Meta-Scripts as a Mechanism for Complex Web Services

Murthy Devarakonda
IBM T. J. Watson Research
Yorktown Hts, NY
mdev@watson.ibm.com

Arup Mukherjee
Carnegie Mellon University
Pittsburgh, PA
arup@cs.cmu.edu

Bill Kish*
IBM T. J. Watson Research
Yorktown Hts, NY
c1kish@watson.ibm.com

Abstract

The World-Wide Web is the 'killer application' of the mid-1990's, and as a result it serves as the market place for various services. This position paper discusses the problem of creating complex Web services from simpler ones, under client control. For example, one might want an annotated forecast of a company's stock price from three simple services, viz., stock price data, forecasting, and annotation. We propose meta-scripts as a solution. A client accesses a meta-script, which contains procedures for accessing and composing simpler services. Meta-identifiers are used in the meta-script instead of static links. Developments in type-consistency mechanisms and various enabling services of the Web help in realization of the meta-script based approach. This model of service composition separates control from data manipulation, and gives this control to a client. As clients are empowered with the selection and coordination of service providers, this can lead to a free market for services.

1 Introduction

The World-Wide Web [1] is a rapidly emerging paradigm for accessing information on the Internet. The basic method of access in the Web is through specification of a Universal Resource Locator (URL) [2]. A Web client determines how to present the accessed information based on the MIME type [3] of the retrieved document. The MIME type is usually supplied by the information server, or deduced from the filename extension (e.g., "`.html`") of the URL. For example, most HTML [4] documents have the MIME type `text/html`, and a Web client will usually choose to format and display such documents itself. A postscript file with a MIME type of `application/postscript` might, on the other hand, be turned over to an application such as GNU's ghostscript for previewing. HTML also supports form-based information access.

As the Internet and Web gain popularity, various kinds of Web services will become available. For example, a Web server may simply provide raw data in a semi-structured format, such as a flat file representing the daily closings of a company's stock price over a period of time. Other services may offer data enhancement, rather than raw data, such as the generation of annotated graphical presentations, or the computation of various statistical measures, given a flat file of stock prices as input.

Given such a set of simple services, consider the following complex service one might want from the Web: Forecasting a company's stock price for the next few weeks using a sophisticated linear regression and presenting results in an annotated plot. Although there may not be a single Web server capable of doing it all, there may be a stock price provider, linear regression analyzer, and a plotting service. The present Web access method requires direct coordination among servers to provide the desired service. The plotting service must know of and access the linear regression analyzer, which in turn must know of and access the stock price provider. In the real world, the servers may not have the authorization, ability, or willingness to coordinate with each other as necessary. In this position paper we propose an access method where the coordination of the services takes place in the client, thus avoiding this problem.

2 Meta-Scripts for Complex Services

Figure 1 and Figure 2 show the distinction between the present access method and the one being proposed. In the present scheme, as shown in Figure 1, a client receives service from a single logical server. In providing the service, the server may access other servers or may offer a complete solution all by itself. In the present method, the only input a client sends to the server is a URL specification or a filled-in form. Specification of search strings can also be thought of as form filling.

In the proposed method, a client first contacts a *meta-server*. The meta-server provides *meta-scripts*, which can be written using an extended HTML and meta-identifiers instead of the usual static links. For example, for the stock price forecasting example, the meta-script outlines the procedure for accessing individual servers, inputs and outputs, and the sequence.

*B. Kish is a consultant to IBM from Brown Cow Engineering, Millerton, NY.

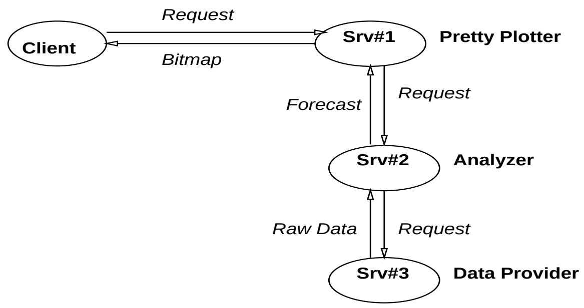

Figure 1. The present approach to complex services

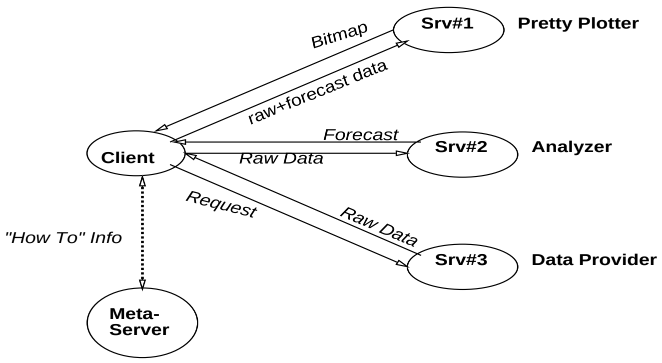

Figure 2. A Client-Based Approach to Obtaining Complex Services

The names of the individual service providers are not hardwired in the script but they are plugged in just before use (i.e. late binding).

The service provider names may be filled in by the meta-server using the identity of the client and additional information provided by the client. A client may have a preferred list of servers for well known services, or its location on the net may suggest “nearby” servers for efficiency.[1] Alternatively, the client may fill in the service provider names using cached provider information. In either case, the server names provided by the meta-server are treated by the client as suggestions and may be overridden if other, better, servers are available.

A meta-script could be written in many ways to provide the stock price forecasting service as outlined in the Introduction. One way to achieve the service is shown in Figure 2. The client obtains raw data about a company's stock price from Server #1. The client then provides this raw data to Server #2 and obtains a forecast for the company's stock price. Lastly, the client provides the raw and forecast data to Server #3 and obtains a bitmap of the annotated plot.

[1]A catalog service may even exist to recommend servers based on proximity, cost, or other metrics.

A clear difference between the proposed method and the present one is that the client has control over what servers are chosen for the services and what information passes between the servers. This can be important in real life situations where fees are charged for a service and auditing of data access is essential. The negative side of this approach is the set of extensions necessary to the Web software. Even more significantly, a large amount of data now passes from client to the server (such as the raw data for the stock price in the above example). The second problem can be addressed using a wide-area file system such as the AFS [5], and by passing around an AFS path name instead of actual data. Even if a wide-area file system is not used, a URL can be used instead of passing the data around. The key point is that the client retains control over what servers are used and what data is passed from one server to another. This proposed approach leads to a free market approach to obtaining complex services rather than being forced to receive a bundled service from a single provider.

3 Background

The meta-script mechanism separates simple services and their composition, and composition is carried out under explicit control of the client. Earlier, Mukherjee and Siewiorek [6] proposed a broad application taxonomy based on the communication, data manipulation, and control characteristics of applications. Their analysis suggests that, in the context of a mobile environment, services may benefit in several ways from appropriate separation of data manipulation and control processing. The meta-script mechanism advocates one such separation, corresponding to the "user agent" scenario of [6]. Individual services manipulate the data under the control of a client-side meta-script, resulting in a more flexible service architecture.

The meta-script model might also be regarded as a client-centric approach to service composition, modelled after the client-centric arbitration protocols of the Calypso distributed token manager [7]. Tokens are used in some distributed file systems to convey the authorization to access data and thus to support cache consistency. Most such systems employ a server-centric protocol, wherein the file server is responsible for revoking outstanding tokens [8]. However, in the client-centric approach, each requesting client is responsible for revoking outstanding tokens after obtaining the list of token holders from the server.

In a position paper, Black and Walpole [9] propose the HTTP daemon as an object-oriented, next generation operating system. They point out that the present Web architecture lacks general purpose mechanisms for dynamic concatenation of 'services' and extensibility, which are the hallmarks of an object-oriented system. The meta-scripts mechanism along with type-oracles and enabling services discussed later in this paper, may fulfill these requirements.

The meta-script based approach can also be seen as a vehicle for resource discovery, coordination, and caching in a distributed system. By taking this view, one can begin to identify some of the basic facilities necessary for meta-scripting. Resource discovery requires a standard means of naming, describing, and cataloging available Web services. Some of these facilities are already becoming available or being planned for the Web as we discuss later (Section 5) in this paper. The basic coordination function needs to be built into the Web clients such as Mosaic and Netscape. Such coordination also raises the issue of type-consistency between the producer and the consumer. In Section 4, we discuss this issue in further detail.

A meta-script implementation, as proposed in this paper, need not provide fully general client-side computation. Minimally, it requires a mechanism to allow the composition of simpler component services to obtain a desired result, and a means for the late binding of service provider names to the capability-oriented service descriptions found in a meta script. As such, a minimalistic implementation avoids the potential security problems that arise from supplying a client with fully general computation capabilities. However, in future, it may become feasible to implement meta scripts as a subset of the capabilities of a general scripting language that has been designed to be practical and safe. Current scripting languages, such as tcl/tk [10] offer much promise, but must first be restricted to their "safe" subsets, a goal that is the focus of efforts such as safe-tcl [11].

4 Data Type Consistency

When two or more simple services are combined, whether by means of a meta-script, or by the conventional approach, the services must be type-consistent with each other, i.e. every service that is a producer of data must emit a format that is acceptable as input to the services that consume it. Although this problem is not specific to meta-script oriented services, the need to ensure type-consistency limits the extent to which a meta-script can be independent of the underlying services it employs.

The HTTP 1.0 standard [4] provides a coarse-grained solution to the type-consistency problem. A client may send `Accept` and `Accept-Encoding` headers, when making a request of a server, to indicate the data types it can accept as output. A meta-server could make use of this mechanism to guarantee that each simple service produces data in a format appropriate to its later use. Unfortunately, however, this approach is rather limited. The MIME types used to specify data formats must be endorsed by a centralized authority for widespread applicability, subtyping is not well supported by this approach (MIME-types provide only a two-level hierarchy), and there is little or no support for the recognition of distinct types that are equivalent. (Two distinct types are equivalent if a sequence of convertors exists to convert one type to the other and vice versa.)

A more flexible solution to this problem lies in the currently ongoing development of *type oracles* [12] and similar services for the Web. Type oracles present a distributed, extensible lattice of types that can be used to describe data. Clients can create types with a granularity as fine as they need. For example, a client might accept as input a "table with 1 row and 7 columns" type, rather than the general MIME type `text/plain`, when its input is the performance of a stock over the course of a week. Type oracles also allow the registration of type-conversion services, are able to determine type equivalences, and can perform data conversions when required. If a type oracle is available to a meta-script, the script can be written at a level that is much less dependent upon the specific services it accesses – the script need not anticipate the advent of new data types, or consider output formats, when selecting from amongst otherwise identical servers.

5 Enabling Services

In order to fully realize the Web's potential as a generic interface for distributed services, some additional infrastructure is required. Currently, work is in progress to address the need for privacy and authentication, both at the HTTP level [13] and below it [14]. *Type oracles*, as described above, can help ensure that data is available in a format that the client can understand. *Uniform Resource Names* [15] are being considered to permit documents to change their physical location without affecting existing links. Other services that may prove valuable include Auto Fill Forms, Common Gateway Interface (CGI) for Clients, and Standard Search Interfaces.

Auto Fill Forms: Currently, many forms returned by HTTP servers contain fields that could be easily filled in by the browser with specified default values. For example, a field marked "enter your full email address" could be automatically filled if there was a standard tag to indicate to the client that this was a *UsersEmailAddr* field. This would be valuable as more services become reserved for *Registered Users Only.*

CGI for Clients: A Common Gateway Interface for *clients* has been proposed [16]. This may be a smarter way to achieve auto fill forms and more. A secure and authenticated method for authorized servers to execute scripts on a client machine would permit the server to find out about the user's preferences and capabilities of the client machine. More personalized services could be constructed that could call a client back when the interesting information is available. This is ripe for abuse and requires an intelligent, secure implementation, but this type of service promises to add significant value to the Web.

Standardized Search Interface: The Standardized Search Interface would allow a client to spawn off multiple search threads against different servers without knowing about the servers specific search interface. Currently, most search engines on the Web provide unique forms interfaces that require human intervention or at least knowledge of the form to submit. A standardized mechanism for specifying search criteria and the returned references would permit more types of automated searching.

6 Conclusion

Meta-scripts are an interesting approach to providing Web clients with an enhanced level of control over the actions taken to obtain a specific data object. By separating service accesses and service composition from the actual data manipulation, the user gains additional control over which servers get to process requests and what the final result of a request looks like. We plan to investigate the flexibility and scope of this approach through its application to a real life example. Several research issues arise, such as how much of a meta-script can be modified, can modification be modified further, how to syntactically define modifiable entities (self describing or pre-programmed), and suitability of the HTML for these purposes.

Acknowledgements

We thank the anonymous referees for their comments, which guided us in preparing the final version of the paper.

References

[1] T. Berners-Lee, R. Cailliau, J-F Groff, and B. Pollermann. Word-Wide Web: The Information Universe. *Electronic Networking: Research, Applications, and Policy*, 1(2), Spring 1992.

[2] T. Berners-Lee, L. Masinter, and M. McCahill. Uniform Resource Locator. Technical report, Internet Engineering Task Force, Draft RFC, 1994.

[3] MIME (Multipuprpose Internet Mail Extensions). Internet RFC 1521.

[4] T. Berners-Lee, R. T. Fielding, and H. Frystyk Nielsen. Specification of the HTTP 1.0 Standard. Available on the Web at http://www.w3.org/hypertext/WWW/Protocols/HTTP/HTTP2.html.

[5] J. H. Howard, M. L. Kazar, S. G. Menees, D. A. Nichols, M. Satyanarayanan, R. N. Sidebotham, and M. J. West. Scale and Performance in a Distributed File System. *ACM Transactions on Computer Systems*, 6(1), February 1988.

[6] A. Mukherjee and D. P. Siewiorek. Mobility: A Medium for Computation, Communication, *and* Control. In *Proc. 1994 Workshop on Mobile Computing Systems and Applications*, Dec 1994.

[7] A. Mohindra and M. Devarakonda. Distributed Token Management in Calypso File System. In *Proc. of IEEE Symp. on Parallel and Distributed Processing*, Oct 1994.

[8] M. L. Kazar and others. DEcorum File System Architectural Overview. In *Proc. of Summer USENIX Conference*, June 1990.

[9] A. P. Black and J. Walpole. Objects to the Rescue! or httpd: the next generation operating system. In *Proc. 6th ACM SIGOPS European Workshop*, Sep 1994.

[10] John K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, 1994.

[11] Nathaniel S. Borenstein. E-mail With a Mind of Its Own: The Safe-Tcl Language for Enabled Mail. In *IFIP 6.5 International Working Conference on Upper Layer Protocols, Architectures and Applications ULPAA'94*, 1994.

[12] John Ockerbloom. Introducing Structured Data Types into Internet-Scale Informations Systems. Ph.D. Thesis Proposal, Carnegie Mellon University, Available on the Web at http://www.cs.cmu.edu:8001/afs/cs.cmu.edu/user/spok/www/proposal.html, May 1993.

[13] The Secure HyperText Transfer Protocol. Internet Draft, Available on the Web at http://www.eit.com/projects/s-http/shttp.txt.

[14] Netscape. Internet Data Security: An Overview of Implementation and Plans from Netscape Communications Corporation. Available on the Web at http://home.mcom.com/info/security-overview.html.

[15] Tim Berners-Lee. WWW Names and Addresses, URIs, URLs, URNs. Specification available on the Web at http://www.w3.org/hypertext/WWW/Addressing/Addressing.html, 1994.

[16] Tim Berners-Lee, Ed. Notes from the Development Priorities BOF. Proc. of World-Wide Web Conference, Available on the Web at http://www.w3.org/hypertext/WWW/WorkingNotes/WWW94/DevPriBOF.html, May 1994.