ECE251: Programming Languages & Translators Fall 2010

Lecture 30 — November 30, 2010

Patrick Lam *version 1*

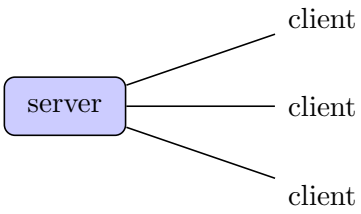
Domain-Specific Languages

Apart from some discussion of DSLs at the beginning of this term (for instance, during our discussion of scanning), I've mostly been describing standard compilers techniques in this class: scanning, parsing, symbol tables, type checking, code generation.

Your course project, however, is one example of a domain-specific language: WIG makes it easier to write CGI scripts (for those Web 1.0 applications). Let's start our discussion of DSLs by surveying domain-specific languages for the Web.

How the Web works. Let's get everyone on the same page by describing the Web and the Internet in general. At a high level, the Internet connects a set of devices, each which a unique address[1]. Any device can open a connection to another device, specifying a port for the connection.

The World Wide Web builds on top of the Internet. In particular, the *Hypertext Transfer Protocol* (HTTP) is a set of rules which allows clients to request resources from servers over the Internet:



In an HTTP request, a client requests a specific *resource* from the server. The server responds, typically by sending an HTML page, but perhaps also images, PDFs, etc.

Early implementations of HTTP had a 1:1 mapping between resources and files on the server. Almost no pages are like that nowadays—I can't name any, can you?

Server-side Processing

The next obvious step is for the server to do some computation before sending a resource. My course web pages are like this: Apache includes headers and footers for each page. Preprocessors are a well-known computer science idea, and we're just applying that idea here.

[1] Not quite true, but close enough.

From the Apache 1.3 manual, `http://httpd.apache.org/docs/1.3/howto/ssi.html`:

> SSI (Server Side Includes) are directives that are placed in HTML pages, and evaluated on the server while the pages are being served. They let you add dynamically generated content to an existing HTML page, without having to serve the entire page via a CGI program, or other dynamic technology.

Apache SSIs let you include the results of various (small) computations. This is a small domain-specific language, whose syntax is embedded in HTML comments. Let's think about the compiler technologies that Apache uses to implement SSI.

- Finding input files: Apache's configuration tells it which files it should server-side parse, e.g. with the `XBitHack` directive.
- Scanning/parsing: SSI directives always look like `<!--#element attribute=value ... -->`; what technology do we need?
- Interpreting: Elements are commands. They can read (`echo`) and write (`set`) variables, as well as evaluate conditional expressions (`if`), include files (`include`) and execute programs and include their outputs (`exec`). Note the use of a map of variables here.

CGIs. It's also possible to have the web server delegate to an external process. The *Common Gateway Interface* specifies a standard way for servers to execute external processes. The server communicates information to the process via environment variables and on standard input (namely arguments to the page), and the process returns output to the server on standard output, which the server serves to the client.

Programming languages for dynamic webpages. CGI (and its successors, e.g. FastCGI[2]) is independent of the technology used to actually implement the CGI scripts.

Your WIG compiler produces CGI scripts which Web servers can execute. Running these scripts from the command-line doesn't do much, but I'll provide a way for you to test these scripts from the web server on `ece251.patricklam.ca`. WIG was designed to exercise all the parts involved in typical domain-specific languages, and you've seen that in your project.

There are many languages for writing dynamic webpages, including PHP, Ruby on Rails, and so on. Sometimes the server itself is written in Java and calls servlets, as in the case of Java Enterprise Edition.

Databases. I should mention that databases are also ubiquitous in web programming. They are much better at running queries than filesystems. So DSLs for the web ought to make it easy to interact with databases, at least in the common case. PHP doesn't seem to do this (you pass an SQL database a string to do a query), but Ruby on Rails does.

[2]Eliminates startup delay by making the script stay around waiting for more work.