PERBANDINGAN ANTARA "BIG" WEB SERVICE DENGAN RESTFUL WEB SERVICE UNTUK INTEGRASI DATA BERFORMAT GML

Adi Nugroho[1] dan **Khabib Mustofa**[2]

[1] Fakultas Teknologi Informasi – Universitas Kristen Satya Wacana, Salatiga – Jawa Tengah & Mahasiswa program S3 Fakultas Matematika dan Ilmu Pengetahuan Alam – Universitas Gadjah Mada di Jogyakarta.

[2] Fakultas Matematika dan Ilmu Pengetahuan Alam – Universitas Gadjah Mada, Jogyakarta.

ABSTRAK: Teknologi *Java Web Service* berbasis SOAP *(JAX-WS/Java API for XML Web Service)* dan *Java RESTful Web Service (JAX-RS/Java API for XML RESTful Web Service)* saat ini merupakan teknologi yang saling bersaing satu sama lain dalam hal penggunaannya untuk mengintegrasikan data yang berada di berbagai sistem yang memiliki baik platform perangkat keras maupun perangkat lunak (sistem operasi) yang berbeda. Kedua teknologi *Web Service* itu tentu saja memiliki kekurangan dan kelebihannya masing-masing. Dalam tulisan ini, kami membahas perbandingan kedua teknologi *Java Web Service* itu dalam kaitannya dengan pengembangan aplikasi SIG *(Sistem Informasi Geografis)* terintegrasi yang menggunakan data berformat GML *(Geography Markup Language)*, yang disimpan di dalam sistem basisdata XML *(eXtensible Markup Language)*.

Kata kunci: *Web service, GIS, integrasi data spasial, "Big" web service, RESTful web service, java web service*

ABSTRACT: *Web Service with Java: SOAP (JAX-WS/Java API for XML Web Services) and Java RESTful Web Service (JAX-RS/Java RESTful API for XML Web Services) are now a technology competing with each other in terms of their use for integrates data residing in different systems. Both Web Service technologies, of course, have advantages and disadvantages. In this paper, we discuss the comparison of the two technologies is a Java Web Service in relation to the development of GIS application (Geographic Information System) integrates the use of data-formatted GML (Geography Markup Language), which is stored in the system database XML (eXtensible Markup Language)*.

Keywords: *Web service, GIS, integration of spatial data, "Big" web service, RESTful web service, java web service*

PENDAHULUAN

Para pengembang aplikasi *Geographical Information System*/Sistem Informasi Geografis (SIG) saat ini memiliki 2 alternatif untuk sistem basisdata yang akan digunakannya, yaitu menggunakan sistem-sistem basisdata relasional yang mampu menyimpan data berformat spasial di kolom-kolom di dalam tabelnya (*XML Enabled Database*), misalnya Oracle 10g, SQL Server 2008, PostgreSQL 9, dan sebagainya, atau menggunakan sistem basisdata *Native XML Database* (XML), misalnya Oracle Berkeley DB XML, eXist, BaseX, MonetDB/ XQuery, EMC xDB, Sedna, dan sebagainya, untuk menyimpan data spasial dalam bentuk/format data *Geography Markup Language* (GML) [13, 20]. Dalam hal ini, GML yang didefinisikan oleh Open Geospatial Consortium, Inc., sesungguhnya merupakan suatu 'kosa kata' lain dari XML *eXtensible Markup Language* (XML) yang memang dikhususkan untuk menyimpan data spasial [4, 18].

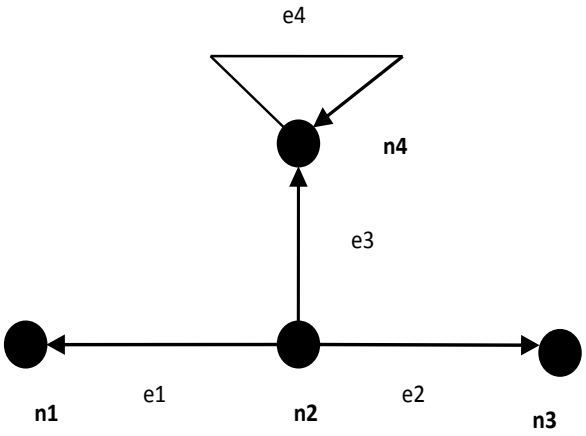
Saat mengembangkan aplikasi-aplikasi SIG terintegrasi (misalnya aplikasi-aplikasi SIG yang berjalan di lingkup jaringan komputer atau Internet), para pengembang seringkali juga membutuhkan data spasial dari berbagai sumber, baik dari sistem basis data relasional maupun dari sistem basis data XML (*Native XML Database*) yang mampu menyimpan data dalam format GML. Dalam hal ini, jika pengembang aplikasi SIG menggunakan bahasa pemrograman Java, saat ini mereka memiliki 2 pilihan untuk melakukan integrasi data spasial itu, yaitu menggunakan teknologi Java Web Service berbasis *Simple Object Access Protocol* (SOAP) (sering disebut sebagai "Big" Web Service), atau menggunakan REST (*REpresentational State Transfer*) Web Service (sering disebut sebagai RESTful Web Service) [12]. Kedua teknologi Web Service ini tentu saja memiliki keunggulan dan kelemahannya masing-masing saat bekerja dengan data spasial berformat GML [11, 12, 14]. Tujuan tulisan ini pada dasarnya adalah melakukan perbandingan secara teknis terhadap kedua teknologi Java Web Service ini saat keduanya bekerja untuk data spasial berformat GML yang disimpan dalam sistem basis data XML. Dalam hal ini, sebagai batasan permasalahan, kami tidak akan melakukan pembahasan teknis terhadap sistem basisdata relasional konvensional (meskipun

beberapa sistem basisdata relasional terkemuka saat ini mampu menyimpan data spasial di dalam kolom-kolomnya).

GML

Geographic Markup Language (GML), yang saat ini sering digunakan oleh SIG-SIG dan telah dibakukan oleh Open Geospatial Consortium, Inc., sesungguhnya adalah suatu berkas *eXtensible Markup Language* (XML) yang khusus dikembangkan untuk menyimpan data spasial [7]. GML sesungguhnya merupakan sebuah 'kosa kata' XML yang ditulis dalam bentuk XML Schema khusus yang digunakan untuk memodelkan, mentransportasikan, dan menyimpan data spasial dan informasi geografis yang terkait [7]. GML memiliki sejumlah besar objek untuk mendeskripsikan objek-objek geografis, termasuk di dalamnya fitur-fitur, sistem-sistem koordinat rujukan, bentuk-bentuk geometri, topologi, waktu, unit-unit pengukuran, dan nilai-nilai yang bersifat umum (misalnya warna, suhu, dan sebagainya) [7]. Dalam hal ini, suatu fitur geografis sesungguhnya merupakan "suatu abstraksi dari fenomena yang memang hadir di dunia nyata, dan ia merupakan suatu fitur geografis jika ia berhubungan secara langsung dengan suatu lokasi tertentu di permukaan bumi" [7, 10]. Dengan demikian, representasi digital dari dunia nyata sesungguhnya dapat dipikirkan sebagai sejumlah fitur-fitur geografis. Dalam hal ini, suatu fitur geografis sesungguhnya dapat didefinisikan menggunakan sejumlah properti, dimana masing-masing properti dapat dipikirkan sebagai suatu kesatuan nama (*name*), jenis (*type*), dan nilai (*value*) [10]. Fitur-fitur geografis dalam GML mencakup di dalamnya tutupan (*coverage*) dan nilai-nilai observasi langsung di lapangan sebagai subtype-nya. Suatu tutupan (*coverage*) dapat merepresentasikan suatu fitur atau sejumlah fitur yang digunakan untuk "memodelkan dan memperlihatkan hubungan-hubungan spasial dan sebaran spasial di antara fenomena-fenomena yang ada di permukaan bumi" [10].

Saat pengembang aplikasi SIG akan menggunakan data berformat GML untuk aplikasi SIG-nya, mereka harus terlebih dahulu memiliki dokumen GML Schema (suatu bentuk khusus dari XML Schema yang biasanya berekstensi xsd) yang berguna untuk mendefinisikan tipe-tipe data dan juga mendefinisikan elemen-elemen XML yang berhubungan langsung dengan objek-objek GML yang kelak akan dibuat. GML Schema ini bisa dibuat dengan memanfaatkan spesifikasi yang dikeluarkan oleh Open Geospatial Consortium Inc. [7]. Sementara itu, dokumen-dokumen GML yang menyimpan informasi geografis, yang sering juga disebut sebagai XML Instance Document (yang biasanya berekstensi xml), selanjutnya bisa dikembangkan dengan basis penafsiran GML Schema [10, 13, 18]. Secara umum, spesifikasi GML bersifat cukup terbuka serta bersifat lintas-*platform*, sehingga sangat layak untuk digunakan sebagai sarana pertukaran data spasial melintas berbagai SIG yang sangat beragam jumlahnya di dunia ini [7].



Gambar 1. Jaringan Sederhana

Untuk memperlihatkan bagaimana GML dapat digunakan untuk mendeskripsikan data spasial/ geografis, kita melihat contoh penerapannya berikut ini. Gambar 1 merepresentasikan rekonstruksi simpul (*node*) dan garis (*edge*) yang, sebagai contoh saja, secara sederhana menggambarkan rute pemberhentian bus di suatu jaringan jalan raya. Simpul-simpul, demi kemudahan, dianggap tidak memiliki struktur internal di dalamnya. Sementara itu, garis-garis memiliki simpul awal (yang memiliki nilai negatif) dan simpul akhir (yang memiliki nilai positif) yang dideskripsikan sebagai pasangan properti-properti simpul berarah, yang selanjutnya akan membentuk batasan garis yang bersangkutan. Dengan data diatas, kita bisa memberikan pertanyaan-pertanyaan yang berkaitan dengan "bagaimana urutan garis-garis yang diperlukan untuk melakukan pergerakan dari suatu simpul ke simpul yang lainnya?" Sebagai contoh, pergerakan dari n1 ke n3 akan memiliki urutan {+e1, +e2}, dan dari n3 ke n4 memiliki urutan {-e2, +e3}. Dalam hal ini, perhatikanlah bahwa kita dapat merujuk garis-garis dalam 2 arah. Karena suatu garis pada dasarnya memiliki arah yang bersifat implisit (sebagai contoh, e1 dapat dikatakan berarah dari n1 ke n2), garis itu dapat dilintasi baik pada arah itu maupun pada arah sebaliknya. Dengan demikian, -e1 dapat dikatakan bergerak dari n2 ke n1.

Jika kita melihat jaringan sederhana (*simple network*) seperti yang diperlihatkan Gambar 1, maka sesungguhnya kita dapat merepresentasikannya dalam bentuk simpleNetwork.xml berikut, yang ditulis menggunakan spesifikasi yang telah dikenal oleh GML [7].

```
<?xml version="1.0" encoding="UTF-8"?>
<Topology
xmlns="http://www.opengis.net/app"
xmlns:sch="http://www.ascc.net/xml/sche
matron"
xmlns:xlink="http://www.w3.org/1999/xli
nk"
xmlns:gml="http://www.opengis.net/gml"
xmlns:app="http://www.opengis.net/app"
xmlns:xsi="http://www.w3.org/2001/XMLSc
hema-instance"
xsi:schemaLocation="http://www.opengis.
net/app networkExamples.xsd">
<gml:Node gml:id="n1"/>
<gml:Node gml:id="n2"/>
<gml:Node gml:id="n3"/>
<gml:Node gml:id="n4"/>
<gml:Edge gml:id="e1">
<gml:directedNode orientation="-"
xlink:href="#n1"/>
<gml:directedNode orientation="+"
xlink:href="#n2"/>
</gml:Edge>
<gml:Edge gml:id="e2">
<gml:directedNode orientation="-"
xlink:href="#n2"/>
<gml:directedNode orientation="+"
xlink:href="#n3"/>
</gml:Edge>
<gml:Edge gml:id="e3">
<gml:directedNode orientation="-"
xlink:href="#n2"/>
<gml:directedNode orientation="+"
xlink:href="#n4"/>
</gml:Edge>
<gml:Edge gml:id="e4">
<gml:directedNode orientation="-"
xlink:href="#n4"/>
<gml:directedNode orientation="+"
xlink:href="#n4"/>
</gml:Edge>
</Topology>
```

Berkas GML di atas adalah representasi jaringan sederhana (*simple network*) dalam bentuk XML seperti yang diperlihatkan dalam Gambar 1. Dokumentasi spesifikasi GML yang diluncurkan oleh Open Geospatial Consortium, Inc. tentunya mendeskripsikan sintak-sintak untuk objek-objek yang jauh lebih kompleks dari yang diperlihatkan dalam jaringan sederhana di atas [7, 10]. Selain itu, simpul-simpul (*node*) yang ada mungkin saja dikembangkan lebih lanjut menjadi gambaran objek-objek kompleks yang sesungguhnya (misalnya stasiun pemberhentian bus yang memiliki banyak objek lain di dalamnya), yang menggambarkan fakta yang ada di dunia nyata. Garis-garis (*edge*) mungkin juga dikembangkan lebih lanjut menjadi garis-garis majemuk yang juga menggambarkan suatu peta jalan sesungguhnya yang memang dijumpai di dunia nyata.

BASIS DATA XML

Seperti telah kita singgung di atas, para pengembang SIG, saat mereka akan mengembangkan aplikasi-aplikasi SIG, memiliki 2 pilihan untuk sistem basis data yang mendasari, yaitu sistem basis data relasional yang mampu menyimpan data berformat XML di dalamnya (*XML Enabled Database*), atau sistem basis data XML (*XML Native Database*), yang dirancang secara khusus untuk mengelola dan menyimpan data dalam bentuk XML (atau dalam konteks aplikasi SIG, dalam bentuk GML) [13]. Dalam hal ini, untuk aplikasi-aplikasi SIG, sistem basis data XML (*XML Native Database*) memiliki keunggulan dibandingkan sistem basis data relasional konvensional (*XML Enabled Database*) karena kecepatannya [2, 4, 5]. Alasan untuk hal ini adalah bahwa, saat kita menggunakan sistem basisdata relasional, data spasial harus dikonversi terlebih dahulu (menggunakan bahasa nirprosedural SQL [*Structured Query Language*] dan prosedur-prosedur/ fungsi-fungsi tersimpan [*stored procedure/function*] yang terkait dengannya) menjadi data berformat XML/GML untuk kebutuhan visualisasi di aras aplikasi SIG [2, 4, 5].

Jika pengembang aplikasi SIG menggunakan sistem basisdata relasional konvensional, mereka harus menggunakan SQL untuk melakukan manipulasi data yang tersimpan dalam tabel-tabel. Sementara itu, untuk sistem basis data XML (dalam hal ini, kami menggunakan Oracle Berkeley DB XML [BDB XML] yang dapat diunduh secara bebas dari situs milik Oracle Corp.), untuk menyimpan dokumen XML/GML kita harus menggunakan API (*Application Programming Interface*) XQuery [17]. (XQuery API for Java adalah salah satu API dalam bahasa Java yang berpadanan dengan SQL, hanya saja alih-alih melakukan *query* pada tabel relasional, XQuery bekerja dengan cara melakukan *query* atas berkas-berkas berformat XML [17].) Dalam hal ini, saat kita akan menyimpan dokumen XML/GML di BDB XML, pertama kali kita harus membuat pemuat (*container*) menggunakan perintah XQuery `create Container()`, dan selanjutnya kita menyimpan dokumen XML/GML di dalamnya menggunakan perintah `putDocument()` dan hal ini dilakukan lewat *shell* yang dimiliki BDB XML. Selanjutnya, setelah dokumen XML/GML berhasil disisipkan ke dalam pemuat, kita kemudian bisa melakukan *query* terhadap berkas XML/GML yang disimpan dalam pemuat tersebut menggunakan perintah-perintah XQuery. Dalam hal ini, tentu saja perintah-perintah

XQuery sesungguhnya cukup kompleks, sesuai dengan kompleksitas XML/GML yang akan di-*query* [17]. Meskipun demikian, dalam tulisan ini, kami hanya membahas operasi CRUD (*Create-Read/ Retrieve-Update-Delete*) terhadap sistem basisdata BDB XML yang bersifat mendasar saja. Dalam hal ini, alih-alih bekerja langsung melakukan *query* XQuery menggunakan *shell* BDB XML, demi menghindari kerumitan-kerumitannya, kami melakukannya dari arah aplikasi yang ditulis menggunakan bahasa pemrograman Java.

BDB XML bekerja atas dokumen XML menggunakan kelas `XmlDocument`, meski demikian di belakang layar, BDB XML sesungguhnya menggunakan API Xerces DOM (*Document Object Model*) untuk menyimpan dan memanipulasi dokumen-dokumen XML [5, 11]. Seperti yang telah diperlihatkan sebelumnya, dokumen XML dapat ditambahkan ke pemuat menggunakan metoda `Xml Container. putDocument()` [5]. Saat dokumen XML dibuat menggunakan kelas `XmlDocument` yang berbasis bahasa Java, alih-alih bekerja menggunakan *shell* BDB XML, pembuatannya bisa dilakukan menggunakan kode-kode Java di bawah ini.

```
document = manager.createDocument()
document.setName('simpleNetwork')
document.setContent("<document
gml:id='n1'></document>")
container.putDocument(document,
manager.createUpdateContext())
```

Selanjutnya, kita membahas operasi pembacaan dokumen XML dari pemuat (*container*). Dalam hal ini, pada objek `XmlResults` yang dihasilkan dari `XmlManager.Query()` sebelumnya dapat dilakukan iterasi (*looping*) untuk melihat isi dokumen XML yang telah dibuat sebelumnya. Adapun kode dalam bahasa Java-nya adalah sebagai berikut.

```
container =
manager.openContainer("TEST")
results =
manager.query("collection('TEST')/Word",
manager.createQueryContext())
for value in results:
document = value.asDocument()
print document.getName()
print document.getContent()
```

Adapun pembaharuan (*updation*) isi dokumen XML yang ada di pemuat juga dapat dilakukan dengan cara yang relatif sederhana dengan menggunakan metode `setContent()` dan `update Document()` seperti terlihat dalam kode-kode Java berikut ini.

```
container =
manager.openContainer("TEST")
document =
container.getDocument("simpleNetwork")
document.setContent("<document
gml:id='n1'>Road</document>")
container.updateDocument(document,
manager.createUpdateContext())
```

Dalam hal ini, penghapusan isi dokumen XML juga dapat dilakukan menggunakan metode `update Document()` dengan parameter kosong. Meski demikian, metode `updateDocument()` lebih sering digunakan saat bagian suatu dokumen XML -isi atau metadata- diperbaharui (seperti yang kita lakukan di atas). Berkaitan dengan hal ini, jika kita mau menghapus suatu dokumen XML dari suatu pemuat, kita, sebagai alternatif, kita juga bisa menggunakan metode `deleteDocument()`, seperti yang terlihat pada contoh kode-kode bahasa Java di bawah ini.

```
container =
manager.openContainer("TEST")
updateContext =
manager.createUpdateContext()
document =
container.getDocument("simpleNetwork")
container.deleteDocument(document,
updateContext)
```

Terlihat dari kode-kode Java di atas, pada prinsipnya kita, pada sistem basis data XML, melakukan akses lebih langsung pada berkas XML, alih-alih memanfaatkan kerja server basis data seperti yang terjadi pada sistem basis data relasional. Itulah sebabnya penggunaan sistem basis data XML untuk aplikasi-aplikasi SIG memiliki kinerja yang relatif lebih baik (lebih cepat) daripada penggunaan sistem basisdata relasional konvensional [5, 20]. Selain itu, jaminan kinerja sistem basisdata XML juga terjadi akibat dari tidak diperlukannya konversi dari data yang ada pada kolom (*field*) tabel relasional menjadi data berformat XML/GML seperti yang terjadi saat pengembang aplikasi SIG menggunakan sistem basisdata relasional konvensional [2, 5, 20].

Seperti dapat kita lihat diatas, operasi-operasi CRUD pada sistem basis data XML, dapat dilakukan menggunakan API XQuery, baik menggunakan *shell* BDB XML, atau dari arah aplikasi yang dibuat menggunakan bahasa pemrograman Java. Tantangan selanjutnya, adalah bagaimana caranya operasi CRUD itu dapat dilakukan dalam konteks Web Service sebab tujuan kita sesungguhnya adalah melakukan perbandingan-perbandingan aplikasi SIG terintegrasi yang menggunakan sistem basis data XML di dalam konteks Java Web Service.

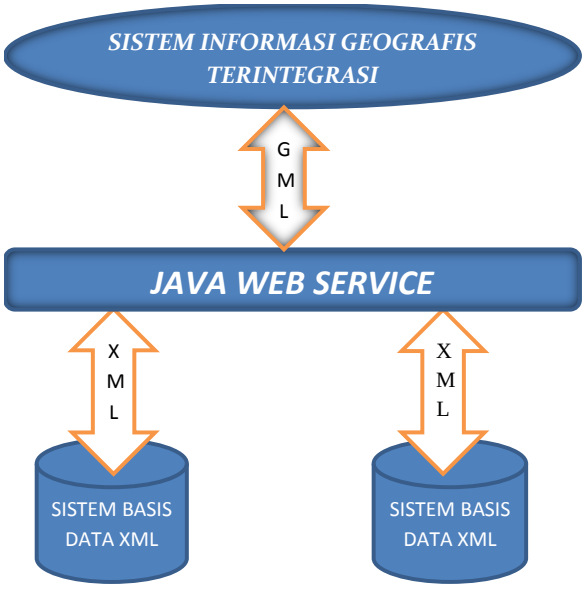
JAVA WEB SERVICE

Web Service sesungguhnya merupakan suatu perangkat lunak/aplikasi yang menyediakan layanan (*service*) bagi aplikasi-aplikasi lainnya (aplikasi klien)

dan mengikuti aturan yang selama ini didefinisikan oleh Web Service Interoperability Organization (WSIO) [8, 15]. Web Service memungkinkan dikembangkannya suatu aplikasi yang memanfaatkan layanan (aplikasi klien) yang disediakan oleh penyedia layanan (server), tanpa aplikasi yang bersangkutan terlalu perlu peduli pada *platform* perangkat keras dan perangkat lunak (sistem operasi) tempat layanan dikembangkan [8, 15]. Baik server layanan maupun kliennya dapat saling berkomunikasi menggunakan protokol HTTP (*HyperText Transport Protocol*) [8, 15]. Dalam konteks penggunaan protokol transpor HTTP ini, pendekatan Web Service dalam konteks bahasa pemrograman Java sering dikategorikan menggunakan 2 hal yang berkaitan dengan pesan (*message*) yang dikirimkan, yaitu: SOAP over HTTP ("Big" Web Service) serta POX-HTTP (*Plain Old XML over HTTP*) (RESTful Web Service) [11].

Saat kita membicarakan Web Service dalam konteks bahasa pemrograman Java, pembahasan kita tentang XQuery di bagian sebelumnya menjadi sangat relevan. Operasi-operasi CRUD yang dilakukan atas dokumen-dokumen XML, menggunakan teknik-teknik tertentu yang akan kita bahas selanjutnya, bisa langsung dipetakan menjadi operasi-operasi CRUD yang berpadanan dalam konteks Java Web Service. Meski demikian, sebelum kita membahas operasi-operasi XQuery dalam konteks Java Web Service, terlebih dahulu kita akan membahas tentang Web Service itu sendiri dalam konteks bahasa pemrograman Java karena sesungguhnya Web Service tidak hanya dikenal dalam konteks bahasa pemrograman Java. Teknologi Microsoft .NET misalnya, juga memiliki teknologi .NET Web Service yang bermanfaat untuk mengintegrasikan aplikasi-aplikasi yang memiliki platform dasar .NET (terutama yang masing-masing berjalan di atas sistem operasi Windows) [3].

Implementasi Web Service pada Java EE 6 (dan yang terbaru saat ini Java EE 7) menggunakan teknologi JAX-WS (*Java API for XML Web Service*) sering dinamakan sebagai "Big" Web Service. "Big" Web Service selalu menggunakan berkas-berkas XML yang mengikuti standar SOAP sebagai sarana dan format pertukaran pesan (*message*) antar sistem. Dalam hal ini, jika "Big" Web Service menggunakan berkas-berkas SOAP sebagai sarana pertukaran pesannya, "Big" Web Service juga menggunakan berkas XML lain yang disebut sebagai WSDL (*Web Services Description Language*) sebagai sarana bagi "Big" Web Service yang bersangkutan untuk mendefinisikan layanan-layanan (*services*) yang dimilikinya sedemikian rupa sehingga kelak dapat diakses dengan mudah oleh klien-kliennya [1, 9, 11]. Dalam hal arsitektur sistem, sistem yang menggunakan/memanfaatkan teknologi "Big" Web Service sering dinamakan sebagai SOA (*Service Oriented Architecture*) [11].



Gambar 2. Arsitektur Sistem Informasi Geografis Terintegrasi dengan Basis Data XML

RESTful Web Service *(JAX-RS/Java API for XML RESTful Web Service)* yang diprakarsai oleh Roy Fielding dalam disertasinya di Universitas California (2000) [15], pada dasarnya menggunakan metode operasional yang sederhana, yang dimiliki oleh protokol HTTP, untuk saling berkomunikasi. Pada umumnya API (*Application Programming Interface*) RESTfull Web Service menggunakan metoda-metoda standar HTTP itu sebagai parameter-parameter untuk URL (*Uniform Resource Locator*) yang dikirimkan klien ke server RESTful Web Service [15]. Dalam hal arsitektur sistem, sistem yang menggunakan/memanfaatkan teknologi RESTful Web Service sering dinamakan sebagai ROA (*Resource Oriented Architecture*) [11]. Saat mengimplementasikan RESTful Web Service, saat ini para pengembang pada umumnya mengimplementasikan Project Jersey, sebuah implementasi 'kode terbuka' (*open source*) dari JAX-RS, yang juga termuat dalam spesifikasi pada Java EE 6 (dan yang terbaru saat ini Java EE 7) [12]. Dalam hal ini, JAX-RS mendefinisikan 4 anotasi umum yang dapat secara langsung dipetakan ke operasi-operasi HTTP yang bersifat spesifik, yang memiliki padanannya dengan operasi-operasi CRUD yang terdapat di sistem basis data XML yang digunakan. Operasi-operasi HTTP dan padanannya dengan operasi CRUD itu adalah sebagai berikut [12].

- javax.ws.rs.POST (aksi CREATE)
- javax.ws.rs.GET (aksi READ)

- javax.ws.rs.PUT (aksi UPDATE)
- javax.ws.rs.DELETE (aksi DELETE)

Dalam hal visualisasi di aras aplikasi SIG, perhatikan Gambar 2, kita bisa langsung memvisualisasikan data berformat GML itu, karena saat ini ada beberapa penampil data GML yang mampu melakukannya dengan baik (misalnya *GML Viewer*). Meski demikian, ini bukan satu-satunya alternatif. Alternatif lainnya, kita bisa menampilkan data berformat GML tersebut dengan terlebih dahulu mentransformasinya ke bentuk SVG (*Scalable Vector Graphics*), karena format SVG ini saat ini dikenali oleh berbagai perambah (*browser*) yang terkemuka saat ini, misalnya Internet Explorer, Opera, dan Mozzila Firefox. Meski demikian, jika alternatif terakhir ini yang pengembang aplikasi SIG ambil, ada pekerjaan tambahan yang perlu dilakukan, yaitu mentransformasi data berformat GML yang dihasilkan dari lapisan Web Service menjadi data berformat SVG, dimana secara teknis hal ini dapat dilakukan dengan bantuan prosesor XSLT (*eXtensible Stylesheet Language Transformation*). Dalam hal ini, API XSLT ini sudah terintegrasi dengan Java EE 6/7, dan pengembang aplikasi SIG bisa memanfaatkannya, setelah mereka berhasil mengembangkan *XSLT-Stylesheet*-nya [8].

Aplikasi SIG Terintegrasi yang Dikembangkan Menggunakan "Big" Web Service

Saat pengembang aplikasi SIG akan menggunakan/memanfaatkan layanan (*service*) "Big" Web Service, karena "Big" Web Service memiliki standar pesan SOAP (*Simple Object Access Protocol*), maka data berformat GML yang ada di basis data XML terlebih dahulu harus diubah dulu ke format SOAP, yang memiliki elemen XML peringkat teratas (`envelope`) dan yang memuat di dalamnya elemen-elemen `header` dan `body` [12]. Program Java yang pengembang aplikasi SIG buat harus mampu menyusun/merakit pesan SOAP yang berisi data berformat GML, dimana dalam hal ini XML Schema bisa digunakan untuk mendeskripsikan struktur internal pesan SOAP itu. Untuk melakukan hal ini, pada aras rendah, pengembang aplikasi SIG bisa melakukannya dengan membuat suatu lapisan program kecil (adapter) yang memanfaatkan API SAAJ (*Simple Object Access Protocol with Attachments API for Java*), yang memiliki antarmuka DOM (*Document Object Model*) `Node`, yang merupakan kelas dasar dari semua kelas dan antarmuka (*interface*) yang ada di dalam pesan-pesan SOAP [8]. Menggunakan API SAAJ, seperti sesuai dengan peringkatnya yaitu berada di aras rendah, kita bisa 'merakit' pesan SOAP dari data GML yang berasal dari sistem basis data XML yang kita gunakan [8]. Alternatif lainnya, kita bisa menggunakan XSLT (*eXtensible Stylesheet Language Transformation*) [3], yang merupakan suatu API dalam bahasa pemrograman Java yang dapat melakukan transformasi dokumen GML menjadi dokumen SOAP, asalkan kita menyediakan/memliki *XSLT-Stylesheet*-nya [8]. Pada lapisan Web Service, pesan-pesan SOAP yang berasal dari berbagai sistem basis data XML dapat diintegrasikan dan kemudian diuraikan kembali menjadi data berformat GML menggunakan beberapa *parser* bahasa Java yang tersedia saat ini, misalnya JAXP (*Java API for XML Processing*), SAX (*Simple API for XML Processing*), JAXP (*Java API for XML Processing Transformation*), JAXB (*Java API for XML Binding*), DOM (*Document Object Model*), dan sebagainya, untuk keperluan visualisasi [8]. Data GML yang dihasilkan melalui proses integrasi itu selanjutnya dapat divisualisasikan dalam bentuk aplikasi SIG terintegrasi [19].

Dalam hal ini, tentu ada suatu permasalahan tambahan, yaitu bagaimana caranya pesan SOAP yang berasal dari berbagai sistem basis data XML itu dapat diintegrasikan dan kemudian dapat dikenali oleh aplikasi? Hal ini dapat dilakukan menggunakan berkas XML lain yang sering dinamakan sebagai *Web Service Description Language* (WSDL) [8]. WSDL mendefinisikan antarmuka layanan (*service interface*) secara sintak yang bersifat abstraktif. *Port-type* yang ada dalam WSDL memuat operasi-operasi abtrak, yang berhubungan dengan pesan-pesan yang masuk dan yang keluar. *Binding* yang ada pada WSDL selanjutnya akan mengaitkan sejumlah operasi abstrak yang ada dalam berkas WSDL dengan protokol transpor dan format serialisasi yang sesungguhnya. Dalam hal ini, kita umumnya menganggap *binding* dilakukan di atas protokol HTTP (*SOAP over HTTP*), tetapi jika kita mau, sesungguhnya bukan hal yang tidak mungkin saat kita menggunakan protokol-protokol lain selain HTTP.

Aplikasi SIG Terintegrasi yang Dikembangkan Menggunakan RESTful Web Service

REST (*REpresentational State Transfer*) sejak awal memang memang diperkenalkan sebagai arsitektur sistem untuk mengembangkan sistem/aplikasi *hypermedia* yang berskala besar, yang digunakan terutama berbasis pada protokol HTTP [1, 9, 14]. Arsitektur REST (POX-HTTP/*Plain Old XML over HTTP*) sesungguhnya berbasis pada 4 hal yang bersifat mendasar, yaitu [1, 9, 14]:

1. Identifikasi sumberdaya menggunakan URL (*Uniform Resource Locator*).
2. Antarmuka yang seragam, yaitu `PUT`, `GET`, `POST`, dan `DELETE`.

3. Pesan-pesan yang bersifat deskriptif. Sumber-sumber daya yang diperlukan oleh aplikasi SIG dipisahkan dari bagaimana caranya kelak mereka akan dipresentasikan sehingga isinya mungkin berupa data yang memiliki format beragam (misalnya HTML, XML, GML, PDF, JPEG, TIFF, dan sebagainya).
4. Interaksi di dalam aplikasi SIG dilakukan melalui *hyperlink*.

Dengan kata lain, saat kita melakukan integrasi aplikasi SIG atas berbagai sistem basisdata XML menggunakan RESTful Web Service, kita sesungguhnya bisa langsung melakukan pemetaan operasi-operasi CRUD XQuery untuk sistem basisdata XML itu ke operasi-operasi CRUD yang berpadanan pada RESTful Web Service. Selain itu, untuk melakukan integrasi atas data GML yang berasal dari beberapa sistem basis data XML yang ada, kita bisa melakukannya dengan menggunakan URL untuk masing-masing sistem basis data XML itu. Setelah data GML itu diintegrasikan, selanjutnya pengembang aplikasi SIG bisa menvisualisasikannya ke dalam bentuk aplikasi SIG terintegrasi [19].

Jenis Web Service	*Kelebihan*	*Kekurangan*
"Big" Web Service	1. "Big" Web Service yang menggunakan standar pesan SOAP dan standar antarmuka layanan WSDL relatif rumit dalam hal pemrogramannya. Meski demikian, keduanya saat ini telah diterima secara luas di seluruh dunia. 2. Pesan SOAP tidak melulu dapat menggunakan protokol HTTP, tetapi juga dapat dikirimkan menggunakan berbagai sistem *middleware* yang ada saat ini. 3. Penggunaan WSDL untuk mendeskripsikan antarmuka pesan menyembunyikan rincian protokol komunikasi, rincian serialisasi, serta *platform* implementasi layanan (sistem operasi dan bahasa pemrograman) yang ada di bawahnya, sehingga klien Web Service tidak perlu mengetahui bagaimana sesungguhnya layanan yang diperlukannya diimplementasikan. 4. Teknologi SOAP dan WSDL telah matang sehingga saat ini banyak *tool* yang dapat digunakan untuk mengotomatisasi pembentukannya. Beberapa IDE (*Integrated Development Environment*) bahasa Java yang ada saat ini, misalnya Netbeans, Oracle JDeveloper, atau Eclipse bisa menyembunyikan rincian pembentukan SOAP dan WSDL dan sekaligus mengotomatisasi pembentukannya.	1. Permasalahan yang berkaitan dengan interoperabilitas dapat terjadi dalam kaitannya dengan adanya berbagai tipe data yang ada di sisi server layanan dan penggunaannya di sisi klien. 2. Konversi/transformasi antarberbagai bentuk berkas XML, jika dilakukan dengan sangat intensif, dapat menimbulkan permasalahan-permasalahan yang berkaitan dengan kinerja sistem. 3. XML Schema adalah berkas yang sangat kompleks sehingga sukar bagi pengembang untuk mengidentifikasi konstruksi yang mendukung penuh implementasi SOAP dan WSDL.
RESTful Web Service	1. Implementasi RESTful Web Service relatif sederhana dalam hal pemrogramannya karena menggunakan standar-standar yang telah diterima secara luas (HTTP, XML, dan URL). 2. Server dan klien HTTP dikenali oleh sebagian besar bahasa pemrograman dan hampir semua *platform* perangkat keras/perangkat lunak yang saat ini populer. 3. Pengembangan RESTful Web Service cukup sederhana dan mirip dengan pengembangan aplikasi Web pada umumnya. 4. Perbaikan/peningkatan kinerja sistem/aplikasi Web Service dimungkinkan dengan adanya mekanisme *caching* dalam protokol HTTP. 5. Navigasi di dalam aplikasi RESTful Web Service mudah dilakukan dengan mengikuti *hyperlink*.	1. Struktur data yang sangat kompleks sukar diadaptasi ke dalam URL. 2. Implementasi dan kinerjanya sangat bergantung pada kapasitas jaringan yang digunakan.

Gambar 3. Perbandingan Implementasi "Big" Web Service dengan RESTful Web Service

PERBANDINGAN ANTARA "BIG" WEB SERVICE DENGAN RESTFUL WEB SERVICE UNTUK INTEGRASI DATA BERFORMAT GML

Kita sudah singgung di bagian sebelumnya bahwa ada karateristik dasar yang membedakan implementasi Java Web Service menggunakan "Big" Web Service dengan implementasinya menggunakan

RESTful Web Service [1, 2, 3, 4, 6, 8, 9, 14, 16, 19] saat pengembang aplikasi akan bekerja untuk suatu aplikasi SIG terintegrasi. Perbedaan karakteristik ini tentunya akan menimbulkan implikasi yang berkaitan dengan kelebihan serta kelemahannya masing-masing.

Kelebihan dan kelemahan masing-masing implementasi Java Web Service dapat kita lihat pada Gambar 3. Kami tentunya tidak bisa merekomendasikan implementasi Java Web Service mana yang lebih baik, karena hal-hal itu sangat bergantung pada preferensi pengembang aplikasi SIG serta sangat bergantung pada sumberdaya manusia (analis sistem dan pemrogram) yang diperlukan untuk mengembangkan aplikasi SIG terintegrasi yang memanfaatkan sistem basis data XML. Kami hanya berusaha memberikan pandangan-pandangan secara teknis, selanjutnya pengembang aplikasi SIG bisa memilih implementasi Java Web Service mana yang lebih sesuai.

KESIMPULAN

Tidak ada satu pun teknologi yang bersifat sempurna, yang selalu baik untuk setiap kasus yang ada. Demikian juga dengan implementasi Java Web Service untuk mengintegrasikan data berformat GML yang tersimpan di berbagai sistem basis data XML. Baik "Big" Web Service maupun RESTful Web Service memiliki kelebihan dan kelemahannya masing-masing, sehingga pemilihannya sangat bergantung pada preferensi pengembang aplikasi SIG. Meskipun demikian, ada hal yang bersifat mendasar di antara keduanya. Saat pengembang aplikasi SIG menggunakan sistem basisdata XML untuk aplikasi SIG-nya, mereka pada dasarnya harus memetakan operasi CRUD yang difasilitasi menggunakan XQuery terhadap operasi-operasi yang serupa di tingkat lapisan pengintegrasi (lapisan Web Service). Integrasi itu sendiri, saat pengembang aplikasi SIG memilih menggunakan "Big" Web Service, bisa dilakukan dengan cara memanfaatkan standar pesan SOAP serta memanfaatkan standar antarmuka layanan WSDL. Jika pengembang aplikasi SIG memilih menggunakan RESTful Web Service, mereka bisa memanfaatkan URL untuk masing-masing sumber data (sistem basis data XML yang berbeda) dan memetakan operasi-operasi CRUD XQuery pada sistem basis data XML dengan operasi-operasi padanannya di tingkat RESTful Web Service. Dalam aplikasi SIG yang berukuran sangat besar, sesuai dengan karakteristiknya masing-masing, baik "Big" Web Service maupun RESTful Web Service mungkin saja hadir bersamaan untuk membentuk aplikasi SIG terintegrasi yang utuh.

DAFTAR PUSTAKA

1. Adamczyk, Adam, Patrick H. Smith, Ralph E. Johnson, Munawar Hafiz. *REST and Web Services: In Theory and In Practice*. https://netfiles.uiuc.edu/mhafiz/www/research/soaprest/ASJH10-REST.pdf. Diakses 15 Agustus 2011.
2. Amirian, Pouria, Ali A. Alesheikh, 2008. *Publishing Geospatial Data through Geospatial Web Services and XML Database System*. American Journal of Applied Science 5 (10): 1358-1368. ISSN 1546:9239.
3. Amirian, Pouria, Ali A. Alesheikh, 2008. *A Hybrid Architecture for Implementing Efficient Geospatial Web Services: Integrating .NET Remoting and Web Services Technologies*. American Journal of Applied Science 5 (10): 730-742. ISSN 1812:5654.
4. Amirian, Pouria, Ali A. Alesheikh, 2008. *Implementation of a Geospatial Web service Using Web Services Technologies and Native XML Databases***.** Middle-East Journal of Scientific Research 3 (1): 36-48, 2008 ISSN 1990-9233m © IDOSI Publications.
5. Brian, Danny, 2006. *The Definitive to Berkeley DB XML*. Springer-Verlag, New York, USA.
6. Choimeun, C., N. Phemujaya, S. Pomnackchim, C. Chantrapornchai, 2011. *Using GIS Tool for Presenting Spatial Data: Case Study Phatom Province*. International Journal of u- and e-Service, Science and Technology Vol. 4, No. 2, June, 2011.
7. Cox, Simon, Paul Daisey, Ron Lake, Clemens Portele, Arliss Whiteside, 2005. *OpenGIS Geography Markup Language (GML) Encoding Spesification*. Open Geospatial Consortium.
8. Hewitt, Eben, 2009. *Java SOA Cookbook*. O'Reily Media, Inc., Sebastopol, USA.
9. Jucyte, Kristina, Karolis Kevelaitis, Sung Won Park, 2006. *Web Service Implementations with SOAP and REST*. RUC Datalogi, Module 2.
10. Lu, Chang-Tien, Raimando do Santos Jr., Laksmi N. Sripada, Yufeng Kuo, 2007. *Advances GML for Geospatial Applications*. Geoinformatica (2007)

11:131–157 DOI 10.1007/s10707-006-0013-9. Published online: 18 January 2007 # Springer Science + Business Media.

11. Mazzeti, P., S. Nativi, J. Caroon, 2009. *RESTful Implementation of Geospatial Services for Earth and Space Science Applications*. International Journal of Digital Earth, Vol. 2, Supplement 1, 2009, 40-61.
12. Nugroho, Adi. 2010. *Implementasi Java Web Service Menggunakan "Big" Web Service dan REST (REpresentational State Transfer) : Sebuah Studi Perbandingan*. Dipresentasikan di KNSI (Konferensi Nasional Sistem Informasi) di STMIK Potensi Utama, Medan, Indonesia.
13. Nugroho, Adi, Sri Hartati, 2011. *Pengukuran Kinerja Beberapa Sistem Basis Data Relasional Dengan Kemampuan Menyimpan Data Berformat GML (Geography Markup Language) Yang Dapat Digunakan Untuk Mendasari Aplikasi-aplikasi Sistem Informasi Geografis*. Jurnal Teknik Informatika, Universitas Kristen Petra, Surabaya, Indonesia.
14. Pautasso, Cesare, Olaf Zimmerman, Frank Leyman, 2008. *RESTful Web Services v.s. Big Web Service: Making The Right Architectural Decision*. WWW 2008/Refereed Track: Web Engineering - Web Service Deployment, Beijing, China.
15. Richardson, Leonard, Sam Ruby, 2007. *RESTful Web Service*. O'Reily Media, Inc., Sebastopol, USA.
16. Deskripsi Data Services. http://www.xml.com/pub/a/2007/10/25/data-sources-as-web-services.html. Diakses 17 Agustus 2011.
17. Klasifikasi dan spesifikasi XQuery. http://www.xquery.com/standards/. Diakses 16 Agustus 2011.
18. Sumber data XML (*XML Data Source*). http://blog.infotoad.com/category/XML-and-Web-Service-Data-Sources.aspx. Diakses 14 Agustus 2011.
19. _____, 2007. *Geoservices Service Oriented Architecture*. ESRI, New York, USA.
20. _____. *Native Oracle XML DB Web Services in Oracle 11g Release 1*. http://www.oracle-base.com/articles/11g/NativeOracleXmlDbWebServices_11gR1.php. Diakses 17 Agustus 2011.