ENTERPRISE SERVICE BUS PRO GEOPORTÁL

Jan Růžička

Institut geoinformatiky, Hornicko-geologický fakulta, VŠB-TUO,
107. listopadu 15, 708 33, Ostrava – Poruba, Česká republika
jan.ruzicka@vsb.cz

Abstrakt

Direktiva INSPIRE připravila pravidla pro integraci řady různých zdrojů geodat. Pokud jsou tyto zdroje dostupné v souladu s těmito pravidly, pak kdokoli může vytvořit nástroj zvaný geoportál. Geoportál umožňuje zpřístupnit celou řadu datových zdrojů přes jedno logické přístupové místo. Takovýto typ nástroje je typickým příkladem integrace geografických informačních systémů. Integrace může být řešena řadou specifických způsobů. Když se zabýváme integrací informačních systémů, vždy se snažíme aby byla tato integrace co nejvíce efektivní. Řada způsobů integrace je velmi pěkně a detailně popsána Gerogorem Hohpem a Bobby Wolfem ve vynikající knize Enterprise Integration Patterns. Vybrat nejlepší řešení je velmi obtížné. Rozhodli jsme se prozkoumat jednu z možností integrace služeb platformy GeoWeb. Tato možnost založená na Enterprise Service Bus (ESB) je poměrně nová dokonce i v oblasti integrace jiných služeb než jsou služby platformy GeoWeb. Příspěvek popíše dva příklady možné integrace, kdy v jednom případě je využití ESB neefektivní (resp. může integraci spíše zkomplikovat) a v druhém případě se integrace s využitím ESB jeví jako efektivní. V další části příspěvku se věnujeme výběru vhodného nástroje pro tvorbu ESB. Pro účely zkoumání byly testovány tři nástroje, které jsou distribuovány včetně zdrojového kódu: MULE ESB, Service Mix a Open ESB. V příspěvku budou všechny tři stručně popsány a porovnány. Poslední neméně důležitou částí pak bude popis prototypu ESB, který umožňuje zapojování služeb platformy GeoWeb. Prototyp byl vytvořen s využitím MULE ESB.

Abstract

INSPIRE directive prepared rules how to integrate several different geodata sources. If the sources are available according to the rules, anyone can create a tool generally named geoportal. A geoportal allows to access different geodata sources via one logical access point. This kind of tool is a typical example of a geoinformation systems integration. Such integration can be done in many specific ways. When dealing with information systems integration we always looking for solution that will be the most efficient. There are several ways of integration very nicely and in detail described by Gregor Hohpe and Bobby Woolf in the excellent book Enterprise Integration Patterns. To select the best option is very difficult. We have decided to research one possible option in the area of GeoWeb services. This option based on Enterprise Service Bus (ESB) is quite new even for integration based on non geospatial services. The paper will discus one example when is probably efficient to use ESB and one example when the usage of ESB will not be very efficient (may not be even effective). There is a second step, in a case of selection of ESB way, to select appropriate software tool. For research purposes were analysed three main open source solutions: MULE ESB, Service Mix and Open ESB. The paper is going to describe each of them very briefly and compare them from selected points of view. The last part of the paper will be focused on description of a prototype that allows plug-in GeoWeb Services into ESB selected for the research, which is MULE ESB.

Klíčová slova: ESB; MULE; Web Map Service; Monitoring

Keywords: ESB; MULE; Web Map Service; Monitoring

ÚVOD

Enterprise Service Bus je technologie, která pomáhá v případě integrace informačních systémů, v případě chybného využití však může samotnou integraci naopak zkomplikovat. Tato studie může pomoci v případě rozhodování zda technologii ESB nasadit v případě projektu, který je zaměřen na integraci geowebových služeb formou geoportálu. Studie je založena na teoretickém základu o integraci informačních systémů a vychází ze zkušeností při tvorbě pilotní aplikace, která je popsána v závěru příspěvku.

POUŽÍT NEBO NEPOUŽÍT ESB

K prvnímu zásadnímu rozhodnutí, zda vůbec do projektu nasadit ESB, můžeme využít články [Dörnenburg 2009] a [Mason 2009]. Oba články byly využity pro následující návod, který by měl pomoci s rozhodnutím.

1. Nakreslete schéma architektury vytvářeného systému. Toto schéma využijete v dalším kroku návodu.

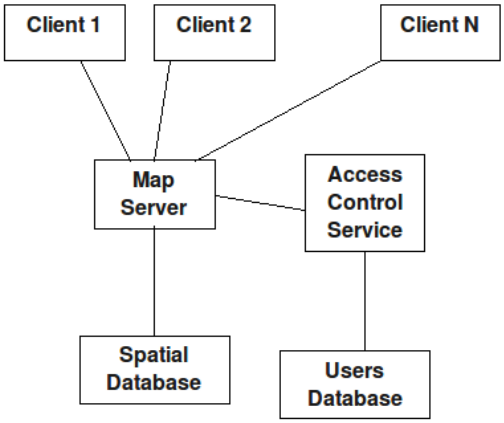
2. Odpovězte na otázky z [Mason 2009] ESB selection checklist (body 1-5 a 7-8).

3. Pokud odpovíte Ne na jakoukoli z otázek z bodu 2, pak ESB technologie není pravděpodobně vhodná pro váš systém.

4. Pokud se vám nedaří na všechny otázky jasně odpovědět Ano nebo Ne, můžete zkusit využít následující dva příklady, které popisují dva systémy, kde je nutná integrace.

Př. 1: Geoportál s jedním datovým skladem

Řada geoportálů je v současné době řešena podobně jako na následujícím schématu. ESB má řadu nástrojů jak monitorovat všechny operace, které jsou v systému prováděny. Mechanismy jako "message routing" a "filtering" mohou být např. využity pro kontrolu přístupu k datům. Takový typ geoportálu je vytvářen jednou společností, která jej zároveň provozuje. Tento typ geoportálu využívá pouze jednu lokální (centralizovanou) logickou databázi.



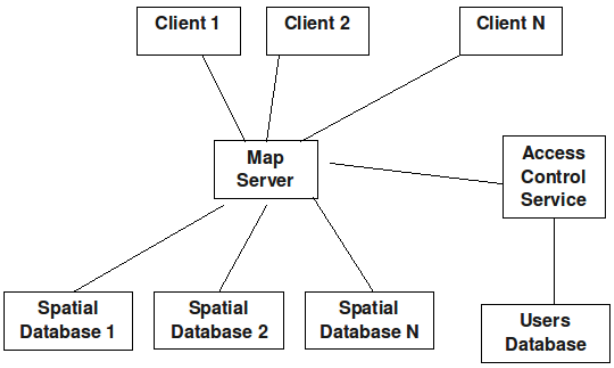
Obr. 1. Architektura geoportálu s jedním datovým skladem

Tab 1. Odpovědi pro geoportál s jedním datovým skladem.

Otázka	Odpověď	Poznámka
Integrujete tři nebo více aplikací/služeb?	NE	Integrace je pouze mezi mapovým serverem a službou pro kontrolu přístupu.
Bude v budoucnu opravdu potřebovat připojit další aplikace?	ANO	
Potřebujete více typů komunikačních protokolů?	ANO	HTTP pro klienty. přístup k databázi (nativní, JDBC), protokol pro přístup ke službě ke kontrole přístupu (SOAP, RMI, .NET Remoting).
Potřebujte využít směrování zpráv (message routing) v podobě jejich rozdělování a následné agregace nebo směrování založené na obsahu zprávy?	NE	Potřebujeme filtrování na základě obsahu, ale směrování ne.
Potřebujete vystavovat služby, tak aby je mohly využívat externí aplikace?	NE	HTTP je jediný komunikační protokol. Možná bude požadován FTP pro stažení dat, ale není nutný.

Př. 2: Geoportál s více datovými zdroji

Řada geoportálů je naopak řešena tak, že využívá více datových zdrojů a zpřístupňuje je uživateli. Cílem je připojit co nejvíce vhodných datových zdrojů. Schéma systému může být daleko více komplikované než na následujícím obrázku. Např. každý ze zdrojů může mít vlastní systém kontroly přístupových práv nebo může být pro kontrolu přístupových práv využíván systém federací.



Obr. 2. Architektura geoportálu s více datovými zdroji

Tab 2. Odpovědi pro geoportál s více datovými zdroji.

Otázka	Odpověď	Poznámka
Integrujete tři nebo více aplikací/služeb?	ANO	
Bude v budoucnu opravdu potřebovat připojit další aplikace?	ANO	
Potřebujete více typů komunikačních protokolů?	ANO	V budoucnu může být zájem připojit data dostupná prostřednictvím různých komunikačních protokolů (např. SOAP, REST, FTP)
Potřebujte využít směrování zpráv (message routing) v podobě jejich rozdělování a následné agregace nebo směrování založené na obsahu zprávy?	ANO	Potřebujeme směrování na základě obsahu zprávy. Podle obsahu bude následně zpráva zaslána na příslušný datový zdroj.
Potřebujete vystavovat služby, tak aby je mohly využívat externí aplikace?	ANO	Protože využíváme služby různého charakteru s využitím různých protokolů je jisté, že i naši partneři budou od nás očekávat vystavení našich služeb.

OPEN SOURCE SOFTWARE PRO TVORBU ESB

K dispozici je celá řada nástrojů pro tvorbu ESB. Testovány byly tři nástroje, které jsou k dispozici jako open source: Open ESB, Service Mix a MULE ESB. V plánu bylo také testovat nástroj Spring Integration, což se z časových důvodů bohužel nepodařilo.

Open ESB

Open ESB je vyvíjeno společností Sun (aktuálně Oracle). Je možné jej charakterizovat následujícími vlastnostmi:

- Založeno na technologiích EJB (Enterprise Java Beans) a JBI (Java Business Integration).
- Jako vývojové prostředí se užívá NetBeans IDE. NetBeans IDE zapouzdřují celý vývoj takovým způsobem, že se prakticky vývojář k samotnému ESB ani nedostane (black box).
- Konfigurace ESB je založena na jazyku BPEL (Business Process Execution Language). K dispozici je vizualizace nakonfigurovaného flow.
- Jako aplikační server je využit Glass Fish Server.
- K dispozici je řada předdefinovaných komponent pro připojování služeb (XMPP, SOAP, JMS, SAP, …).
- Základní dokumentace je v dobrém stavu. Detailní dokumentace je nekompletní.
- Dosti vysoké paměťové nároky.
- WSS (Web Service Security (OASIS)) by měla být k dispozici jako JBI komponenta. Nikde však není dostupná dokumentace k jejímu použití.

ServiceMix

ServiceMix vzniká v rámci Apache Group. Je možné jej charakterizovat následujícími vlastnostmi:

- Založeno na technologiích EJB (Enterprise Java Beans) a JBI (Java Business Integration).
- K dispozici není IDE pro vývoj. Vývoj je realizován s využitím technologie Maven.
- Konfigurace message flows je založena na jazyce Camel nebo je možné využít BPEL.
- Jako aplikační server lze použít Apache Tomcat nebo Apache Geronimo.
- K dispozici je řada předdefinovaných komponent pro připojování služeb (XMPP, SOAP, JMS, SAP, …).
- K dispozici je řada návrhových vzorů, které je možno přímo využít s pomocí nástroje Maven.
- Základní dokumentace je v dobrém stavu. Detailní dokumentace obsahuje řadu chyb.
- Je paměťově nenáročný.
- Kompilace a ladění je poměrně časově náročné. Pravděpodobně je možné celý proces urychlit lepší konfigurací nástroje Maven.
- Pro zpracování zpráv je využito ActiveMQ z dílny Apache Group.
- WSS (Web Service Security (OASIS)) by měla být k dispozici jako JBI komponenta. Nikde však není dostupná dokumentace k jejímu použití.

MULE ESB

MULE ESB je možné charakterizovat následujícími vlastnostmi:

- jako vývojové prostředí je možno využít Mule IDE založené na Eclipse.
- Konfigurace message flows je založena na vlastním jazyce. Je možné také zapojit externí BPEL konfiguraci. Od verze 3 je k dispozici také prostředí, kde se návrh provádí pomocí diagramů podobně jako v případě Open ESB. Tento způsob zatím nebyl testován.
- Jako aplikační server je využíván integrovaný Apache Tomcat.
- K dispozici je řada předdefinovaných komponent pro připojování služeb (XMPP, SOAP, JMS, …).
- Základní dokumentace je v dobrém stavu. Detailní dokumentace obsahuje řadu příkladů. Bohužel jak se ukázalo při vlastní implementaci existují zásadní rozdíly mezi verzemi ESB, kdy příklady nefungují v novějších verzích nebo naopak příklady vyžadují novou verzi. Diskuse ve fóru často končí s odkazem na možnost zaplacení podpory nebo konzultace.
- Je paměťově nenáročný.
- Pro zpracování zpráv je využito Mule MQ
- WSS (Web Service Security (OASIS)) by měla být k dispozici jako WSS4J interceptor. K dispozici jsou příklady jak jej použít.

Srovnání

Následující tabulka může pomoci v rozhodování jaký software zvolit. Bohužel je celé srovnání ovlivněno tím, že Open ESB a Service Mix, byly testovány pouze krátce, zatímco MULE bylo testováno delší dobu a vznikl v s jeho využitím i pilotní prototyp systému.

Tab 3. Srovnání na základě základních charakteristik

Vlastnost	Open ESB	ServiceMix	MULE
Architektura založena na	JBI	JBI	MULE (možné využít i JBI API)
Orchestrace	BPEL	Camel, BPEL	MULE, BPEL
Sběrnice	V paměti	V paměti, JMS	V paměti, JMS
Počet komponent	++	++	++
Implementace EIP	+	++	++
Přechovávání zpráv	BPEL engine, JDBC	JMS, JDBC, (BPEL engine)	JMS, JDBC, (BPEL engine)
Typy zpráv	založené na XML	založené XML	libovolné (včetně XML)
Integrace platformy Spring	-	++	++
Paměťové nároky	-	++	++
Clustering	++	++	+ (Chybí dokumentace)
Podpora Web Service Security (OASIS)	+	+	++
Správa ESB	+ (K dispozici je zdokumentované API zatím začínajícího projektu)	++ (K dispozici je JMX konzole a dokumentace)	+ (K dispozici je nezdokumentované API nebo placená verze Mule ESB management console)
Dokumentace	+ (Detailní nekompletní)	+ (Řada chyb)	+ (Problémy s verzemi)
Licence	CDDL	Apache	CPAL

PILOTNÍ PROTOTYP MULE-WMS

Systém umožňuje před existující server komunikující přes protokol WMS 1.1.1 (Web Map Service) předřadit ESB (Enterprise Service Bus) MULE. V rámci MULE je pak možno využívat konfiguraci zajišťující zejména zařazení transformátorů a loggerů. Transformátory umožňují změnit zaslaný WMS požadavek např. odfiltrovat vrstvy, které nejsou pro zvoleného uživatele zpřístupněny. Loggery umožňují záznam o průběhu filtrace do textového souboru a záznam o času zpracování WMS požadavku. Server byl testován pouze na OS GNU/Linux.

Logování

Logování se konfiguruje s využitím souboru log4j.properties, který se nachází v adresáři muledir/conf. Implicitně se log zapisuje do souborů muledir/bin/wms.log a muledir/bin/wmsmonitor.log. Standardní komponenty log-component a logger nebylo možno využít tak, aby zpráva obsahovala potřebné informace. Vytvořena byla komponenta CustomLogComponent, která dědí z komponenty LogComponent. Způsob logování se nepodařilo vyřešit zcela uspokojivě (viz Omezení a nevyřešené problémy).

Log wms

Soubor wms.log obsahuje pro každý požadavek klienta jeden záznam, který má strukturu podobnou jako Apache HTTP Custom Log Format (http://httpd.apache.org/docs/1.3/logs.html#common). Zaznamenán je datum a čas požadavku na MULE službu. V části obsahující požadavek jsou na rozdíl od Apache HTTP formátu uvedeny dva požadavky a je vynechána metoda (GET - je použita vždy) a protokol (vždy jde o

HTTP/1.1). První požadavek je požadavek zaslaný klientem. Druhý požadavek je požadavek po transformaci (např. na základě autorizace).

Log wmsmonitor

Soubor wmsmonitor.log obsahuje pro každý požadavek klienta jeden záznam. Záznam má tři položky. První je požadavek v takovém tvaru jak byl zaslán na WMS server (tj. po transformaci). Druhý je počet milisekund než byla z WMS serveru načtena odpověď. Třetí je počet bajtů, které představují odpověď WMS serveru.

Transformace

K dispozici je jednoduchá ukázka transformace na základě principu jednoduché konfigurace. Pro každého uživatele může být nadefinován soubor uzivatel.xml, který obsahuje Capabilities serveru s uvedením jen těch částí, na které má uživatel právo. Soubor uzivatel.xml se ukládá do adresáře muledir/apps/wms/classes/. Pro pilotní prototyp byla implementována transformace na základě ohraničujícího obdélníka a seznamu vrstev. Uživatel je dle dohody v rámci pilotního prototypu identifikován primitivně s využitím parametru USER HTTP/GET požadavku.

Ukázka části konfigurace ESB

```
    <!-- Prvni flow, ktere je spusteno pri vyvolani sluzby -->
    <flow name="WMS">
        <composite-source>

            <!-- Incoming HTTP requests -->
            <inbound-endpoint address="http://localhost:8888" exchange-
pattern="request-response">
                <not-filter>
                    <wildcard-filter pattern="/favicon.ico"/>
                </not-filter>
            </inbound-endpoint>

            <!-- Incoming Servlet requests -->
            <inbound-endpoint address="servlet://name" exchange-
pattern="request-response">
                <not-filter>
                    <wildcard-filter pattern="/favicon.ico"/>
                </not-filter>
            </inbound-endpoint>

        </composite-source>

            <!-- Zaznam zacatku transakce -->
            <component class="cz.vsb.gis.ruz76.wms.CustomLogComponent"/>

            <!-- Transformace HTTP/GET pozadavku na WMS Request -->
            <transformer ref="HttpRequestToWMSRequest"/>

            <!-- V zavislosti na vysledku predchozi transformace se rozhoduje o
vyvolani prislusne flow. V pripade,
            ze pro transformaci dojde k chybe je chyba vypsana do mule.log -->

        <choice>
            <when expression="payload instanceof
org.geotools.data.wms.request.GetMapRequest" evaluator="groovy">
```

```
                <vm:outbound-endpoint path="getmap" exchange-pattern="request-
response"/>
            </when>
            <when expression="payload instanceof
org.geotools.data.ows.GetCapabilitiesRequest" evaluator="groovy">
                <vm:outbound-endpoint path="getcapabilities" exchange-
pattern="request-response"/>
            </when>
            <when expression="payload instanceof
org.geotools.data.wms.WMS1_1_1.GetFeatureInfoRequest" evaluator="groovy">
                <vm:outbound-endpoint path="getfeatureinfo" exchange-
pattern="request-response"/>
            </when>
            <when expression="payload instanceof java.lang.Exception"
evaluator="groovy">
                <vm:outbound-endpoint path="userErrorHandler" exchange-
pattern="request-response"/>
            </when>
        </choice>

        <!-- Neocekavane chyby -->
        <default-service-exception-strategy>
            <vm:outbound-endpoint path="systemErrorHandler" exchange-
pattern="one-way"/>
        </default-service-exception-strategy>
    </flow>

    <!-- Chyba transformace HTTP/GET pozadavku je prevedena na retezec a ten je
odeslan zpet klientovi.
         Pozn. Pokud service nebo flow nekonci outbound prvkem dojde k
automaickemu odeslani ke klientovi.
     -->
    <model>
        <service name="UserErrorHandler">
            <inbound>
                <vm:inbound-endpoint path="userErrorHandler"
responseTransformer-refs="ExceptionToString" exchange-pattern="request-
response"/>
            </inbound>
        </service>
    </model>

    <!-- Zpracovani GetMap pozadavku -->
    <flow name="GetMap">
        <!-- Pozadavek je filtrovan v pripade, ze obsahuje parametr USER -->
      <vm:inbound-endpoint path="getmap" transformer-
refs="WMSGetMapRequestToFilteredWMSGetMapRequest" exchange-pattern="request-
response"/>
      <choice>
          <!-- Pokud filtrace dopadla uspokojive je pozadavek odeslan na WMS
server -->
            <when expression="payload instanceof
org.geotools.data.wms.request.GetMapRequest" evaluator="groovy">
                <!-- Fitrovany pozadavek je preveden na retezec -->
```

```
                <transformer ref="FilteredWMSRequestToString" />
                <!-- Je zaznamenan zacatek volani WMS sluzby -->
                <component class="cz.vsb.gis.ruz76.wms.CustomLogComponent" />
                <!-- Volani WMS sluzby. Parametry jsou nacteny z wms.properties
-->
            <http:outbound-endpoint name="wmsservergetmap" host="${wmshost}"
port="${wmsport}" path="${wmspath}#[payload:java.lang.String]" method="GET"/>
            <!-- Tady dochazelo z nejakeho duvodu k chybe pokud zustal payload
typu ReleasingInputStream a pak se logovalo. Pred logovanim je tedy preveden na
ByteArray -->
                  <transformer ref="ReleasingInputStreamToByteArray"/>
                  <!-- Zaznam ziskani vysledku odpovedi WMS serveru a jeho vypis
do log souboru. -->
                  <component class="cz.vsb.gis.ruz76.wms.CustomLogComponent" />
                  <!-- Nastaveni Content-type odpovedi klientovi na zaklade
Content-type odpovedi WMS sluzby -->
                  <message-properties-transformer scope="outbound">
                              <add-message-property key="Content-Type"
value="#header:INBOUND:Content-Type"/>
                        </message-properties-transformer>
            </when>
            <!-- Pokud filtrace odstranila vsechny vrstvy, klient obdrzi vyjimku
-->
            <when expression="payload instanceof
org.geotools.ows.ServiceException" evaluator="groovy">
                <!-- Zaznam do logu je nutny, kvuli zachovani tri zaznamu. Popis
v DOkumentace.pdf -->
                  <component class="cz.vsb.gis.ruz76.wms.CustomLogComponent" />
                  <!-- Prevedeni vyjimky do XML -->
                  <transformer ref="ServiceExceptionToXML"/>
                  <!-- Vymazani Content-type -->
                  <transformer ref="delete-content-type-header" />
                  <!-- Pridani Content-type=text/xml. Mozna zmenit na vnd.ogc
-->
                  <transformer ref="add-xml-content-type-header" />
                  <!-- Zaznam odeslani odpovedi klientovi. -->
                  <component class="cz.vsb.gis.ruz76.wms.CustomLogComponent" />
            </when>
        </choice>

    </flow>

    <!-- Zpracovani GetCapabilities pozadavku -->
    <flow name="GetCapabilities">
        <!-- Pozadavek je preveden na retezec -->
      <vm:inbound-endpoint path="getcapabilities" transformer-
refs="FilteredWMSRequestToString" responseTransformer-refs="delete-content-type-
header add-xml-content-type-header" exchange-pattern="request-response"/>
      <!-- Je zaznamenan zacatek volani WMS sluzby -->
      <component class="cz.vsb.gis.ruz76.wms.CustomLogComponent" />
      <!-- Volani WMS sluzby. Parametry jsou nacteny z wms.properties -->
      <http:outbound-endpoint name="wmsservergettapabilities" host="${wmshost}"
port="${wmsport}" path="${wmspath}#[payload:java.lang.String]" method="GET"/>
```

```
      <!-- Odpoved je upravena. Nahrazeny jsou cesty ke sluzbe, tak aby
odpovidaly ceste k MULE sluzbe a nikoli k volanemu WMS serveru. -->
      <transformer ref="CapabilitiesToProxyCapabilities" />
      <!-- Zaznam ziskani vysledku odpovedi WMS serveru a jeho vypis do log
souboru. -->
      <component class="cz.vsb.gis.ruz76.wms.CustomLogComponent" />
      <!-- Nastaveni Content-type odpovedi klientovi na zaklade Content-type
odpovedi WMS sluzby -->
      <message-properties-transformer scope="outbound">
                  <add-message-property key="Content-Type"
value="#header:INBOUND:Content-Type"/>
            </message-properties-transformer>
    </flow>

    <!-- Zpracovani GetFeatureInfo pozadavku -->
    <flow name="GetFeatureInfo">
      <!-- Pozadavek je preveden na retezec -->
      <vm:inbound-endpoint path="getfeatureinfo" transformer-
refs="FilteredWMSRequestToString" responseTransformer-refs="delete-content-type-
header add-xml-content-type-header" exchange-pattern="request-response"/>
      <!-- Je zaznamenan zacatek volani WMS sluzby -->
      <component class="cz.vsb.gis.ruz76.wms.CustomLogComponent" />
      <!-- Volani WMS sluzby. Parametry jsou nacteny z wms.properties -->
      <http:outbound-endpoint name="wmsservergetfeatureinfo" host="${wmshost}"
port="${wmsport}" path="${wmspath}#[payload:java.lang.String]" method="GET"/>
      <!-- Tady dochazelo z nejakeho duvodu k chybe pokud zustal payload typu
ReleasingInputStream a pak se logovalo. Pred logovanim je tedy preveden na
ByteArray -->
      <transformer ref="ReleasingInputStreamToByteArray"/>
      <!-- Zaznam ziskani vysledku odpovedi WMS serveru a jeho vypis do log
souboru. -->
      <component class="cz.vsb.gis.ruz76.wms.CustomLogComponent" />
      <!-- Nastaveni Content-type odpovedi klientovi na zaklade Content-type
odpovedi WMS sluzby -->
      <message-properties-transformer scope="outbound">
                  <add-message-property key="Content-Type"
value="#header:INBOUND:Content-Type"/>
            </message-properties-transformer>
    </flow>

    <!-- Vypis neocekavanych chyb do konzole a mule.log souboru -->
    <flow name="SystemErrorHandler">
        <vm:inbound-endpoint path="systemErrorHandler" exchange-
pattern="request-response"/>
        <outbound-endpoint address="stdio://ERR" exchange-pattern="one-way"/>
    </flow>
```

Omezení a nevyřešené problémy

- Podporována je pouze metoda GET protokolu HTTP.
- Podporována je pouze WMS 1.1.1.
- Uživatel je identifikován pouze s využitím parametru nikoli třeba sessionid.

- Transformace je implementována pouze pro ohraničující obdélník a seznam vrstev. Transformace je implementována pouze pro požadavek GetMap. V případě, že je požadován extent větší než povolený je vrstva z požadavku odstraněna nikoli ořezána.
- Při prvním požadavku na MULE po jeho spuštění neodejde požadavek na WMS korektně. Zřejmě bug v MULE. Další požadavky už odcházejí korektně.
- CustomLogComponent je pevně vázána na trojí volání v rámci konfiguračního souboru mule-config.xml. Poprvé se volá při získání požadavku od klienta. Podruhé se volá po transformaci požadavku, a před posláním požadavku na WMS. Potřetí se volá po obdržení odpovědi od WMS.
- Přes wms.properties není možné přidat vendor speciffic parameters pro WMS. Je nutné to udělat přímo v muledir/apps/wms/mule-config.xml.

Testování performance

Cílem bylo zjistit jakým způsobem ovlivní rychlost zpracování požadavků zařazení MULE před WMS server. Bylo provedeno celkem 10 nezávislých testů s využitím nástroje WMSTester (http://sourceforge.net/projects/wmstester/ - autor Michal Šeliga). Testování bylo provedeno pouze na běžném PC a to tak aby byl procesor zatížen na max. 90% výkonu. V rámci každého testu bylo spuštěno 10 paralelních klientů, přičemž, každý zaslal 20 požadavků. Interval mezi každým z požadavků byla 1s.

Průměrná doba odezvy na požadavek v případě přímého přístupu na WMS server: 187 ms

Průměrná doba odezvy na požadavek v případě předřazení MULE: 201 ms

Nárůst v případě předřazení MULE je tedy necelých 8% času.

ZÁVĚR

V říjnu 2010 byl pro vývoj prototypu zvolen nástroj MULE. Přestože se již v té době ServiceMix jevil jako lepší nástroj, MULE bylo zvoleno především z toho důvodu, že základní dokumentace neobsahovala chyby a vývoj byl snadnější než v případě ServiceMix. Následně se ukázalo, že při řešení již specifických úkolů dokumentace nedostačuje a jsou problémy zejména se změnou API mezi verzemi. Z časových důvodů už však nebylo možné se vrátit k nástroji ServiceMix a tento zkusit využít pro vývoj prototypu. Pokud by však mělo dojít k rozhodnutí dnes, volba by padla pravděpodobně na ServiceMix. Vytvořený prototyp je v současné době testován, v případě, že se osvědčí, bude doplněn o možnosti filtrování WFS požadavků.

PODĚKOVÁNÍ

Rád bych poděkoval společnosti T-Mapy spol. s r.o., která finančně podpořila vývoj prototypu z vlastních prostředků a dotace Moravskoslezského kraje v rámci projektu Monitor služeb.

LITERATURA

Apache Group (2011). ServiceMix. http://servicemix.apache.org/home.html

Dörnenburg, E (2009). Making ESB pain visible. http://erik.doernenburg.com/2009/07/making-esb-pain-visible/

Mason, R (2009). To ESB or not to ESB. http://rossmason.blogspot.com/2009/07/to-esb-or-not-to-esb.html

MuleSoft Inc. (2011). MULE ESB. http://www.mulesoft.org/

Oracle (2011). Open ESB. https://open-esb.dev.java.net/

Wheeler, J (2009). Using Interceptors. http://www.mulesoft.org/documentation/display/MULE2USER/Using+Interceptors