Pengembangan Aplikasi Semantic Web Untuk Membangun Web yang Lebih Cerdas

Niko Ibrahim
Jurusan Sistem Informasi
Fakultas Teknologi Informasi, Universitas Kristen Maranatha
Jl. Prof. Drg. Suria Sumantri No. 65 Bandung 40164
Email: niko.ibrahim@eng.maranatha.edu

Abstract

The Web, previously only for a repository for text and images, is evolving into a provider of services. Nowadays, the Web acts as information-providing services, such as search engines, flight-booking programs, a variety of e-commerce and B2B applications. However, the Web still has a problem because it is designed primarily for human interpretation and use. It presents large volumes of information in a format tailored for viewing by people so that machines cannot understand the content of that information and manipulate it meaningfully. This paper discusses Semantic Web envisioned by Tim-Berneers Lee that can be an answer to the problem. Moreover, it discusses the technology behind Semantic Web such as XML, RDF, OWL. In the last section, we also discuss a Java framework for developing Semantic Web applications called Jena.

***Keywords**: Semantic Web, RDF, OWL, SPARQL, Jena*

1. Pendahuluan

Website merupakan suatu kebutuhan bagi masyarakat modern sekarang ini, baik itu digunakan untuk melakukan transaksi, penyebaran informasi, maupun pencarian informasi. Website yang memiliki mesin pencari informasi seperti *google* atau *yahoo* kini telah menjadi alternatif utama bagi masyarakat modern dalam mencari berita atau informasi.

Namun demikian, walaupun mesin-mesin pencari ini sanggup memberikan berbagai informasi yang dibutuhkan, seringkali ketepatan dalam mencari informasi tersebut dipertanyakan. Sebagai contoh, saat kita mencari informasi mengenai "Bangau", mesin pencari akan menampilkan berbagai informasi mengenai "Bangau", tanpa mempedulikan apakah yang dicari adalah nama burung, nama sekolah, atau bahkan merek suatu produk. Hal ini sebenarnya merupakan fenomena yang wajar mengingat teknologi informasi di dunia Internet yang belum memiliki mekanisme pengorganisasian data secara terstruktur.

Kejadian di atas sama halnya dengan keadaan dimana belum ditemukannya metode pengorganisasian buku-buku di perpustakaan. Sebelum adanya metode pengorganisasian buku yang salah satunya kita kenal dengan nama "*Dewey Decimal System*", proses pencarian buku di sebuah perpustakaan akan sangat

memakan banyak waktu sekaligus melelahkan. Demikian halnya dengan dunia Internet sekarang ini di mana pencarian informasi seringkali memakan begitu banyak waktu dan sangat melelahkan.

Untuk itulah, para ahli dan peneliti Internet bersepakat untuk mengatasi permasalahan ini. Internet membutuhkan suatu mekanisme yang memampukan komputer mengerti arti kata yang kita cari. Dengan kata lain, kita membutuhkan suatu cara agar kata-kata yang tertera di dalam suatu dokumen Web dapat dibaca dan dimengerti oleh mesin (*machine-readable data)*. *Website* yang memiliki kemampuan seperti ini seolah-olah memiliki kecerdasan buatan yang sanggup memberikan jawaban yang tepat terhadap pertanyaan atau kebutuhan para penggunanya.

Para peniliti setuju bahwa *Semantic Web* merupakan suatu cara untuk melakukan revolusi di dunia Internet yang akan menyatukan interaktifitas pengguna, kolaborasi informasi, dan kecerdasan buatan pada sebuah *Website* [Joh07]. Sebelum membahas lebih jauh mengenai *Semantic Web*, kita akan menelaah secara singkat mengenai berbagai perkembangan yang telah terjadi di dunia Internet sampai dengan saat ini.

2. Klasifikasi Teknologi Web

Teknologi Web sedemikian berkembangnya sehingga para ahli telah memberikan penomoran untuk mengklasifikasikan generasi teknologi Web yang digunakan.

2.1 Web 1.0

Web 1.0 merupakan teknologi Web generasi pertama yang merupakan revolusi baru di dunia Internet karena telah mengubah cara kerja dunia industri dan media. Pada dasarnya, *Website* yang dibangun pada generasi pertama ini secara umum dikembangkan untuk pengaksesan informasi dan memiliki sifat yang sedikit interaktif. Berbagai *Website* seperti situs berita "cnn.com" atau situs belanja "Bhinneka.com" dapat dikategorikan ke dalam jenis ini.

2.2 Web 2.0

Istilah Web 2.0 pertama kalinya diperkenalkan oleh O'Reilly Media pada tahun 2004 sebagai teknologi Web generasi kedua yang mengedepankan kolaborasi dan *sharing* informasi secara *online*.

Menurut Tim O'Reilly, Web 2.0 dapat didefinisikan sebagai berikut:

"*Web 2.0 adalah revolusi bisnis di industri komputer yang disebabkan oleh penggunaan internet sebagai platform, dan merupakan suatu percobaan untuk memahami berbagai aturan untuk mencapai keberhasilan pada platform baru tersebut. Salah satu aturan terutama adalah: Membangun aplikasi yang mengeksploitasi efek jaringan untuk mendapatkan lebih banyak lagi pengguna aplikasi tersebut*"

Berbagai layanan berbasis web seperti jejaring sosial, *wiki* dan *folksonomies* (misalnya: "flickr.com", "del.icio.us") merupakan teknologi Web 2.0 yang menambah interaktifitas di antara para pengguna Web.

Pada umumnya, *Website* yang dibangun dengan menggunakan teknologi Web 2.0 memiliki fitur-fitur sebagai berikut:

- CSS (Cascading Style Sheets)
- Aplikasi Rich Internet atau berbasis Ajax
- Markup XHTML
- Sindikasi dan agregasi data menggunakan RSS/Atom
- URL yang valid
- Folksonomies
- Aplikasi wiki pada sebagian atau seluruh *Website*
- XML Web-Service API

2.3 Web 3.0 / *Semantic Web*

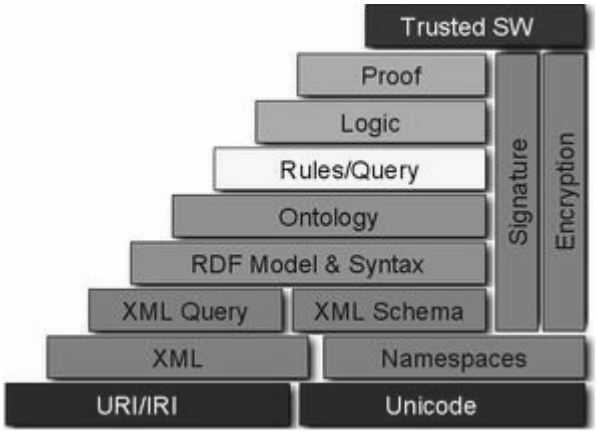
Walaupun masih dalam perdebatan di kalangan analis dan peneliti, istilah Web 3.0 tetap berpotensi menjadi generasi teknologi di dunia Internet. Saat ini, definisi untuk Web 3.0 sangat beragam mulai dari pengaksesan *broadband* secara *mobile* sampai kepada layanan Web berisikan perangkat lunak bersifat *on-demand* [Joh07]. Namun, menurut John Markoff, Web 3.0 adalah sekumpulan teknologi yang menawarkan cara baru yang efisien dalam membantu komputer mengorganisasi dan menarik kesimpulan dari data *online*.

Berdasarkan definisi yang dikemukakan tersebut, maka pada dasarnya *Semantic Web* memiliki tujuan yang sama karena *Semantic Web* memiliki isi Web yang tidak dapat hanya diekpresikan di dalam bahasa alami yang dimengerti manusia, tetapi juga di dalam bentuk yang dapat dimengerti, diinterpretasi dan digunakan oleh perangkat lunak (*software agents)*. Melalui *Semantic Web* inilah, berbagai perangkat lunak akan mampu mencari, membagi, dan mengintegrasikan informasi dengan cara yang lebih mudah [Tim01].

3. Komponen-komponen *Semantic Web*

Pembuatan *Semantic Web* dimungkinkan dengan adanya sekumpulan standar yang dikoordinasi oleh *World Wide Web Consortium* (W3C). Standar yang paling penting dalam membangun *Semantic Web* adalah XML, XML Schema, RDF, OWL, dan SPARQL.

Berikut ini adalah *layer* dari *Semantic Web* sebagaimana direkomendasikan oleh W3C (www.w3c.org):



Gambar 1: *Layer* pada *Semantic Web*

3.1 XML dan XML Schema

Extensible Markup Language (XML) merupakan bahasa *markup* yang didesain untuk menjadi sarana yang mudah dalam mengirimkan dokumen melalui Web. Berbeda dengan *Hypertext Markup Language* (HTML), XML memungkinkan penggunanya untuk mendefinisikan *custom tag*. Namun, standard XML tidak memiliki *constraint* semantik pada arti dari dokumen tersebut.

XML Schema merupakan bahasa yang digunakan untuk mendefinisikan sekumpulan aturan (*schema*) yang harus dipatuhi oleh dokumen XML. Struktur dari dokumen XML yang dibuat harus sesuai dengan *schema* yang telah didefinisikan tersebut.

Berikut ini adalah contoh sederhana definisi *schema* yang dibuat untuk mendeskripsikan sebuah kota dengan menggunakan XML Schema (kota.xsd):

```
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="kota" type="Kota"/>
 <xs:complexType name="Kota">
  <xs:sequence>
   <xs:element name="nama" type="xs:string"/>
   <xs:element name="populasi" type="xs:decimal"/>
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

Berdasarkan *schema* di atas, kita dapat membuat sebuah dokumen XML sebagai berikut:

```
<kota
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="kota.xsd">
```

```
  <nama>Bandung</nama>
  <populasi>3.5</populasi>
</kota>
```

3.2 RDF dan RDF Schema

Resource Description Framework (RDF) adalah spesifikasi yang dibuat oleh W3C sebagai metode umum untuk memodelkan informasi dengan menggunakan sekumpulan format sintaks. Ide dasar dari RDF adalah bagaimana kita dapat membuat pernyataan mengenai sebuah resource Web dalam bentuk ekpresi "Subjet-Predikat-Objek". Dalam terminology RDF, SPO ini seringkali disebut dengan istilah *N-triple*.

Subjek mengacu pada resource yang ingin dideksripsikan. Predikat menggambarkan kelakuan atau karakteristik dari resource tersebut dan mengekspresikan hubungan antara subjek dan objek.

Sebagai contoh, kita ingin mepresentasikan ide "Dokumen ini berjudul Jadwal Ujian dan dipublikasi oleh Fakultas IT UKM". Dengan menggunakan bentuk *N-triple* dari RDF, pernyataan kalimat tersebut dapat memiliki bentuk sebagai berikut:

```
<http://www.itmaranatha.org/jadwal>
<http://purl.org/dc/elements/1.1/title> "Jadwal Ujian"
<http://www.itmaranatha.org/jadwal>
<http://purl.org/dc/elements/1.1/publisher> "Fakultas IT UKM"
```

Selain menggunakan bentuk S-P-O (*N-triple)*, pernyataan tersebut dapat diekpresikan dengan menggunakan RDF/XML sebagai berikut:

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:dc="http://purl.org/dc/elements/1.1/">
   <rdf:Description
rdf:about="http://www.itmaranatha.org/jadwal">
        <dc:title>Jadwal Ujian</dc:title>
        <dc:publisher>Fakultas IT UKM</dc:publisher>
   </rdf:Description>
</rdf:RDF>
```

Mekanisme pendeskripsian *resource* inilah yang merupakan komponen utama yang dikemukakan oleh *W3C's Semantic Web* di mana perangkat lunak dapat menyimpan, menukar, dan menggunakan informasi yang dapat dibaca mesin yang didistribusikan melalui Web, yang pada akhirnya memampukan pengguna dalam menangani informasi tersebut dengan tingkat efisiensi dan tingkat kepastian yang lebih baik.

RDF Schema dapat dipandang sebagai kamus data atau *vocabulary* untuk mendeskripsikan *properties* dan *classes* dari *resources* RDF.

3.3 OWL

Web Ontology Language (OWL) adalah suatu bahasa yang dapat digunakan oleh aplikasi-aplikasi yang bukan sekedar menampilkan informasi tersebut pada manusia, melainkan juga yang perlu memproses isi informasi isi. *Ontology* sendiri dapat didefinisikan sebagai suatu cara untuk mendeskripsikan arti dan relasi dari istilah-istilah. Deskripsi tersebut berisi *classes, properties,* dan *instances.* Deskripsi ini dapat membantu sistem computer dalam menggunakan istilah-istilah tersebut cengan cara yang lebih mudah [Lee06].

Dengan menggunakan OWL, kita dapat menambah *vocabulary* tambahan disamping semantiks formal yang telah dibuat sebelumnya menggunakan XML, RDF, dan RDF *Schema*. Hal ini sangat membantu penginterpretasian mesin yang lebih baik terhadap isi Web. Untuk mendeskripsikan *properties* dan *classes*, OWL menambahkan vocabulary seperti:

- "among others"
- Relasi antar classes (misalnya: "disjointness")
- Kardinalitas (misalnya: "exactly one")
- Kesamaan (equality)
- Karakteristik property (misalnya: "symmetry")
- Enumerated classes

OWL menyediakan tiga buah subbahasa yang dirancang untuk digunakan oleh para pengguna tertentu, yaitu:

- OWL Lite, digunakan oleh pengguna yang membutuhkan suatu hirarki pengklasifikasian dan berbagai *constraints* sederhana.
- OWL DL, digunakan oleh pengguna yang menginginkan tingkat ekpresi maksimal dan semua konklusi yang dihasilkan dapat dihitung dalam waktu yang terbatas (*finite*)
- OWL Full, digunakan oleh pengguna yang menginginkan tingkat ekpresi maksimal dan kebebasan sintaks dari RDF tanpa mempertimbangkan komputasi yang dibutuhkan.

Berikut ini adalah contoh untuk mendeskripsikan *class* Airport menggunakan OWL:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base="http://www.daml.org/2001/10/html/airport-ont">

<owl:Ontology rdf:about="">
  <owl:versionInfo>$Id: airport-ont.daml,v 1.1 2002/03/14
06:24:16 mdean Exp $</owl:versionInfo>
  <rdfs:comment>Airport</rdfs:comment>
```

```
</owl:Ontology>

<rdfs:Class rdf:ID="Airport">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#name"/>
      <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#iataCode"/>
      <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#icaoCode"/>
      <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#location"/>
      <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#latitude"/>
      <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#double"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#longitude"/>
      <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#double"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#elevation"/>
      <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#double"/>
    </owl:Restriction>
  </rdfs:subClassOf>
```

```
</rdfs:Class>

<owl:DatatypeProperty rdf:ID="elevation"/>
<owl:DatatypeProperty rdf:ID="iataCode"/>
<owl:DatatypeProperty rdf:ID="icaoCode"/>
<owl:DatatypeProperty rdf:ID="latitude"/>
<owl:DatatypeProperty rdf:ID="location"/>
<owl:DatatypeProperty rdf:ID="longitude"/>
<owl:DatatypeProperty rdf:ID="name"/>

</rdf:RDF>
```

3.4 SPARQL

SPARQL Protocol and RDF Query Language (SPARQL) adalah sebuah *protocol* dan bahasa *query* untuk *Semantic Web's resources*. Sebuah *query* yang menggunakan SPARQL dapat terdiri atas *triple patterns*, konjungsi (*or*), dan disjungsi (*and*).

Berikut ini adalah contoh *query* yang menghasilkan semua ibu kota di Indonesia:

```
PREFIX abc: <http://mynamespace.com/exampleOntologie#>
SELECT ?capital ?province
WHERE {
  ?x abc:cityname ?capital.
  ?y abc:provincename ?province.
  ?x abc:isCapitalOf ?y.
  ?y abc:isInCountry abc:indonesia.
}
```

Untuk menjalankan SPARQL kita dapat menggunakan beberapa tools dan APIs seperti: ARQ, Rasqal, RDF::Query, twingql, Pellet, dan KAON2 [Lei05].

Tools tersebut memiliki *API* yang memampukan pemrogram untuk memanipulasi hasil query dengan berbagai aplikasi yang ada. Namun, sebagai standar kita dapat menggunakan *SPARQL Query Results XML Format* [Dav07] yang direkomendasikan oleh W3C.

Berikut ini adalah hasil dari *query* di atas:

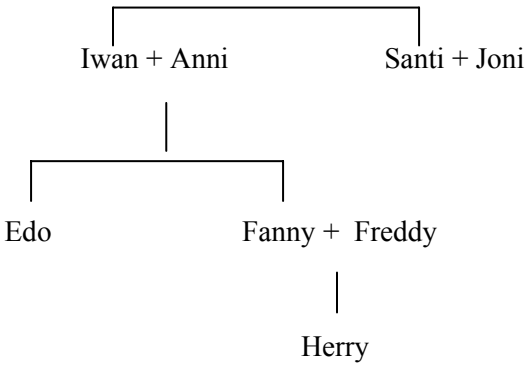
```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">

  <head>
    <variable name="capital"/>
    <variable name="province"/>
  </head>

  <results>

    <result>
      <binding name="capital">
       <literal
datatype="http://www.w3.org/2001/XMLSchema#string">
         Bandung
       </literal>
      </binding>
      <binding name="province">
       <literal
datatype="http://www.w3.org/2001/XMLSchema#string">
         Jawa Barat
       </literal>
      </binding>
    </result>

    <!-- more results -->

  </results>
</sparql>
```

4. Jena: Framework Pengembangan Aplikasi *Semantic Web* Berbasis Java

Jena Java RDF API and toolkit merupakan *framework* berbasis bahasa Java untuk mengkonstruksi aplikasi *Semantic Web*. *Framework* ini menyediakan lingkungan pemrograman untuk RDF, RDF Schema, OWL, dan SPARQL serta memiliki mesin inferensi berbasis aturan (*rule-based inference engine*). Jena juga memiliki kemampuan untuk digunakan sebagai basis data RDF melalui *layer* yang dikenal dengan nama Joseki [Jer07].

Untuk membuat aplikasi *Semantic Web*, pertama-tama kita harus membuat model RDF.

Sebagai contoh, kita membuat model pohon keluarga sebagai berikut:



Gambar 1: Contoh Pohon Keluarga

Untuk contoh ini, misalnya kita telah memiliki *vocabulary "Relationship"* [Ian05] yang didalamnya terdapat *properties siblingOf*, *spouseOf*, *parentOf*, dan *childOf*.

Jena memiliki kelas *ModelFactory* yang dapat digunakan untuk membuat berbagai model. Melalui model inilah kita akan membuat sebuah *Resource* yang merepresentasikan setiap orang yang ada pada pohon keluarga di atas.

Setelah semua *resource* dibuat, selanjutnya kita dapat menambahkan *statements* kepada *resource* tersebut. Pada Jena, subjek setiap *statement* selalu berupa sebuah *Resource*, sedangkan predikat direpresentasikan oleh *Property*, dan objek bisa direpresentasikan oleh sebuah *Resource* lain maupun sebuah nilai literal. Untuk menggambarkan relasi pada pohon keluarga tersebut, kita harus menambahkan empat buah *instance Property* dengan cara memanggil *method addProperty( )*.

Berikut ini adalah potongan kode yang merepresentasikan model pohon keluarga (keluarga.rdf) pada gambar 1:

```
// URI declaration
String familyUri = "http://family/";
String relationshipUri = "http://purl.org/vocab/relationship/";

// Create an empty Model
Model model = ModelFactory.createDefaultModel();

// Create a Resource for each family member, identified by their URI
Resource iwan = model.createResource(familyUri+"iwan");
Resource santi = model.createResource(familyUri+"santi");
Resource joni = model.createResource(familyUri+"joni");
Resource anni = model.createResource(familyUri+"anni");
// and so on for other family members

// Create properties for the different types of relationship
//  to represent
Property childOf = model.createProperty(relationshipUri,"childOf");
Property parentOf = model.createProperty(relationshipUri,"parentOf");
Property siblingOf = model.createProperty(relationshipUri,"siblingOf");
Property spouseOf = model.createProperty(relationshipUri,"spouseOf");

// Add properties to iwan describing relationships to other family
//  members
iwan.addProperty(siblingOf,santi);
iwan.addProperty(spouseOf,anni);
iwan.addProperty(parentOf,edo);

// Can also create statements directly . . .
Statement statement = model.createStatement(iwan,parentOf,fanny);

// but remember to add the created statement to the model
```

Program 1: Pendeklarasian Pohon Keluarga

Dengan menggunakan OWL, kita juga dapat menambahkan karakteristik dari semuar *resources*. Misalnya, OWL dapat digunakan untuk menyatakan bahwa *property* `childOf` adalah kebalikan dari *property* `parentOf.`

Setelah model terbentuk, selanjutnya kita dapat melakukan *query* terhadapnya. Berikut ini contoh sederhana dari penggunaan SPARQL dengan menggunakan *framework* Jena:

```
// Open the "keluarga RDF graph" from the filesystem
InputStream in = new FileInputStream(new File("keluarga.rdf"));

// Create an empty in-memory model and populate it from the graph
Model model = ModelFactory.createMemModelMaker().createModel();
model.read(in,null); // null base URI, since model URIs are absolute
in.close();

// Create a new query
String queryString =
         "PREFIX fam: <http://mynamespace.com/keluarga> " +
         "SELECT ?name " +
         "WHERE {" +
         "      ?name fam:childOf \"Iwan\" . " +
         "      }";

Query query = QueryFactory.create(queryString);

// Execute the query and obtain results
QueryExecution qe = QueryExecutionFactory.create(query, model);
ResultSet results = qe.execSelect();

// Output query results
ResultSetFormatter.out(System.out, results, query);

// Important - free up resources used running the query
qe.close();
```

Program 2: Contoh Pengeksekusian Sebuah Query Sederhana

Selain melakukan *query* sederhana seperti di atas, Jena juga memiliki kemampuan untuk melakukan penalaran (*inference)* terhadap model yang telah dibuat. Dengan kata lain, Jena memiliki kemampuan untuk membuat *statements* tambahan yang belum ditulis secara eksplisit di dalam model [Jer04].

5. Kesimpulan dan Saran

Tulisan ini telah membahas secara singkat mengenai teknologi *Semantic Web* yang menjanjikan di mana Web dapat memiliki kecerdasan alami. Artinya, dengan menggunakan *Semantic Web*, informasi yang tertulis pada sebuah *Website* tidak saja berguna sebagai informasi yang bisa dibaca oleh manusia melainkan juga menjadi sumber informasi yang bisa diproses dan dimengerti oleh komputer.

Tulisan ini juga telah memperlihatkan berbagai komponen yang mendukung pembuatan *Semantic Web* yaitu XML, XML Schema, RDF, RDFS, dan SPARQL. Selain itu, tulisan ini juga membahas secara sederhana mengenai Jena yang dapat digunakan oleh *programmer* berbasis Java untuk membuat aplikasi *Semantic Web*.

Walaupun *Semantic Web* dipercaya sebagai generasi yang akan datang di dunia Web, berbagai kritikan juga mulai muncul. Misalnya, sejauh mana tingkat fisibilitas dari *Semantic Web* khususnya di dunia bisnis. Selain itu, masalah lamanya konsumsi waktu pada saat membuat dan mempublikasi isi Web pun masih menjadi perdebatan karena untuk menampilkan sebuah informasi saja akan diperlukan dua format yang berbeda yaitu untuk dilihat manusia dan sekaligus untuk diproses mesin.

Berbagai teknik lain yang menjadi komplemen dari *Semantic Web* pun sebenarnya telah dilakukan oleh beberapa perusahaan. Perusahaan-perusahaan yang bergerak di data mining masih terus berusaha untuk mencari pola dari aliran informasi yang begitu simpang siur di dalam Web. IBM pun telah menawarkan layanan yang mampu menyisir isi *blogs*, *message boards*, dan *newsgroup* dari para klien mengenai produk mereka yang selanjutnya dapat menarik kesimpulan mengenai tren tanpa menggunakan teknik *metadata* seperti yang dilakukan pada *Semantic Web*.

Sebagai kesimpulan akhir, pada dasarnya tidak ada orang yang dapat memastikan teknik pengorganisasian informasi apa yang paling tepat. Mengingat data digital seluruh dunia yang begitu besar di dalam Web, kita tidak mungkin menampungnya hanya dalam sebuah *framework*. *Semantic Web* menawarkan sebuah solusi yang memang luar biasa bagi pemrosesan informasi di Web. Namun, penelitian dan pengembangan *tools* untuk *Semantic Web* masih harus terus dilakukan agar di kemudian hari berbagai aplikasi *Semantic Web* dapat diimplementasikan dan dipergunakan secara luas.

Referensi

[Dav07] *Dave Beckett, Jeen Broekstra (2007). SPARQL Query Results XML Format. Available: http://www.w3.org/TR/rdf-sparql-XMLres/. Accessed: 20/06/07*

[Ian05] *Ian Davis, Eric Vitiello Jr. (2005). RELATIONSHIP: A vocabulary for describing relationships between people. Available: http://vocab.org/relationship/. Accessed: 20/06/07*

[Jer04] *Jeremy J. Carrol, et al. (2004). Jena: implementing the semantic web recommendations. Proceedings of the 13th International WWW Conference on Alternate Track Papers and Posters, ACM Press, 2004, p 74-83.*

[Jer07] *Jeroen Van Der Ham (2007). Semantic Web Tools. Available: http://esw.w3.org/topic/SemanticWebTools. Accessed: 10/06/07*

[Joh07] *John Borland (2007). A Smarter Web. Available: http://www.technologyreview.com/Infotech/18396/. Accessed: 12/06/07*

[Lee06] *Lee Provoost, Erwan Bornier (2006). Service-Oriented Architecture and the Semantic Web: A killer combination? Published Thesis. University of Utrech*

[Lei05] *Leigh Dodds (2005). Introducing SPARQL: Querying the Semantic Web. Avaiblable: http://www.xml.com/lpt/a/1628. Accessed: 20/06/07*

[Mic04] *Michael K. Smith, Chris Welty, Deborah L. McGuinnes (2004). OWL Web Ontology Language Guide. Available: http://www.w3.org/TR/owl-guide/. Accessed: 12/06/07*

[She01] *Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng (2001) . Semantic Web Services. IEEE Intelligent Systems: Apr 2001. Vol 16 Iss 1, p. 46-53.*

[Tim01] *Tim Berners-Lee, J. Hendler, and O. Lassila (2001). The semantic web. Technical report, Scientific American.*