

Intelligent agents and the Semantic Web

Developing an intelligent Web

Skill Level: Intermediate

[Daniel J. Lewis \(daniel.lewis@bcs.org.uk\)](mailto:daniel.lewis@bcs.org.uk)
Technology Evangelist/Computer Scientist

21 Oct 2008

The Semantic Web envisioned by Berners-Lee, Hendler, and Lassila in 2001 was a grandiose vision that involved the use of agents to book doctor appointments and to find the best driving routes with the least hassle. The envisaged system was built upon formal ontologies that had already achieved a large following of scientists and agent developers. Although they raised some important issues and put forward interesting connections between technologies, they missed one thing: the fact that the Web had turned into a web of documents. Therefore, a middle way needed to occur between the formalism of ontologies and the informalism of documents. This is known as Linked Data. Linked Data coupled with agent technology is an ideal way of dealing with Semantic Web data. This article provides an overview of the Interlinked Semantic Web, agent technologies, and an example of the two combined.

Introduction to the Semantic Web

The Semantic Web is a set of tools and frameworks for making, querying, and manipulating knowledge. The key technology behind the Semantic Web is the Resource Description Framework (RDF), which can be seen as both a graphical (a graph of n-triples) and an object-oriented knowledge representational model. Several formats exist to make this model a machine-readable model (for instance RDF/XML, RDF/N3, and RDF/Turtle). Lots of information is available to new developers of the Semantic Web on the Semantic Web Activity page and the now defunct Semantic Web Education and Outreach page. See the [Resources](#) section for links to these pages.

One of the key design features that makes the Semantic Web more meaningful and

useful is the establishment of interlinking data between data sets across the Web. Tim Berners-Lee highlighted this in his 2006 article titled "Design Issues: Linked Data." Linked Data ensures that all Semantic Web data is accessible on the Web by exploiting the full potential of HyperText Transfer Protocol (HTTP). Lots of exemplary Linked Data sets exist today, as you can see in Richard Cyganiak's Linking Open Data datasets cloud (see [Resources](#)), which is all thanks to community efforts under the Linking Open Data initiative.

Introduction to intelligent agents

Combining descriptions from Jennings (1999) and Russell and Norvig (2003), you can understand that an agent is an encapsulated computer system made up of an architecture and a program. This computer system should:

- Be situated in some environment
- Be able to perceive its environment
- Be capable of autonomous action within that environment
- Have some kind of design objectives

An agent system is made up of four essential parts:

- A performance measure
- An environment
- Actuators
- Sensors

The four basic types of agents

There are four generic types of reactive agents that are traditionally discussed in texts:

- Simple reflex agents, which act based on their current perceptions
- Model-based reflex agents, which act based on their current perceptions and partial histories
- Goal-based agents, which use their current perceptions in addition to their desires (goals) to act
- Utility-based agents, which try to maximize their status to achieve higher efficiency of acting

However, some modern books about artificial intelligence expand their discussions on other types of agents that aren't specifically reactive agents as defined above. These include:

- Interface agents
- Mobile agents
- Information agents
- Learning agents
- Robotic agents

The scheduler agent

All agents have specific behaviors. A scheduler agent, which could be seen as an implementation of a Model-based reactive agent, takes time into consideration when executing those behaviors. There are many different types of scheduler behaviors. A simple example is the waker behavior, which runs after a particular time-out period. Another simple example is a ticker behavior, which executes once every set period.

The searcher agent

A searcher agent is a kind of information agent that has one particular role, which is to find one or more items in a set. It can be formally implemented as a goal-based or utility-based agent. A Web spider, which is just one kind of searcher agent, searches the Web to pre-index pages that are ready for users to search. Searcher agents, which are spawned with specific initial goals from builder agents, are also employed in the PubMed system, a service of the U.S. National Library of Medicine that includes over 18 million citations from MEDLINE and other life science journals for biomedical articles back to the 1950s. Text-based searcher agents usually have some kind of natural language processing built in and, in some cases, latent semantic indexing.

Agent communication

Intelligence is limited if there isn't some kind of communication to share knowledge and experiences. This is also the case with agent technology, as it is a form of artificial intelligence. The Foundation for Intelligent Physical Agents (FIPA) has carefully built standards and recommendations for use in multi-agent systems. This ensures that all agents have some form of lingua franca. FIPA is an Institute of Electrical and Electronics Engineers (IEEE) standards organization.

The Object Management Group (OMG) also has standards that complement the FIPA standards, known as the Mobile Agent Facility. It is also worth noting that agents need to share some particular vocabulary. This is often done by building an ontology using the OWL Web Ontology Language.

Agent libraries for Java programming language

There are a number of common FIPA-based agent libraries for the Java™ programming language, including Java Agent Development Framework (JADE), by Telecom Italia, and Agent Building and Learning Environment (ABLE), by IBM®.

Intelligent agents on the Semantic Web

There has been a reasonably clear, distinguishable divide between the business-oriented Web and academia. Businesses have been advertising on paper for thousands of years. So, when the Web was created, they almost immediately saw it as a chance to market their products and services. The Web soon became a document Web made with design and style in mind.

Academia had something else in mind. Computer science as an academic field has decades of artificial intelligence research and, in particular, knowledge engineering research. Academia spawned both agent technology and the Semantic Web, with the former being proposed as a business-friendly technology and the latter as a scientific technology.

The Semantic Web has clear business benefits, but only with the additional rules which are defined by the Linked Data Design Note by Tim Berners-Lee in 2006. By exposing public business data as Linked Data, businesses not only make their products or services more findable in an objective fashion, but they also open it up for machine reading, which is ideal for the multi-agent systems. Exposing data as Linked Data becomes an exciting marketing technique.

The refocus from semantics through formal ontologies to semantics through interlinking will not stop the grandiose vision of Berners-Lee and others. However, the way that it progresses may not be exactly how they envisaged in 2001. Agents are now able to follow one node to another across the global giant graph thanks to Linked Data. This change led to the Semantic Web being "revisited" in 2006 by Shadbolt and others and "rebooted" in 2008 by Cyganiak.

An example of agents on the Semantic Web

There are plenty of things that you can do with agents and the Semantic Web. For instance, you might want to make one of the following:

- A Web-indexing agent that turns documents into formal Semantic Web-based knowledge
- A personal agent that uses formal Semantic Web knowledge bases to book holidays or even doctor appointments

- A multi-agent system that is capable of acting in its own community to build and maintain additional Linked Data sets

For this article, I created a simple example using the JADE framework and using the DBpedia Linked Data Set. It is a single-agent system that passes two plain-text parameters (an object name and one of its attributes names) on activation and then searches through the DBpedia data set to find a result (the attribute value).

Agent systems are built using a variety of techniques. For this example, I use a simple requirements, analysis, design, and development flow.

Requirements

The requirements statement provides a brief description about what the agent system is supposed to do. In this case, the system should be able to take in a noun and an attribute key of that noun, search the Linked Open Data available on the Web, and provide a value for that attribute. [Listing 1](#) provides an example of possible output from the system.

Listing 1: An example output from the required system

```
Noun input: Bob Marley
Attribute key Date of Birth

Agent is fetching...

Attribute value: 1945-02-06
```

Analysis

For this example, I use the Roles Model method of analysis, which is a model within the Gaia Agent Design Methodology (see [Resources](#)). It describes what a particular agent does, why it does it, and what responsibilities and permissions it has:

Role Schema: SimpleNounAttributeSearcher

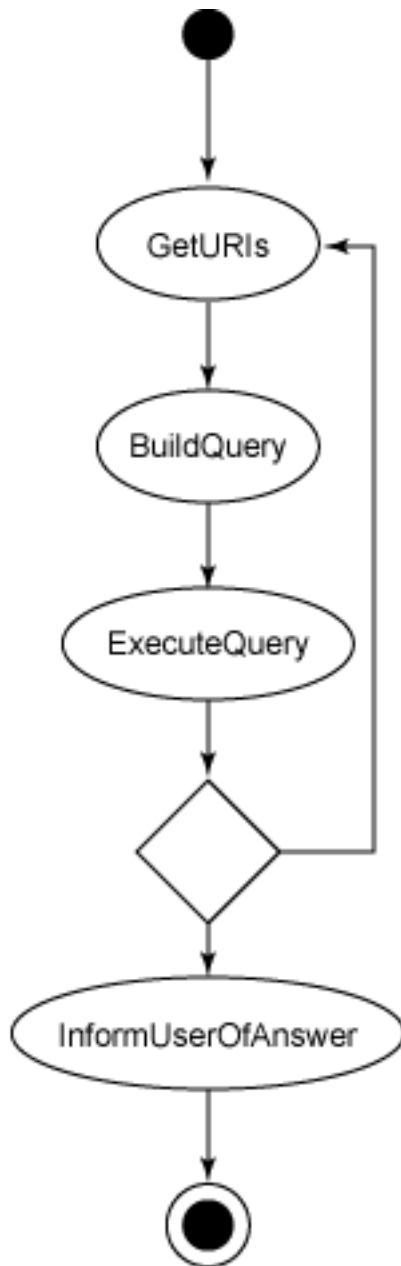
- **Description:** To search the interlinked Semantic Web for an attribute value for a certain object
- **Permissions:** Read and search data from any open Semantic Web data source
- **Responsibilities:**
- **Liveness:**

- SimpleNounAttributeSearcher = (GetURIs. BuildQuery. ExecuteQuery. InformUserOfValue }
- Safety:true

Design

I use a goal flow diagram, shown in [Figure 1](#), to show the flow from one goal to the next to achieve the overall goal of finding an attribute value from an object name and attribute name pair.

Figure 1. Goal flow diagram



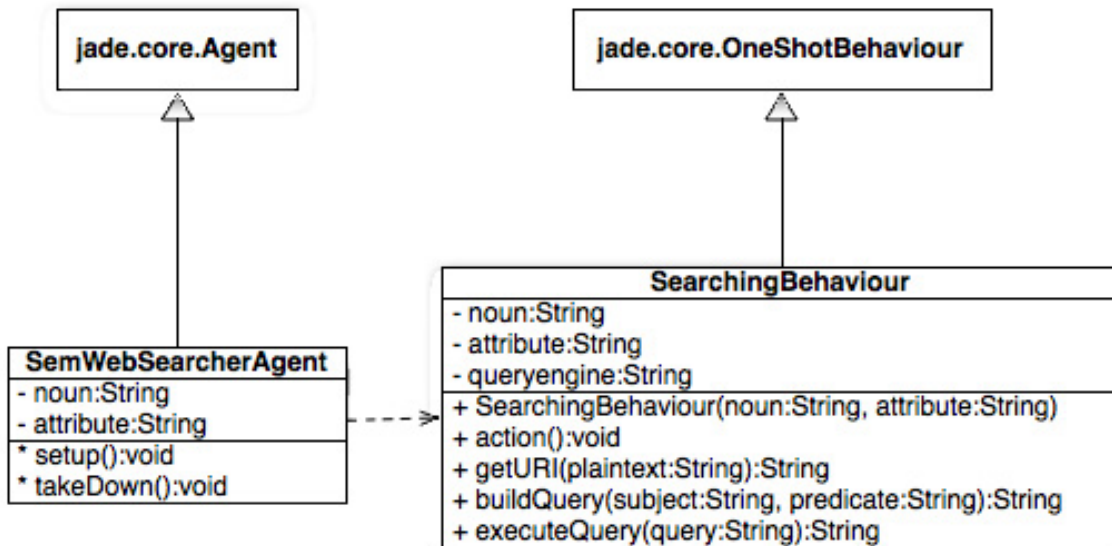
The input from the user is in simple plain text, and Semantic Web systems usually talk in Uniform Resource Identifiers (URIs), and therefore, the `GetURIs` activity fetches a URI based upon the plain text representation (this is thanks to the label relationship found in the RDF Schema). The `BuildQuery` activity takes the URIs and generates a simple query based upon the URIs found in the previous activity. `ExecuteQuery` then executes the query and determines if the result is useful. If it is, the `InformUserOfAnswer` shows the result to the user. If not, the simple agent starts again with different URIs.

The sub-goals described in [Figure 1](#) will be translated into simple public methods in

Java, whereas the overall goal is translated into a JADE Behaviour.

I also use an object-oriented class diagram, shown in [Figure 2](#), to transcribe the goal flow diagram into a form that is ready for development in an object-oriented language such as Java. It refers to `Agent` and `OneShotBehaviour`, which are parts of the JADE library.

Figure 2. Object-oriented class diagram



The main class within the program is the `SemWebSearcherAgent`, which is of type `Agent` in the JADE library. An agent usually has a behaviour. In this case, the `OneShotBehaviour` (which is also part of the JADE library) was chosen and extended to make `SearchingBehaviour`. The `SearchingBehaviour` will do the bulk of the searching work. The code is attached to this article in the [Download table](#) below. Note the `queryengine` property of the `SearchingBehaviour`. This is the URL of the SPARQL-based endpoint in which I will be doing my searches. This example uses the DBpedia endpoint at:

```
http://dbpedia.org/sparql
```

It has the following HTTP parameters:

```
format=text/rdf+n3&query=
```

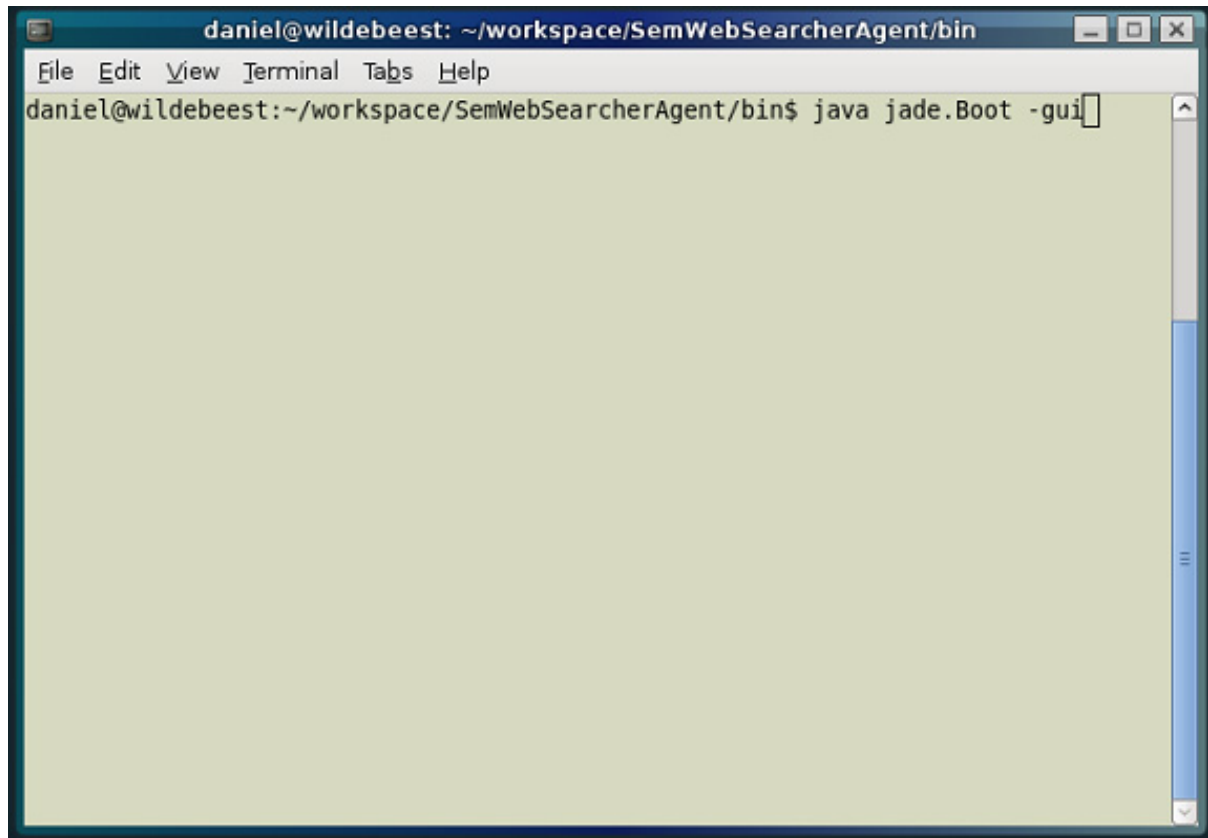
, where `query` is the parameter for sending the SPARQL query. The use of Java methods in the `SearchingBehaviour` should be familiar from the Goal Flow Diagram.

Development flow

An agent lives in a particular environment, and therefore JADE provides one. JADE also provides a graphical user interface (GUI) to initialize, control, and terminate agents. This section goes through the steps to initialize an agent.

First, open the terminal and start the JADE service with the GUI, shown in [Figure 3](#).

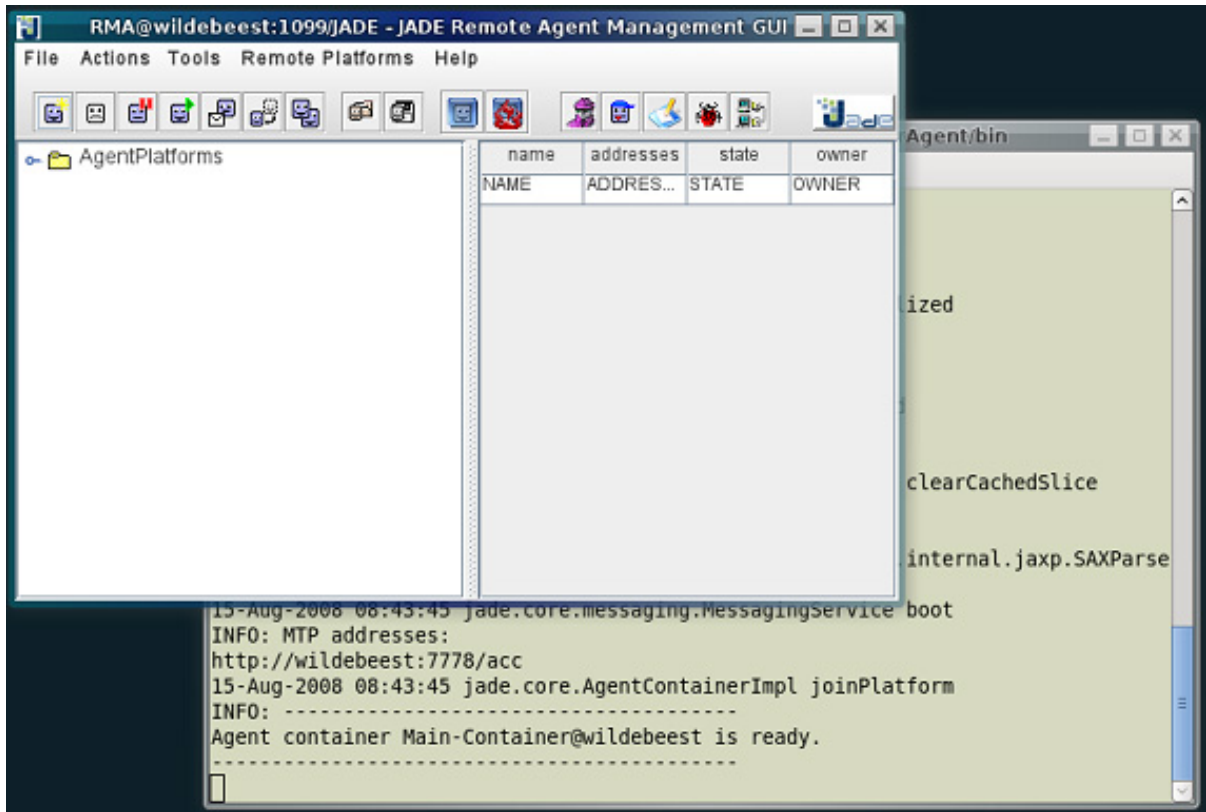
Figure 3. Start JADE from the command line

A screenshot of a terminal window. The title bar reads "daniel@wildebeest: ~/workspace/SemWebSearcherAgent/bin". The terminal has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The command prompt shows "daniel@wildebeest:~/workspace/SemWebSearcherAgent/bin\$ java jade.Boot -gui" with a cursor at the end of the command. The terminal area is mostly empty, indicating the command has been executed and the GUI is running in a separate window.

```
daniel@wildebeest: ~/workspace/SemWebSearcherAgent/bin
File Edit View Terminal Tabs Help
daniel@wildebeest:~/workspace/SemWebSearcherAgent/bin$ java jade.Boot -gui
```

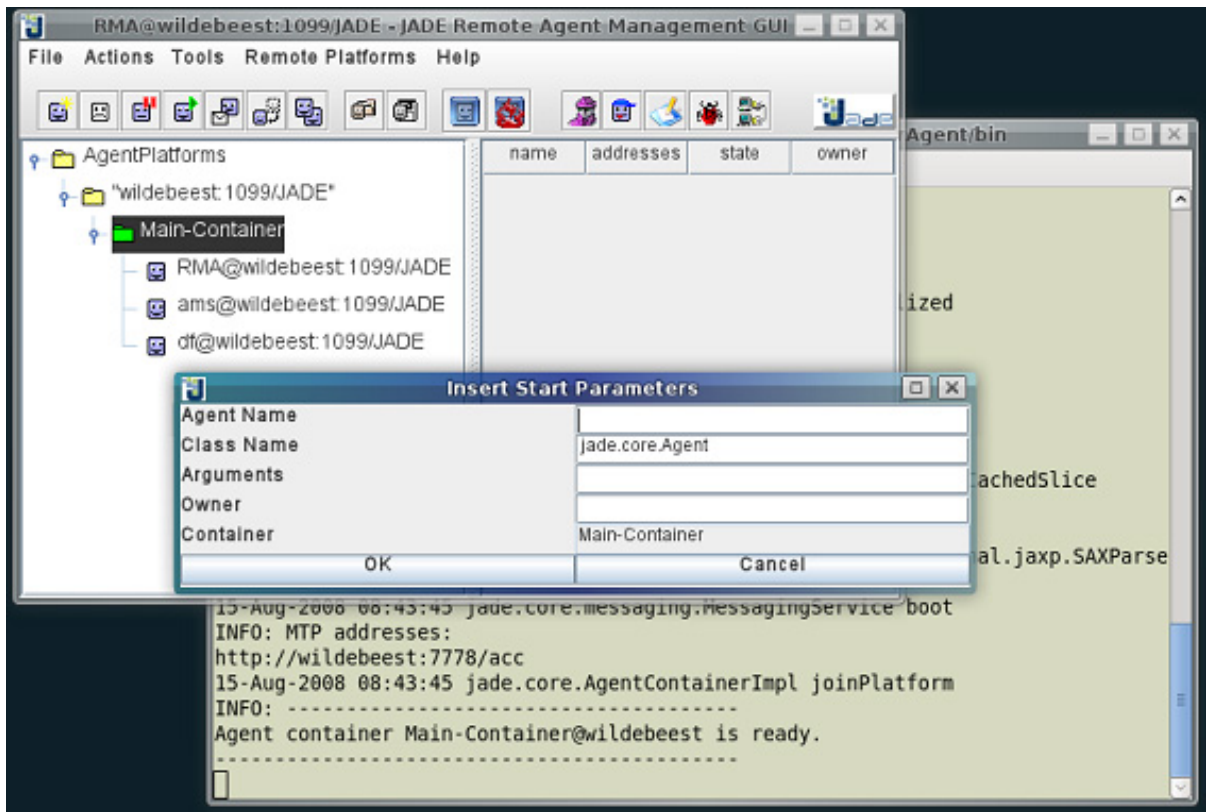
The graphical user interface looks like [Figure 4](#).

Figure 4. The JADE GUI



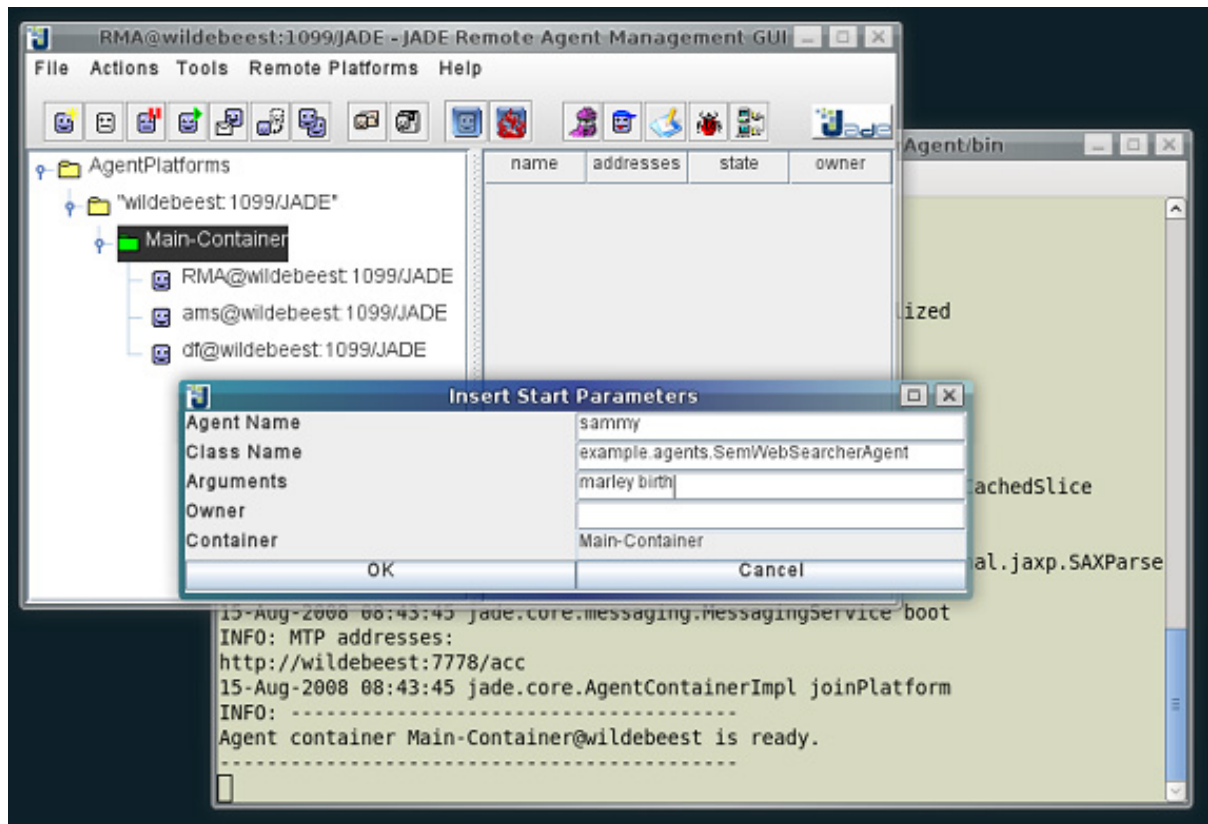
By expanding the AgentPlatforms tree, you can find a container and then start an agent, as shown in [Figure 5](#).

Figure 5. Start an agent in a container



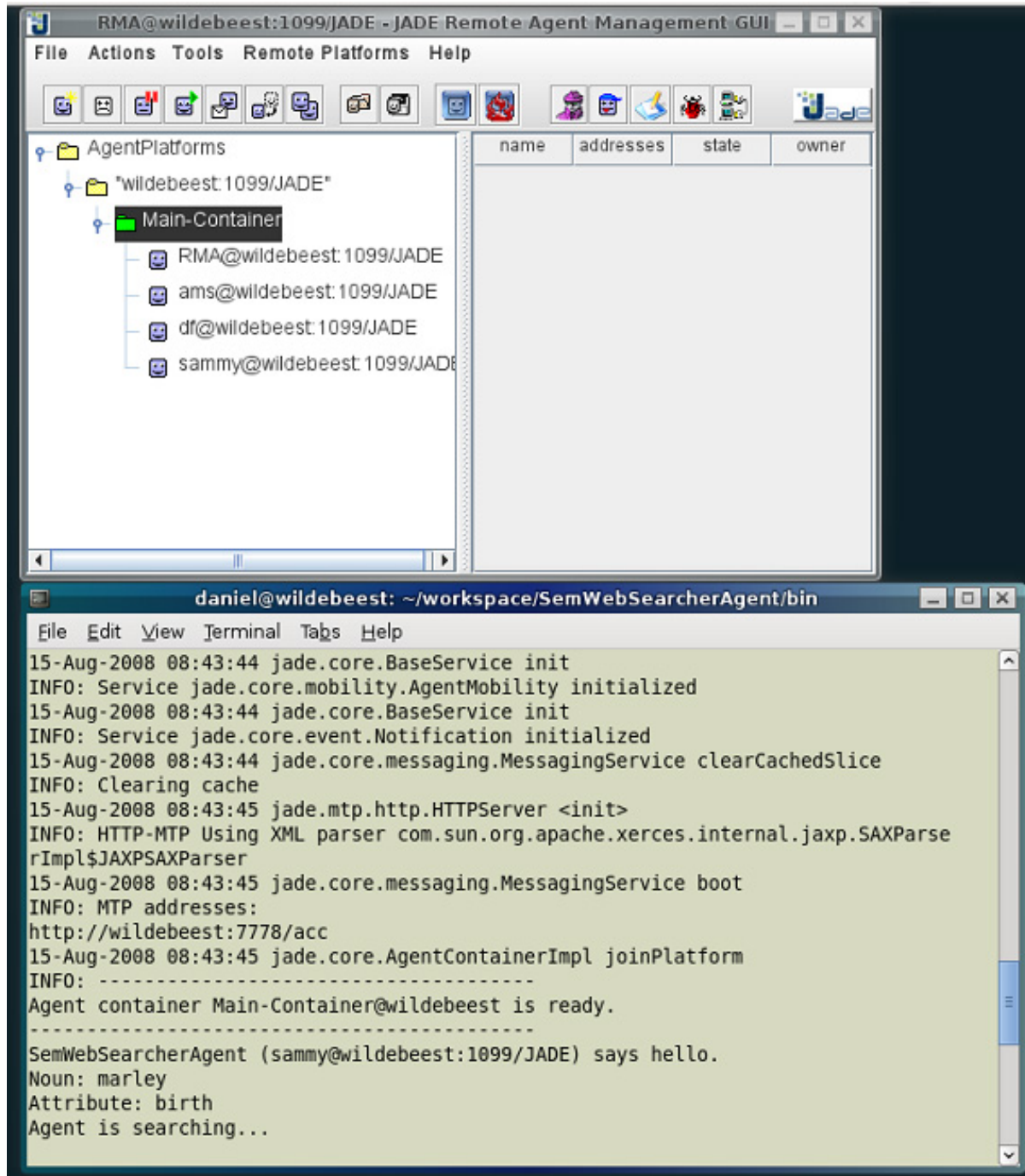
For this example, I start the `example.agents.SemWebSearcherAgent` that I designed and implemented. I need to pass some arguments about what to search for, as shown in [Figure 6](#).

Figure 6. Start the SemWebSearcherAgent in the Main-Container



Here I want to find the object Marley (as in "Bob Marley") and the attribute birth (as in "Date of Birth"). The agent loads, searches, and then prints the result from the global interlinked Semantic Web, as shown in [Figure 7](#).

Figure 7. The SemWebSearcherAgent doing its magic



Conclusion

The Semantic Web in Linked Data format is a perfect way to represent knowledge on the Web because the object-oriented model is simple to understand. Agent technology is the perfect way to model an autonomous process because of its ability to become an artificial society based on real-life society. This article demonstrated

the beginnings of a simple project that could be taken further in several different ways, including:

- Closer modeling of a real-life user agent
- More artificial intelligence
- Extensive use of formal ontologies
- Expanding the Linked Data sets beyond DBpedia
- Making it a multi-agent system
- Translating an entirely free-text question into a SPARQL query

As you can see, the opportunities are endless for both business and academia.

Downloads

Description	Name	Size	Download method
Sample Java (JADE) scripts for this article	SemWebSearchAgent.zip	245	HTTP

[Information about download methods](#)

Resources

Learn

- [FIPA](#) is an IEEE organization that provides standards and recommendations for multi-agent communication.
- The [Semantic Web Activity](#) page of the World Wide Web Consortium (W3C) points to many other resources for the various Semantic Web frameworks, formats, standards, and recommendations.
- The [Semantic Web Education and Outreach \(SWEO\)](#) page of the W3C provides additional resources for learning about the Semantic Web.
- [Mobile Agent Facility, Version 1.0](#) contains a great deal of information about how to design and develop agents for mobile devices.
- The [JADE Programmer's Guide](#) is the starting point for developing software agents using the Java programming language and the JADE library.
- The [JADE Administrator's Guide](#) is the starting point for running and communicating with agents built using the JADE library.
- The W3C's [Linked Data Design Issues](#) document contains the rules and regulations of putting Semantic Web information on the Web in the best possible way—Linked Data.
- "The Semantic Web" (Scientific American, 2001), written by Berners-Lee, Hendler & Lassila, is the infamous article that launched the Semantic Web into the spotlight. It talks about the role of formal ontologies and agent technology in the Web's future.
- [JADE Tutorial: JADE Programming for Beginners](#) is a great resource when you start to use the JADE library.
- [Artificial Intelligence Illuminated](#) (Jones and Bartlett Publishers, Inc., 2004) by Ben Coppin is a great book that covers agent technology in some detail. It's also a good guide for other forms of artificial intelligence that you can include in your multi-agent systems to make them more autonomous.
- Richard Cyganiak's [Linking Open Dataset Cloud](#) is a graphical representation of most of the Linked Data sets and how they are interlinked.
- Richard Cyganiak's "[Searching, publishing and remixing a Web of Semantic Data](#)" is a conference presentation about Linked Data and how it can be reused in different ways.
- [Multiple Approaches to Intelligent Systems](#) (Springer Berlin/Heidelberg, 1999) is an interesting book about how to design, develop, and maintain agent systems.
- "[Intelligent Agents in Medicine](#)" (Lhotská, L. & Prieto, L., 2006) provides an

overview about how intelligent agents can be used in medical and health applications. It also discusses the Agents for Diagnostics and Monitoring (ADAM) and PubMed systems.

- See [PubMed](#) in action and search over 18 million citations from MEDLINE and other life science journals for biomedical articles back to the 1950s with links to full text articles and other related resources.
- *[Artificial Intelligence: A Modern Approach](#)* (Prentice Hall, 2nd edition, 2002) by Stuart Russell and Peter Norvig is the best resource for all kinds of artificial intelligence. This is a must-have book for agent technology.
- "[The Semantic Web Revisited](#)" (Shadbolt, N., Berners-Lee, T. & Hall, W., 2006) is a reflective look at where the Semantic Web stood in 2006, how it got there, and where it is going. It could be seen as a new version of "The Semantic Web" article in Scientific American.
- "[The Gaia Methodology for Agent-Oriented Analysis and Design](#)" by Michael Wooldridge, Nicholas R. Jennings, and David Kinny provides all of the essential information for analyzing requirements and designing multi-agent systems using the Gaia methodology.

Get products and technologies

- [Virtuoso Universal Server](#) is used in the example code to extract Semantic Web data from resources that may not necessarily have explicitly Semantic Web formats.
- [Sesame](#) is a library for Java programming language that enables you to store and manipulate RDF data using an SQL database.
- [Jena](#) is a library for Java programming language that provides advanced manipulation and querying techniques for RDF and other Semantic Web frameworks.
- [Joseki](#) is a library that accompanies Jena to provide a Semantic Web Server.
- [JADE](#) is used in this article to create an agent in Java programming language. It is capable of interacting with other agent systems using FIPA methods.
- [Agent Building and Learning Environment \(ABLE\)](#) is an alternative to JADE created by IBM. It is also FIPA compliant and is, therefore, capable of interacting with many other agent systems, including those built using JADE.

About the author

Daniel J. Lewis

Daniel Lewis is currently a Technology Evangelist for OpenLink Software, where he

demystifies the Semantic Web, enhances Linked Data, and discusses data portability and socio-semantic issues. Between September 2008 and September 2009, he will be a postgraduate at the University of Bristol in the field of Machine Learning and Data Mining. His interests include all kinds of intelligent systems, but particularly dealing with knowledge to try to achieve wisdom. He is also an advocate of open source and cross-platform development. His programming languages of choice include Java, Ruby, Haskell and Prolog. Outside of computing, he enjoys spending time with his girlfriend and reading about religion, philosophy, and psychology. He writes about all of these on [Daniel's Blog](#).