Ukázka možnosti využití programu OpenJUMP v SOA

Martin Prager
Institute of Geoinformatics
Faculty of Mining and Geology, VSB-TUO
E-mail: martin.prager.hgf@vsb.cz

Klíčová slova: Orechestration, chainging, web services, GeoWeb, JUMP, SOA

Abstrakt

Tento krátký článek by měl poukázat prostřednictvím vytvořené extenze na možnost širokého uplatnění open-source produktu OpenJUMP (dříve JUMP) a jeho modularitu. A to konkrétně v oblasti servisně orientované architektury (SOA) se zaměřením na řetězení webových služeb.

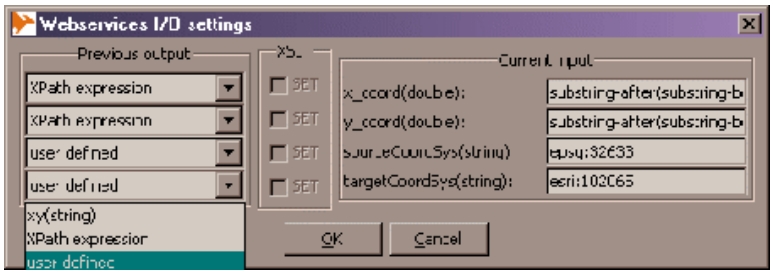
Úvod

Servisně orientovaná architektura (SOA) přitahuje zájem všech oblastí IT průmyslu. Poháněná standardy jako XML, webové služby a SOAP, SOA rychle proniká do hlavních chodů aplikací zásadních pro plnění business operací. S rostoucím počtem dostupných služeb a nároky na efektivnost a obtížnost řešených úloh si už mnohdy nevystačíme pouze s výsledky (zdroji) jednotlivých služeb, případně s jejich statickým propojením. Jsme nuceni začít služby řetězit dynamicky. Spojovat je dle aktuálních potřeb, možností uživatele (finance, přesnost výsledků, rychlost ap.). Existují dvě úrovně řetězení služeb, známé pod názvy orchestrace a choreografie (zaměřena více globálně). Orchestrace a choreografie je spojena s některými standardy (BPEL, WS-CDL, XLANG atd.) a organizacemi (OASIS, W3C, BPMI apod.) [4].

Extenze pro OpenJUMP

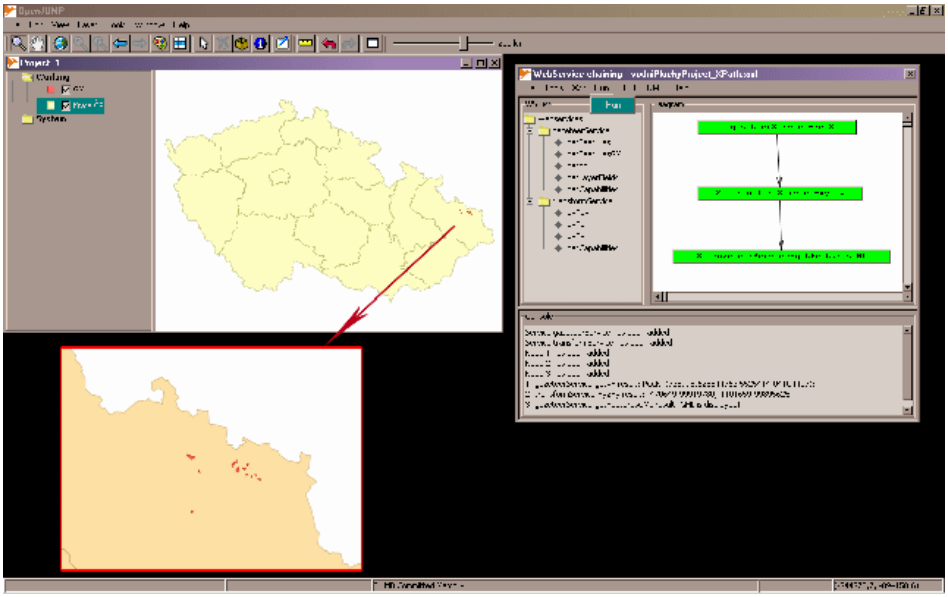
OpenJUMP API poskytuje programátorům přístup ke všem funkcím včetně I/O rozhraní, datovým sadám, vizualizaci a prostorovým operacím. Tato možnost z něj dělá vysoce modulární a rozšiřitelný produkt. Rozšíření jsou zde realizována prostřednictvím zásuvných modulů. Tyto moduly, které jsou po startu „Workbench“ nahrány, můžou mít podobu „plugin“ (položky menu), „cursor tools“ (nástrojové tlačítka), „renderers“ (způsoby vykreslování dat) a „datasources“ (způsoby nahrávání a ukládání různých datových formátů) [3]. Extenze, kterou jsem nazval „WSPlugin“ poskytuje uživateli GUI pro vytváření jednoduchých

sekvenčních řetězů webových služeb. Umožňuje přidávat služby typu WSDL, WMS (WMS pouze na konec řetězu) jednotlivě, nebo hromadně s podporou vyhledávání v katalogu WSCO (http://gisak.vsb.cz/wsco/intranet/). Samotný řetěz poté vytvoří uživatel přesunutím jednotlivých metod (operací „Drag-and-Drop“) na pracovní plochu. Obrázek 1 znázorňuje grafické rozhraní pro propojování parametrů jednotlivých metod (aktivováno kliknutím na konkrétní metodu). Parametry lze propojovat přímo, nebo je transformovat s využitím jazyka XSL, či za pomocí XPath výrazů. Využití XPath (viz http://www.w3.org) výrazů lze je patrné jak na následujícím obrázku, tak v přiloženém ukázkovém BPEL procesu.



Obrázek 1: Grafické rozhraní pro nastavení relací mezi metodami [4]

Na obrázku 2 lze vpravo vidět spuštěnou extenzi s již vykonaným řetězem služeb, který vrací jako odpověď GML vrstvu. Tato vrstva je automaticky vizualizována v mapovém okně OpenJUMP. V případě, že na konci řetězu stojí WMS služba, je zobrazena s následnou možností modifikace jejích parametrů.



Obrázek 2: Spuštěná extenze v prostředí OpenJUMP [4]

Jak již bylo zmíněno v úvodu, řetězení služeb je spojeno se spoustou specifikací. Jazyk BPEL se stal v posledních dvou letech významným standardem, vyzdvihujícím využití SOA z IT úrovně na obchodní úroveň. Umožňuje organizacím automatizování jejich obchodních procesů, prostřednictvím orchestrace služeb uvnitř i vně dané organizace [1]. Vyvinut firmami Microsoft a IBM, standardizován neprofitujícím konsorciem OASIS (http://docs.oasis-open.org).

Z toho důvodu byla do extenze zabudovaná podpora BPEL jazyka. V případě, že by chtěl uživatel daný řetěz služeb poskytovat širší veřejnosti, má možnost ho exportovat (importovat pouze soubory exportované z extenze) jako proces do tohoto jazyka. Slouží zde k tomu jednoduchý průvodce, který umožňuje nastavit některé základní parametry (název procesu, metodu procesu, výběr relevantních vstupních parametrů ap.). Kromě těchto základních parametrů lze nastavit export přímo do formátu „ActiveBPEL engine“ stroje (http://www.active-endpoints.com/open-source-active-bpel-Intro.htm). BPEL proces se jeví navenek jako standardní webová služba, kterou bychom mohli opětovně načíst do extenze a propojit s dalšími službami, případně procesy. Dále v extenzi lze nastavit některé vlastnosti prostředí, pracovat s projektem ap. Extenze v současnosti umožňuje grafický návrh sekvenčního řetězení webových služeb s podporou přímého zobrazování mapových výstupů. Na rozdíl od všech ostatních produktů určených k řetězení webových služeb, umožňuje do řetězu zařazovat kromě WSDL služeb i služby WMS. BPEL jazyk je sice standard, ovšem s přenositelností procesu mezi různými BPEL stroji je problém. Většinou každý z těchto strojů potřebuje k interpretaci kromě standardních souborů BPEL jazyka ještě další podpůrné soubory, které jsou bohužel typické pro jednotlivé stroje. Jak je vidět prostředí OpenJUMP lze asi rozšířit o cokoliv. K tomuto faktu značně přispívá velmi dobrá dokumentace programu a jeho široká komunita.

Základní struktura BPEL procesu

```
<process...>
<partnerLinks.../>
<variables.../>
<correlationSets.../>
<faultHandlers.../>
<compensationHandler.../>
<eventHandlers.../>
activities
</process>
```

Význam jednotlivých elementů je následující [2]:

- partnerLinks – zde jsou definovány služby, které proces využívá a poskytuje. Uvnitř každého partnerLink musí být vymezena aspoň jedna z dvou možných rolí. Roli samotného procesu určuje atribut myRole a naopak roli partnera atribut partnerRole;
- variables – tato část specifikuje proměnné, které proces využívá. BPEL umožňuje deklarovat proměnné třemi způsoby: jako typ WSDL zprávy, jako typ XML Schema (jednoduchý nebo složený) a jako XML Schema element;
- correlationSets – ještě před spuštěním daného procesu dochází k vytvoření jeho instance. Význam tohoto elementu spočívá v tom, že zabezpečuje doručování přicházejících zpráv odpovídajícím instancím procesů;
- faultHandlers – jak už název sám o sobě říká, tento element slouží ke zpracování chyb, které nastanou při běhu procesu. Chyby mohou být vyvolány explicitně (pomocí elementu $< throw >$), nebo implicitně (jako např. výsledek chyby při volání nějaké partnerské služby). Naopak pro zachytávání chyb je využíván element s odpovídajícím jménem $< catch >$;
- compensationHandler – tento element poskytuje možnost zpětného zotavení z chyby v specifikované oblasti. Jinými slovy pokusit se o jakési vyčistění a navrácení se do stavu,

kde může proces po chybě pokračovat;

- eventHandlers – slouží k zachycení událostí. V BPEL jsou dva typy: příchozí zprávy (korespondují s WSDL operacemi) a alarmy (aktivovány po uživatelem zadaném čase). V každém tomto elementu musí být obsažen aspoň jeden zmíněny typ;
- activities – struktura procesu pokračuje tzv. aktivitami (elementy), které už implementují jeho samotný tok. Každý proces má jednu hlavní aktivitu. BPEL obsahuje sadu jednoduchých aktivit, které můžeme skládat a vytvářet tak aktivity složené (viz [2]).

Ukázkový BPEL proces

```
<?xml version="1.0" encoding="UTF-8"?>
<process xmlns:ns1="http://tonda.vsb.cz/gazeteer" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns2="http://postgis.vsb.cz/transform" xmlns:ns0="cz.vsb.jump.wsplugin"
xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
targetNamespace="cz.vsb.jump.wsplugin.VodniPlochy"
exitOnStandardFault="yes" suppressJoinFailure="yes" name="VodniPlochy" >
<partnerLinks >
<partnerLink myRole="VodniPlochyRole" partnerLinkType="ns0:VodniPlochyPLT" name="VodniPlochyLink" />
<partnerLink partnerRole="gazeteerRole" partnerLinkType="ns0:gazeteerPLT" name="gazeteerLink" />
<partnerLink partnerRole="transformRole" partnerLinkType="ns0:transformPLT" name="transformLink" />
</partnerLinks>
<variables >
<variable name="getAreasRequest" messageType="ns0:getAreasRequest" />
<variable name="getAreasResponse" messageType="ns0:getAreasResponse" />
<variable name="getXY_0Request" messageType="ns1:getXYRequest" />
<variable name="getXY_0Response" messageType="ns1:getXYResponse" />
<variable name="xy2xy_1Request" messageType="ns2:xy2xyRequest" />
<variable name="xy2xy_1Response" messageType="ns2:xy2xyResponse" />
<variable name="getFeaturesGML_2Request" messageType="ns1:getFeaturesGMLRequest" />
<variable name="getFeaturesGML_2Response" messageType="ns1:getFeaturesGMLResponse" />
</variables>
<faultHandlers >
<catchAll >
<sequence >
<compensate name="ReverseCompletedTransactions" />
<assign name="GenerateFaultMessage" >
<copy >
<from >
<literal >An error occurred while submitting the order. All transactions have been canceled.</literal>
</from>
<to part="countries" variable="getAreasResponse" />
</copy>
</assign>
<reply variable="getAreasResponse" portType="ns0:VodniPlochyPT" name="SendError"
operation="getAreas" partnerLink="VodniPlochyLink" />
</sequence>
</catchAll>
</faultHandlers>
<sequence >
<receive createInstance="yes" variable="getAreasRequest" portType="ns0:VodniPlochyPT"
operation="getAreas" partnerLink="VodniPlochyLink" />
<assign >
<copy > <from >'obce32633'</from><to part="layer" variable="getXY_0Request" /> </copy>
<copy > <from >'nazob_a'</from> <to part="field" variable="getXY_0Request" /> </copy>
<copy > <from >$getAreasRequest.Obec</from> <to part="name" variable="getXY_0Request" /></copy>
</assign>
<invoke outputVariable="getXY_0Response" inputVariable="getXY_0Request" portType="ns1:gazeteerPort"
name="InvokegetXY" operation="getXY" partnerLink="gazeteerLink" />
<assign >
<copy > <from >substring-after(substring-before($getXY_0Response.xy,' '),'(')</from>
```

```
<to part="x_coord" variable="xy2xy_1Request" /> </copy>
<copy > <from >substring-after(substring-before($getXY_0Response.xy,')'),' ')</from>
<to part="y_coord" variable="xy2xy_1Request" /> </copy>
<copy ><from >'epsg:32633'</from> <to part="sourceCoordSys" variable="xy2xy_1Request" /> </copy>
<copy > <from >'esri:102065'</from> <to part="targetCoordSys" variable="xy2xy_1Request" /> </copy>
</assign>
<invoke outputVariable="xy2xy_1Response" inputVariable="xy2xy_1Request" portType="ns2:transformPort"
name="Invokexy2xy" operation="xy2xy" partnerLink="transformLink" />
<assign >
<copy > <from >substring-before($xy2xy_1Response.xy,',')</from>
<to part="longitude" variable="getFeaturesGML_2Request" /> </copy>
<copy ><from >substring-after($xy2xy_1Response.xy,',')</from>
<to part="latitude" variable="getFeaturesGML_2Request" /> </copy>
<copy > <from >$getAreasRequest.Vzdalenost</from>
<to part="distance" variable="getFeaturesGML_2Request" /> </copy>
<copy ><from >'voda'</from> <to part="layer" variable="getFeaturesGML_2Request" /> </copy>
<copy ><from >'nazev'</from> <to part="field" variable="getFeaturesGML_2Request" /> </copy>
</assign>
<invoke outputVariable="getFeaturesGML_2Response" inputVariable="getFeaturesGML_2Request"
portType="ns1:gazeteerPort" name="InvokegetFeaturesGML" operation="getFeaturesGML"
partnerLink="gazeteerLink" />
<assign >
<copy ><from >$getFeaturesGML_2Response.countries</from> <to part="countries"
variable="getAreasResponse" />
</copy>
</assign>
<reply variable="getAreasResponse" portType="ns0:VodniPlochyPT"
operation="getAreas" partnerLink="VodniPlochyLink" />
</sequence>
</process>
```

Použité zdroje

1. BLANVALVET, S; BOLIE, J; CARDELLA, M; CAREY, S; CHANDRAN, P; COENE, Y; GEMINIUC, K; JURIČ, M; NGUYEN, H; PODUVAL, A; PRAVIN, L; THOMAS, J; TODD, D. BPEL Cookbook: Best Practivec for SOA-based integration and composite applications development. Birmingham: Packt Publishing Ltd., 2006. ISBN 1-904811-33-7
2. OASIS. [online]. 2007. Dostupný na WWW: http://www.oasis-open.org
3. OpenJUMP. [online]. 2007. Dostupný na WWW: http://openjump.org
4. PRAGER, M. Řetězení webových služeb v prostředí open source GIS. Diplomová práce. 2007. Ostrava. Dostupný na WWW: http://gisak.vsb.cz/~pra089/texty/DP_pra089_v1_0.pdf