Objectifying the Web the "light" way: an RDF-based framework for the description of Web objects

Pasqualino “Titto” Assini
University of Essex
Wivenhoe Park
Colchester, Essex - CO4 3SQ - U.K.
+44 1206 874333
titto@essex.ac.uk

ABSTRACT

The advantages of object-oriented (OO) programming are well-known. Nevertheless, distributed OO middleware systems (OOM) such as CORBA, DCOM or Java RMI have not been widely adopted for the developing of Internet applications. Developers seem to perceive OOMs as over-complex, proprietary or incompatible with current WWW development practices. Recognizing these difficulties more WWW-friendly proposals based on XML are starting to appear. This paper describes a simple OOM system that uses RDF as its Interface Definition Language (IDL) and HTTP as its RPC protocol. This approach provides most of the advantages of object-orientation while mantaining full compatibility with the existing WWW infrastructure.

Keywords

RDF, Web objects, Web services, IDL, RPC

1. INTRODUCTION

The imminent advent of the OO Web has been announced many times in recent years. The advantages of OO programming with respect to more traditional approaches are so evident that most observers had assumed that it was only a question of time before OO would become dominant in the field of Internet development. Unfortunately the large majority of developers are still sticking to more conventional development technologies. The OO Web is simply not happening: a simple delay or a major failure? It’s not easy to see why such a promising technology might have failed to capture the developers’ imagination but some of these elements might have contributed:

- There is a considerable conceptual and technical mismatch between OOMs and traditional WWW development techniques
- OOMs are perceived by most developers as excessively complex and unsuitable for the development of simple WWW applications
- A growing number of intranets is protected by firewalls that, almost invariably, will allow only email and HTTP traffic. This is bad news for OOMs that are based on non-HTTP protocols.

1.1 Simplify, Simplify, Simplify

Both CORBA and DCOM originate in the pre-Internet era and lack two essential characteristics: simplicity and compatibility with existing WWW technologies. Simplicity is probably nowhere as important as in the Internet environment. A good example is given by the WWW itself. At the beginning of the Nineties, when the WWW was born, there were a number of much more sophisticated hypertext systems available but none of them has had an impact even remotely comparable to that of the ”humble” WWW. We now live in a world that has been throughly changed by the effects of the marriage of the Internet with such a simple technology. OO development has the potential of propelling the Web to a new era but probably needs to go through a similar process of simplification.

2. THE NEOOM FRAMEWORK

Even if the WWW is not a proper OO system it’s not hard to impose some OO principles on it. The OOM described in this paper, the NESSTAR Obiect Oriented Middleware (NEOOM), aims to do exactly so: provide a distributed OO model that it’s as simple as possible and as compatible as possible with the existing WWW infrastructure. Its basic characteristics are:

- its Object Model is an extension of the OO model defined by the W3C RDF [4] and RDF Schema [5] standards
- remote method calls are mapped to standard HTTP calls

2.1 The Object Model

Informally the NEOOM object model can be defined as follows:

- Each object has a unique identifier (an URL)
- Each object has a type (a class)
- All the objects of a given type share the same properties and methods
- A class can be a subclass of any number of other classes

The properties are the attributes of an object. They can be either literals such as a String or an Integer (or any other basic type as defined by XML Schema [6]) or a reference to another object.

A NEOOM object, just as an ordinary Web object, can be retrieved by performing an HTTP GET at its URL. What it's returned is an RDF description of the object properties plus any additional information that the server deems suitable to transmit together with it (for example the descriptions of objects that are linked to the requested object to spare the client the burden of multiple requests). As an object type is also an object it can similarly be retrieved at its URL together with the description of its properties and methods.

2.2 Methods

The methods are the operations that can be performed on an object. A method can take any number of parameters. Methods and Parameters are not present in the basic RDF model but they can be easily added by defining a *Method* and a *Parameter* classes. Instance methods are defined as classes that extend the basic *Method* class. One advantage of defining a method as a class is that methods can have properties and methods. These can be used to control the execution of the method or to precise its semantic. The *Method* class, as currently defined, has two methods: one to *execute* the method and one to *cancel* it. Another advantage is that the method invocations can be represented very naturally as instances of the method class. Parameters are very similar to RDF properties. They have a *domain* property that specifies the method they apply to and a *range* property that specifies the type of the parameter value. The result of a method call is, just as in the case of a normal HTTP request, either an error or a MIME document. Complex objects or set of objects can be returned in XML or RDF format.

2.3 HTTP goes OO

The Object Model that we have briefly examined is abstract and completly protocol-independent. It might easily be mapped to any RPC system such as the increasingly popular SOAP [7]. In practice we have found little need for SOAP's rather extensive functionality. There is an inherent asymmetry in client/server systems: the clients normally make simple calls and receive complex answers. As the calls are simple they can be mapped to standard HTML Forms [3] calls as follows:

- the object that is the target of the method is specified as the form ACTION property
- the method to perform is specified in a hidden parameter of name *method*
- every method parameter of simple type is represented by a form input parameter of type *text* (or *password* for passwords)
- every method parameter of binary type is represented by a form input parameter of type *file*
- if there are many parameters or if one of the parameters is of binary type the form method must be set to POST with an encoding type of *multipart/form-data*, otherwise GET can be used

The advantages of this mapping are: extreme simplicity, efficiency (as hardly any parsing is required to interpret the method calls) and compatibility with existing Web browsers and HTTP libraries.

3. SOFTWARE DEVELOPMENT

To summarize, the steps needed to deploy a NEOOM object are the following:

1. Define the object class in RDF using the NEOOM extensions for the methods
2. Make available the class definition at the class URL
3. Instances of classes that have properties but no methods (object that have state but no behaviour) can be created by simply describing them in RDF and making them available at the object URL
4. Instances of classes with methods are implemented as server-side processes (such as CGI bin scripts or Java Servlets) that accept HTTP calls as specified in section 2.3

3.1 Java Software Development

In the context of the NESSTAR [2] project we have developed a simple Java SDK that provides:

- a tool to generate Java client stubs and server skeletons from NEOOM classes definitions
- an RDF to Java Mapper to easily convert Java objects to RDF and back
- an Object Browser to inspect and administer NEOOM objects through an user-friendly WWW interface

4. ACKNOWLEDGEMENTS

The work reported has been done as part of the NESSTAR [2] and FASTER [1] projects funded by the European Commission's DGXII under the 4th and 5th Framework Telematics Applications Programmes.

5. REFERENCES

[1] Flexible Access to Statistics, Tables and Electronic Resources (FASTER) Project. http://www.faster-data.org.

[2] NEtworked Social Science Tools And Resources (NESSTAR) Project. http://www.nesstar.org.

[3] HTML 4.01 Specification - Forms. http://www.w3.org/TR/html4/interact/forms.html, December 1999.

[4] Resource Description Framework (RDF) Model and Syntax Specification. http://web4.w3.org/TR/REC-rdf-syntax/, February 1999.

[5] Resource Description Framework (RDF) Schema Specification 1.0. http://web4.w3.org/TR/rdf-schema/, March 2000.

[6] XML Schema Part 2: Datatypes. http://www.w3.org/TR/xmlschema-2/, October 2000.

[7] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Simple Object Access Protocol (SOAP) 1.1. http://www.w3.org/TR/SOAP/, May 2000.