W3P: A Portable Presentation System for the World-Wide Web

Christopher R. Vincent
Intelligent Information Infrastructure Project
MIT Artificial Intelligence Laboratory
cvince@ai.mit.edu
http://web.mit.edu/cvince/

Abstract

As World-Wide Web applications become increasingly interactive, interpreting user input incurs a significant amount of overhead and complexity on the host server. W3P is an abstract, extensible Common LISP system for manipulating input and output as CLOS objects, allowing simpler applications with less code duplication. To facilitate compatibility with existing LISP applications, W3P implements a subset of the LISP interface to the Common LISP Interface Manager. W3P, however, was conceived to specifically address the requirements of a stateless model of interaction such as the World-Wide Web. Once a CLOS class has been profiled with its essential parameters and visual attributes, input type-checking and output rendering can be performed automatically. Developers can utilize standard interface characteristics or choose to define their own visual representations of objects and input parsing routines. W3P has been applied to CL-HTTP, a portable, full-featured World-Wide Web server implemented in Common LISP. Major advantages are evident in form-processing, which mostly involves parsing, validity checking, and building objects from textual input. W3P represents an effort to create a streamlined, highly portable, non-proprietary presentation system tailored to the stateless model of the World-Wide Web.

Introduction

The rapid proliferation of the World-Wide Web has led to fundamental changes in its model of user interaction. Servers have become more than simple information repositories, providing dynamic content tailored to the needs of individual users. Utilizing the World-Wide Web as a medium for interactive applications places considerable strain on conventional development techniques. W3P is a modular, extensible Common LISP system for manipulating input and output as CLOS objects, allowing abstract management of dynamic content and user interaction. W3P implements a subset of the LISP interface to the Common LISP Interface Manager. This design consideration facilitates basic compatibility with existing Common LISP systems. While the core functionality of the W3P system is designed for compatibility with CLIM, it is structured for the requirements of a rather different model of user interaction. The stateless nature of the World-Wide Web requires that a presentation system perform gracefully without a persistent connection to the user, as well as addressing the performance requirements of a production server. CL-HTTP [3], a server implemented in Common LISP, has been used to apply a number of existing dynamic systems to the World-Wide Web. As in any interactive application, a great deal of overhead and complexity is incurred interpreting user input and rendering output. W3P abstracts away much of this functionality, resulting in simpler implementations with less code duplication. Dealing with user interaction at a higher level allows developers to more efficiently construct powerful, dynamic World-Wide Web applications.

System Structure

The basic W3P system facilitates the definition of a presentation type lattice similar to that of the Common LISP Object System, allowing inheritance of specializing parameters and options. Presentation generic functions are defined to act on these presentation types, utilizing CLOS method combination. The system for registering and managing presentation types and generic functions was implemented in about 400 lines of portable Common LISP code. Standard presentation type definitions similar to those in CLIM are supplied, as well as standard presentation methods. Standard methods include those for parsing input, rendering output, handling input errors, and prompting a user for a particular type of data. While this set of methods will be sufficient for most developers, additional functionality can be defined using the interface to the basic system.

The W3P system also allows the definition of presentation views that use CLOS class inheritance. Views facilitate the specialization of presentation methods over different content types or styles. Consider, for instance, a presentation type describing an item of technical documentation. The developer may wish to specify methods for plain textual output such as electronic mail, for a mark-up language like HTML, or even for a graphical representation. In addition, one may with to define an abbreviated view for technical experts and a detailed view for novice users.

W3P condition handling must also be highly modular and configurable. Varying applications will have very different needs when it comes to responding to input errors. Applications such as detailed online surveys will need to gracefully prompt the user for precisely the set of data that was found invalid. Some applications may choose to proceed without certain data rather than initiating another prompt. If error-detection has already taken place on the client end, a server may abort a process as soon as an invalid input item is detected.

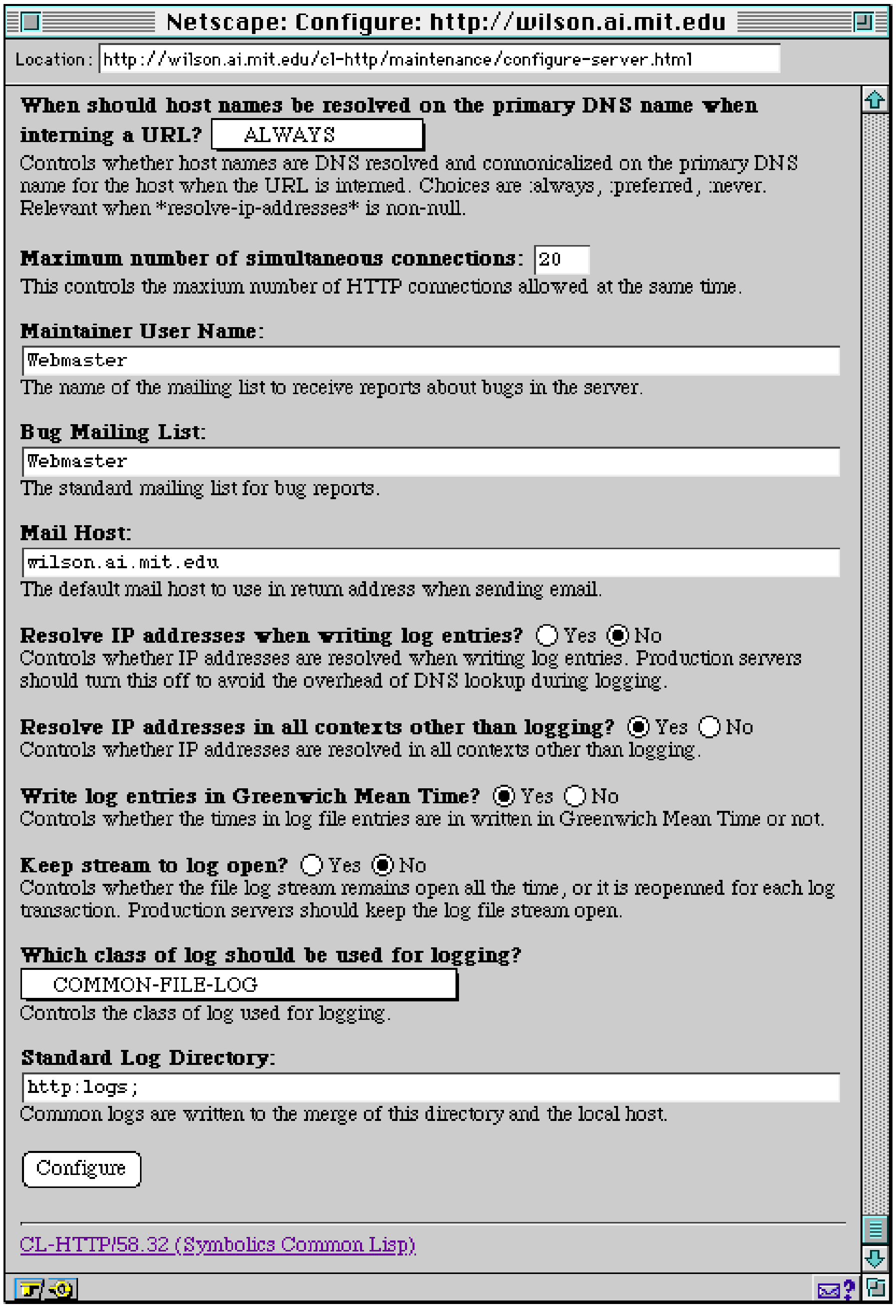
W3P and CLIM

W3P and CLIM operate on the same basic principle: profiling CLOS objects through presentation types, allowing developers to abstractly manage input parsing and output rendering. Some CL-HTTP applications already use CLIM for managing dynamic output. While CLIM can be configured to function in a stateless model like the World-Wide Web, it was often found to be a much more powerful tool than was necessary. CL-HTTP requires a lightweight, highly portable, non-proprietary presentation system designed specifically for the tasks at hand. The basic concepts that W3P follow can be traced back to, among others, Eugene Ciccarelli’s PTBase system [1], which used presentation types in conjunction with presenter and recognizer methods. An early implementation of W3P used a different LISP interface, but a design decision was made to facilitate some basic CLIM compatibility. The design objective was not to recreate a full-featured user interface manager, but to allow developers to utilize existing libraries of presentation types and parsing routines. Many CLIM capabilities such as output histories and real-time completion are largely inappropriate when applied to the World-Wide Web in its current state. Valuable features for conventional LISP applications can become unnecessary overhead in a Web interface. W3P does not suffice as a replacement for CLIM, but will help fill the need for a streamlined presentation system tailored to the World-Wide Web. The hope is that W3P will help bring dynamic object presentation systems to a wider base of developers.

Form Processing

W3P is especially valuable as a tool for form-processing. Implementing World-Wide Web interfaces which utilize HTML forms capabilities consists mostly of parsing textual input, type and validity checking, and building CLOS objects from the parsed input. This type of low-level, repetitious code increases complexity, is difficult to reconfigure, and is prone to programmer error. Abstracting away these tasks allows rapid prototyping and efficient reconfiguration of applications. Generically parsing and rendering objects allows not only dynamic content, but a flexible interface tailored to the needs of a particular user. An application has the ability to deliver a dynamic form, requesting a precise set of required input data. Presentation types and configurable condition handling allow meaningful error reporting when input is invalid.

An initial application of W3P has been a remote server configuration system for CL-HTTP. Through a password protected World-Wide Web document, webmasters can dynamically configure parameters such as the number of simultaneous connections allowed or how logging should be performed. When combined with Web-based log analysis tools, this allows maintainers to remotely observe server performance and act upon the information immediately. W3P is an important part of this process for several reasons. Type and parameter checking are especially important here, protecting against accidental corruption of vital server objects. Using presentation types allows a highly configurable interface. For instance, some maintainers may only be granted access to certain configuration parameters. Some webmasters may require more detailed parameter explanations, while others may prefer a very basic interface.



Netscape: Configure: http://wilson.ai.mit.edu

Location: http://wilson.ai.mit.edu/cl-http/maintenance/configure-server.html

When should host names be resolved on the primary DNS name when interning a URL? ALWAYS

Controls whether host names are DNS resolved and connonicalized on the primary DNS name for the host when the URL is interned. Choices are :always, :preferred, :never. Relevant when *resolve-ip-addresses* is non-null.

Maximum number of simultaneous connections: 20

This controls the maxium number of HTTP connections allowed at the same time.

Maintainer User Name:

Webmaster

The name of the mailing list to receive reports about bugs in the server.

Bug Mailing List:

Webmaster

The standard mailing list for bug reports.

Mail Host:

wilson.ai.mit.edu

The default mail host to use in return address when sending email.

Resolve IP addresses when writing log entries? ○ Yes ● No

Controls whether IP addresses are resolved when writing log entries. Production servers should turn this off to avoid the overhead of DNS lookup during logging.

Resolve IP addresses in all contexts other than logging? ● Yes ○ No

Controls whether IP addresses are resolved in all contexts other than logging.

Write log entries in Greenwich Mean Time? ● Yes ○ No

Controls whether the times in log file entries are in written in Greenwich Mean Time or not.

Keep stream to log open? ○ Yes ● No

Controls whether the file log stream remains open all the time, or it is reopenned for each log transaction. Production servers should keep the log file stream open.

Which class of log should be used for logging?

COMMON-FILE-LOG

Controls the class of log used for logging.

Standard Log Directory:

http:logs;

Common logs are written to the merge of this directory and the local host.

CL-HTTP/58.32 (Symbolics Common Lisp)

Figure 1: A portion of the remote server configuration interface to CL-HTTP.

This interface is generated using W3P in conjunction with HTML synthesis tools and a system for managing CL-HTTP server preferences. The entire interface is implemented in about 70 lines of Common LISP, including methods for computing the form and responding to input. The following selection of code is the loop that generates the html form in Figure 1, where *remote-configuration-form-prototype* is a list of keywords specifying CL-HTTP server preferences.

```
(html:with-fillout-form (:post url :stream stream)
  (loop for item in *remote-configuration-form-prototype*
    for preference = (find-preference item nil)
    when preference
    do (html:with-paragraph (:stream stream)
         (w3p:accept (preference-presentation-type preference)
           :stream stream
           :view w3p:+html-view+
           :present-p t
           :default (funcall (preference-value-getter preference))
           :prompt (preference-prompt preference)
           :prompt-mode :raw
           :display-default nil
           :query-identifier (symbol-name (preference-keyword preference))
           :insert-default t)
         (html:break-line :stream stream)
         (write-string (preference-description preference) stream))
         finally (html:with-paragraph (:stream stream)
                   (html:accept-input ’html:submit-button "SUBMIT"
                                      :display-string "Configure"
                                      :stream stream))))
```

Using W3P along with CL-HTTP HTML synthesis tools allowed the rapid construction of this functional interface to a Common LISP system. Future candidates for W3P include COMLINK, a system that already implements dynamic forms using CLIM substrate.

Client-Side Extensions

Future plans for W3P include the utilization of mobile code to create more powerful user interfaces on the World-Wide Web. Client-side extensions will allow input completion and preliminary type-checking, as well as less cumbersome graphical elements. While mobile code can help provide some real-time feedback to the user, the interface must still perform gracefully without a persistent connection to the server. This means that input completion is somewhat limited, since it is only feasible to send relatively small sets of data along with an HTML form. Mobile type and parameter checking will significantly reduce server computation by moving most input error reporting to the client. A final input validation must still be performed on the server, but without the potential overhead of generating a meaningful error report. While generating this kind of disconnected interface may seem awkward at first, it will be a valuable tool as the World-Wide Web continues to proliferate. Pushing this functionality down toward the client allows servers to interact with a larger base of users. A W3P presentation embedded with mobile code can be cached and redistributed like any other resource.

Consider the case of a widely-distributed user survey conducted over the Web. A form would generally include a number of questions, each with a corresponding set of acceptable input. In the conventional model, a user must commit to a large set of data at once, waiting for the server to validate input. The server potentially becomes a bottleneck, generating customized error reports for thousands of users. When a W3P presentation includes most of this functionality as part of the form, the server can deal more efficiently with incoming requests. Since input validation has already occured, the server check is simply a security consideration. Requests containing invalid input may be discarded and a generic error returned. As the World-Wide Web evolves into a more decentralized, cache-oriented environment, this type of highly portable interface becomes increasingly important.

Conclusions

W3P is an effort to create a streamlined, highly portable presentation system for managing complexity in stateless models of user interaction. While the system was designed with the World-Wide Web in mind, it can be readily applied to other interfaces such as automated electronic mail. As the World-Wide Web becomes increasingly interactive, developers need a greater selection of tools for dealing with user interaction at a higher level. W3P is a simple, yet highly extensible Common LISP system for presenting and accepting CLOS objects to and from the user. Partial CLIM compatibility facilitates use of existing presentation types and parsing routines, as well as introducing a larger base of developers to some of CLIM's basic capabilities. W3P has been tested with CL-HTTP as a tool for creating user interfaces on the World-Wide Web. Major advantages are evident in form processing, where parsing and type-checking incur a significant amount of overhead and complexity. W3P's modular system structure, allowing extension and performance tuning, lends itself to use as part of a production World-Wide Web server. Client-side extensions will use the same presentation model to output interfaces that are more responsive to the user. The evolving role of the World-Wide Web as an interactive medium has emphasized the need for presentation type oriented interface development tools. W3P helps address this need with a simple, portable, configurable presentation system.

References

1. Ciccarelli, E.C., Presentation Based User Interfaces, AI Technical Report 794, August 1984.
2. Symbolics, Common Lisp Interface Manage (CLIM): Release 2.0, Symbolics Inc., Jan. 1994.
3. Mallery, John C., A Common LISP Hypermedia Server, Proceedings of The First International Conference on The World-Wide Web, Geneva: CERN, May 25, 1994.