Chapter 7
An Overview of REST

Jan Newmarch
Box Hill Institute, Australia

ABSTRACT

The REST approach was developed to describe the architecture of distributed resource access, such as the WEB. REST in comparison with WS_*stack works with the Web rather than against it and is getting increasing support of not only from developers but also from vendors. This paper explains the philosophy of REST and highlights its simplicity in accessing resources.

INTRODUCTION

Since the dawn of networking, computer scientists have been trying to develop frameworks and paradigms for large scale distributed systems. The internet itself provides the backbone for transport and routing, while protocols such as TCP and UDP provide application components with the means of communicating at the application layer. Without a doubt, the WEB has proven itself to be the most successful distributed information system, with other systems such as email running a close second.

Applications such as the Web and email essentially involve human interaction as principal component. But there has been much effort devoted to purely machine-to-machine systems, primarily so that business to business transactions can be conducted across the network. Historically, this has evolved from socket communication, to remote procedure calls, to remote object method calls and to downloadable mobile objects. These are typified by Sun’s RPC, CORBA and Java RMI respectively.

Web Services, using technologies such as SOAP, WSDL and UDDI are the current flavour for building machine-to-machine systems. There
are many claims that Web Services are a significant advance in such technologies, but equally there are many claims that these particular technologies are in fact a step backwards and that by breaking the conceptual basis of the Web their use will in the long term be damaging to the Web.

Critics need an alternative: such an alternative has existed in conceptual form for many years, and is known as REST (Representational State Transfer). In this article we discuss the basis of REST and how it is usually a more appropriate solution to machine-to-machine transactions than Web Services. Our arguments are based on software engineering, and attempt to avoid any other biases.

**PROCEDURE CALL SEMANTICS**

In building any kind of system, a program must be written. Programs are complex and in attempting to give them more manageable structure computer scientists have devised procedures, functions and (within the O/O paradigm) method calls. All of these take parameters and return results.

A key to the success of this structuring is that parameters can be divided into *value* and *reference* parameters. With value parameters, the current value of a variable (a simple type, a structure, an array) is passed to a procedure and a local copy is operated on. With reference parameters, the *address* is passed in and changes can be made to this external address within the procedure. There are many variations on reference parameters: Pascal *var* parameters, C passing of explicit addresses, tcl’s *upvar* and Ada’s *in/out* mechanism (which defers writes until exit from the procedure).

Whatever the mechanism, there is an essential difference between the *value* of a variable and the *address* of the variable. Even O/O languages offer the same distinction: for example, in Java you pass a simple value or the address of an object.

**HTML**

HTML documents as presented to a browser or other user agents such as a search engine or portal contain a mixture of text, markup and hypertext references. In effect, the text and markup are the *value* of the document and hypertext references are the *addresses* of other documents. This mixture of value and reference is a significant component in what makes the Web successful: a user can read the content supplied and follow embedded addresses to other documents of interest. If everything was by value, then the retrieval of any document would fetch the whole Web – a silly proposition!

XML as a successor to HTML/SGML contains the value component through the text of an XML document. References are also built into many XML document types: for example, Docbook has the tag *ulink* while Xlink defined the attribute *xlink*.

**HTTP**

The PhD thesis by Fielding (Fielding, 2000) identified further characteristics behind the design of the Web: The Web is built of resources, and each resource has an address (a URI). Fetching that address returns a *representation* of what is at that address. The *value* of an *address* is some piece of data (typically HTML) that represents the resource. The actual resource may be a copy of a static document stored on disk or some data constructed by lengthy computation. That doesn’t matter too much: what is important is that the Web deals with two types of concept: a representation equivalent to a value and URI’s equivalent to references.

Fielding also identified *verbs* associated with addresses. In languages such as Java and C# it has become a design pattern to distinguish data access methods as *getter* methods to retrieve the value, *setter* methods to set a new value while any other methods have unknown semantics and are a little
Related Content

Perceptions of Trust Between Online Auction Consumers
[www.igi-global.com/article/perceptions-trust-between-online-auction/60247?camid=4v1a](www.igi-global.com/article/perceptions-trust-between-online-auction/60247?camid=4v1a)

WebSphere Portal 6.1: An Agile Development Approach
[www.igi-global.com/article/websphere-portal-agile-development-approach/34100?camid=4v1a](www.igi-global.com/article/websphere-portal-agile-development-approach/34100?camid=4v1a)

Industry Portals for Small Businesses
[www.igi-global.com/chapter/industry-portals-small-businesses/17917?camid=4v1a](www.igi-global.com/chapter/industry-portals-small-businesses/17917?camid=4v1a)

An Academic Student-Centered Portal
[www.igi-global.com/chapter/academic-student-centered-portal/17837?camid=4v1a](www.igi-global.com/chapter/academic-student-centered-portal/17837?camid=4v1a)