komunikace přes JAX-WS podle [SUN, 2007]



JAX-WS tvoří abstrakci nad SOAP komunikací webových služeb

Implementace poskytovatele WS pomocí JAX-WS

Použití anotací `@javax.jws.WebService` a `@javax.jws.WebMethod`:

- webová služba je realizována jako rozhraní/třída s anotací `@WebService`,
  (service endpoint interface/service endpoint implementation (SEI))
- implicitní je definice rozhraní služby společně s implementací (ve třídě),
  (přímo u implementace metod třídy pomocí anotace `@WebMethod`)
- parametr `endpointInterface` umožňuje oddělit definici rozhraní služby,
  (rozhraní musí rozšiřovat rozhraní `Remote` a označit metody pomocí `@WebMethod`)
- třída implementující službu nesmí být `final` nebo `abstract`, musí mít implicitní bezparametrický konstruktor,
  (metody s anotací `@PostConstruct` a `@PreDestroy` obslouží vznik a zánik)
- metody rozhraní/třídy pro webovou službu (anotace `@WebMethod`) musí být `public` a nesmí být `static` nebo `final`,
- typy parametrů a návratových hodnot metod rozhraní/třídy pro webovou službu jsou omezeny Java Architecture for XML Data Binding (JAXB),
- z třídy (tj. z implementace služby) lze generovat popis služby.
  (voláním `wsgen -d <output dir> -classpath <cp dir> <SEI class>`)

Ukázka implementace poskytovatele v JAX-WS

```
import javax.jws.*
@WebService()
public class math {
  private double memory = 0;  private String ver = "Math Service v0.1";

  @WebMethod(operationName = "getMemory")
  public double getMemory() { return this.memory; }

  @WebMethod(operationName = "setMemory")
  @Oneway
  public void setMemory(@WebParam(name = "newValue") double newValue) {
    this.memory = newValue; }

  @WebMethod(operationName = "addToMemory")
  public double addToMemory(@WebParam(name = "addValue") double addValue) {
    return this.memory = this.memory + addValue; }

  @WebMethod(operationName = "getVersion")
  public String getVersion() { return this.ver; }
}
```

Zprovoznění poskytovatele WS pomocí JAX-WS

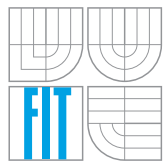
Postup zprovoznění poskytovatele webové služby:

1. tvorba kódu třídy s implementací služby,
2. kompilace třídy s implementací služby,
3. použití nástroje `wsgen` pro generování popisu služby,
4. zabalení zkompilované implementace a popisu do WAR archivu,
5. umístění WAR archivu na aplikační server.
   (apl. server automaticky generuje popis služby vyžadovaný spotřebitelem)

Většina aplikačních serverů umožňuje poskytuje pomocné akce:

Tomcat informace o webové službě na adrese
`http://localhost:8080/ws-demo/math?Tester`
a WSDL popis na adrese
`http://localhost:8080/ws-demo/math?wsdl`

SUN testování webové služby na adrese
`http://localhost:8080/ws-demo/math?Tester`



Implementace spotřebitele WS pomocí JAX-WS

Použití anotace `@WebServiceRef`:

- klient deklaruje odkaz na webovou službu anotací `@WebServiceRef`,
  (parametr `wsdlLocation` udává URI na WSDL odkazované služby)
- anotace se použije s deklarací proměnné zastupující objekt poskytovatele proxy,
  (proměnná se deklaruje jako `static` a bez přiřazení hodnoty, typ proměnné, tj. třída objektu, je třída implementující službu doplněná v názvu slovem „Service“)
- objekt poskytovatele proxy se použije k získání proxy (portu služby),
  (metoda `getNamePort`, kde slovo „Name“ je název třídy implementující službu)
- proxy se používá jako klasický objekt třídy implementující službu.
  (práce s webovou službou zastupovanou proxy je plně transparentní, jako práce s lokálním objektem třídy implementující webovou službu)
- z WSDL lze generovat zdroje pro referenci služby v kódu klienta.
  (voláním `wsimport -d <output dir> <WSDL file>`)

Ukázka implementace spotřebitele v JAX-WS

```
import javax.xml.ws.WebServiceRef;
import cz.vutbr.fit.rychly.wsDemo.service.MathService;
import cz.vutbr.fit.rychly.wsDemo.service.math;

public class MathClient {
  @WebServiceRef(wsdlLocation="http://localhost:8080/ws-demo/math?wsdl")
  static MathService service;

  public static void main(String[] args) {
    try { (new MathClient).doTest(args); }
    catch(Exception e) { e.printStackTrace(); }
  }

  public void doTest(String[] args) {
    try {
      System.out.println("retrieving the port from" + service);
      Math port = service.getMathPort();
      System.out.println("invoking the operation on the port");
      Sestem.out.println(port.getMemory());
    } catch(Exception e) { e.printStackTrace(); }
} }
```

Zprovoznění spotřebitele WS pomocí JAX-WS

Postup zprovoznění spotřebitele webové služby:

1. tvorba kódu tříd spotřebovávajících webovou službu,
2. použití nástroje `wsimport` s WSDL pro generování zdrojů,
3. kompilace tříd s spotřebovávajících webovou službu,
4. spuštění spotřebitele webové služby.

Celý proces může být usnadněn vývojovým prostředím (IDE).

Kombinace JAX-WS s doplňujícími technologiemi

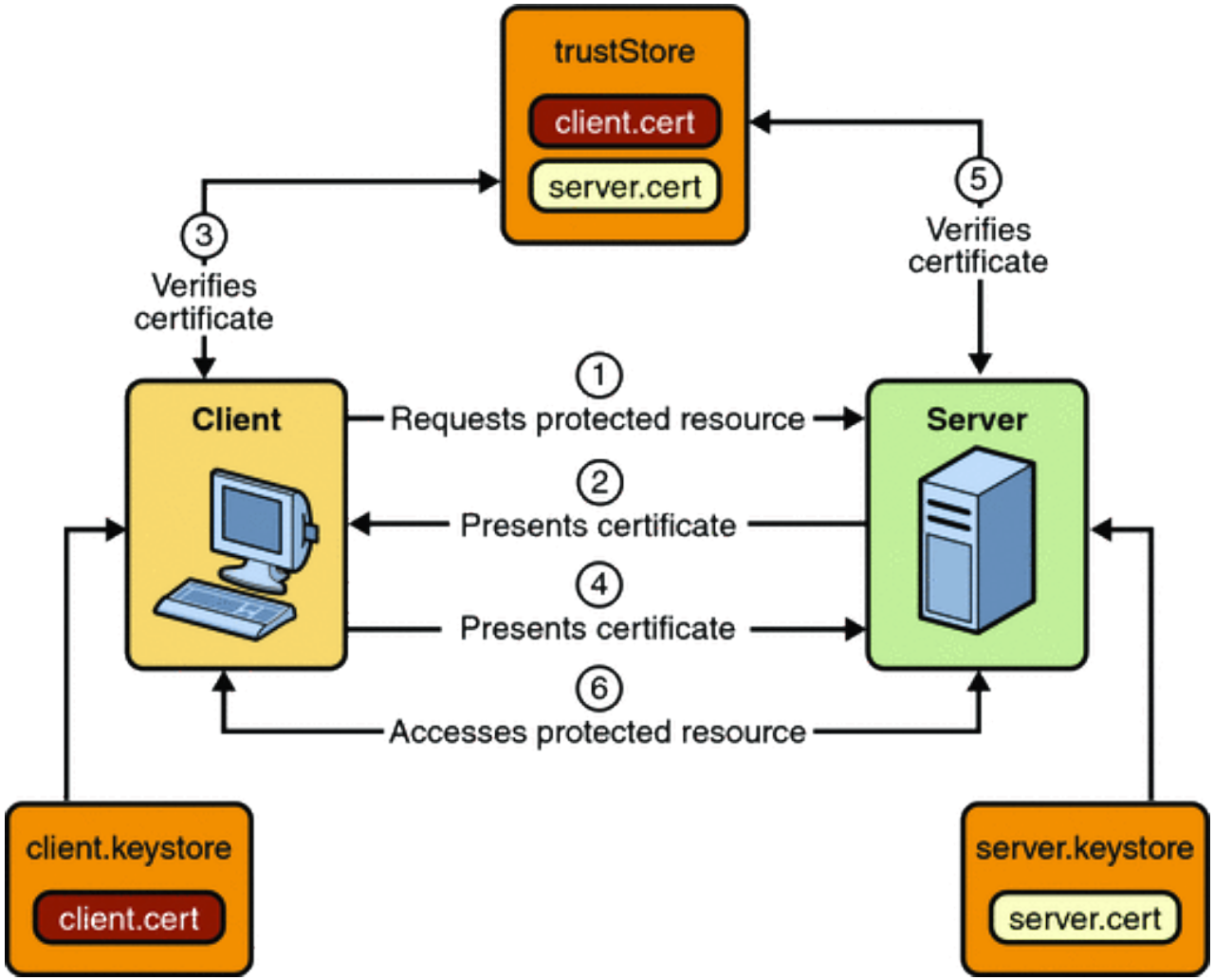
Speciální použití webových služeb:

- Streaming API for XML (StAX) – postupné zpřístupnění XML dokumentu pomocí webové služby (toto není Document Object Model (DOM)),
- SOAP with Attachments API for Java (SAAJ) – volání s přílohami.

Autentizace a autorizace:

- anotace `javax.annotation.security.RolesAllowed` umožňuje specifikovat role, které mohou přistupovat ke službě (SEI třídě) nebo jejím metodám,
- anotace `javax.annotation.security.DeclareRoles` umožňuje nastavit role, ve kterých bude služba (SEI třída) vystupovat,
  (užitečné při (oboustranném) použití certifikátů)
- apl. server autentizuje uživatele a autorizuje je pro přístup k webovým komponentám s definovanými rolemi,
  (autentizační mechanismy Basic, Digest, Form, Client-Cert, a jiné)
- webová služba je webová komponenta, přístup k ní lze zabezpečit podobně jako např. k servletu.

Zabezpečení webové komponenty podle [SUN, 2007]



Použití JAX-WS v aplikačním serveru Tomcat

Nebudeme využívat zabudovaný apl. server v IDE, použijeme[1]

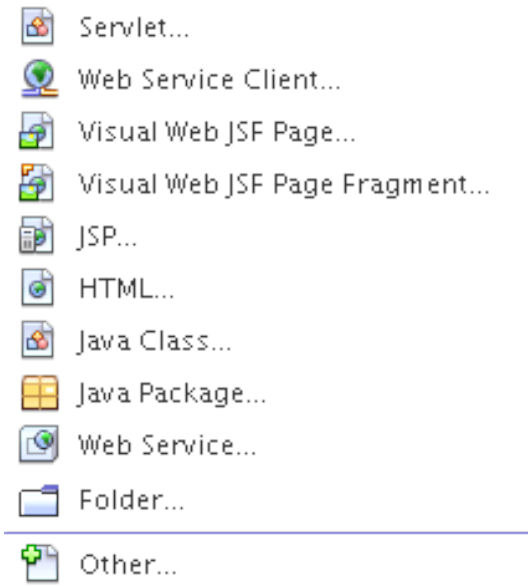
1. Apache Tomcat 6.0 je Servlet-2.5/JSP-2.1 webový kontejner, (nepodporuje JavaBeans, jako např. Sun Java System Application Server),
2. stáhneme z `http://tomcat.apache.org/` a nainstalujeme,
3. přidáme účet správce serveru do `tomcat-users.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="manager"/>  <role rolename="admin"/>
  <role rolename="host-manager"/>
  <user username="x" password="y" roles="admin,manager,host-manager"/>
</tomcat-users>
```

4. spustíme a spravujeme webovým rozhraním na `http://x:y@localhost:8080/manager/html`

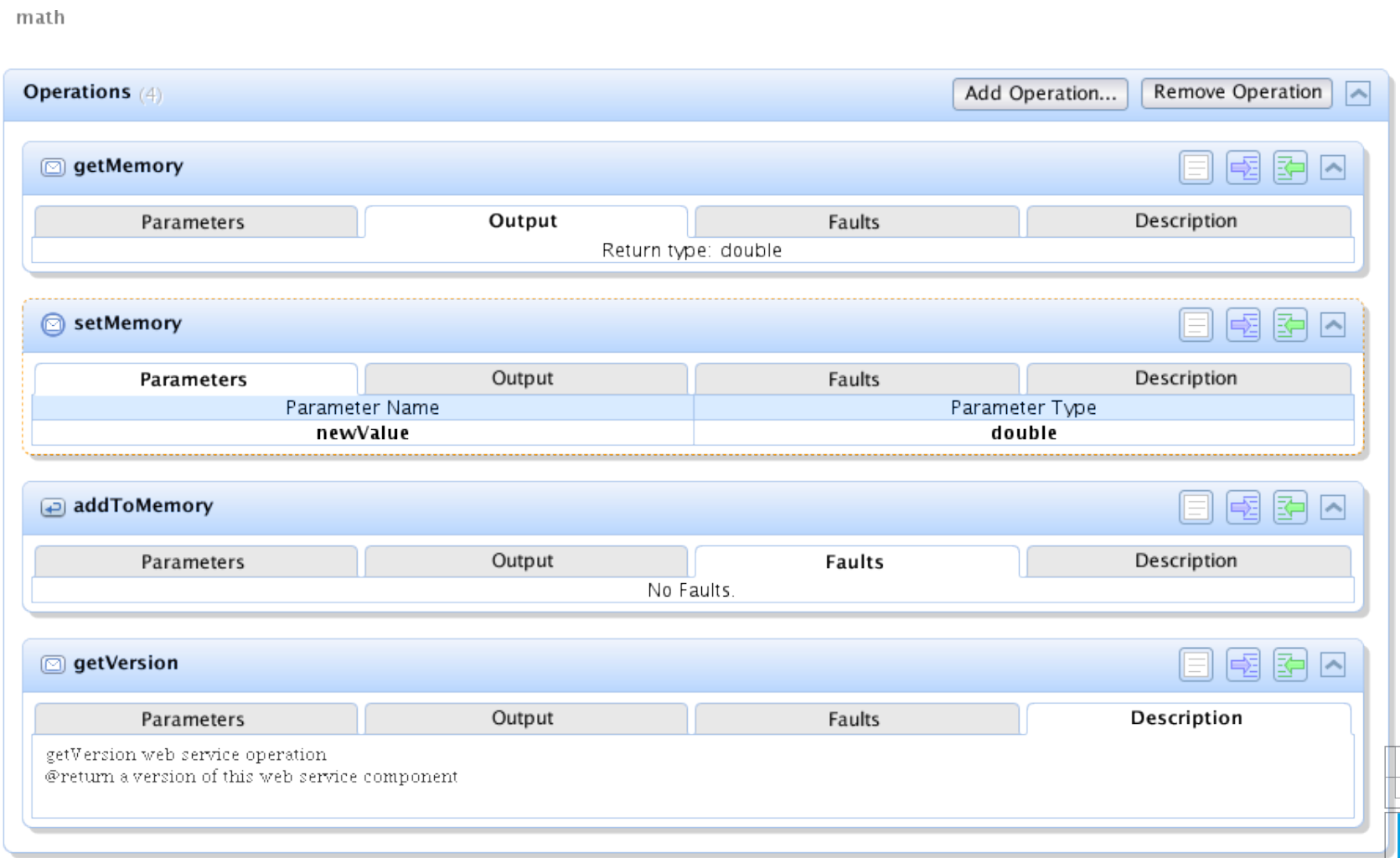
[1] nebo použijeme školní server `pcuifs2`

Podpora JAX-WS ve vývojovém prostředí NetBeans



1. v menu „Tools: Servers“ přidáme server Tomcat příslušné verze,
2. založíme nový projekt typu „Web Application“,
3. v „Project Properties: Run“ nastavíme server a cestu ke komponentě,
4. z kontextové nabídky projektu vybereme typ nové komponenty (viz vlevo),
5. po implementaci dáme „Build“ a „Deploy“.

Konfigurace služby ve vývojovém prostředí NetBeans



Obsah

Shrnutí a závěr

- SOA umožňuje dekomponovat a rozmístit informační systém,
- webové služby (WS) jsou standardizovanou implementací SOA,
- Java EE 5 výrazně usnadňuje realizaci WS pomocí JAX-WS,
- WS jsou webové komponenty se vším, co k tomu patří.

Pokračování?

- práce s Java EE 5, JAX-WS, Tomcat-em a NetBeans na cvičení,
- přednáška o návrhu informačního systému pomocí SOA,
  (v AIS, 3. 12. 2007, 10.00–11.50, místnost D0206, přednáší ing. Petr Weiss)
- projekt z PDI zaměřený na Java EE, JAX-WS, persistenci, atd.

Literatura

Monson-Haefel, R. (2003).

J2EE Web Services: XML SOAP WSDL UDDI WS-I JAX-RPC JAXR SAAJ JAXP.

Addison-Wesley Professional.

7th Printing, March 2007.

SUN (2007).

The Java™ EE 5 Tutorial.

Sun Microsystems, 4150 Network Circle, Santa Clara, CA 95054, U.S.A.