Gregor: A DSL for building semester calendars

Matthew M. Burke
Department of Computer Science
The George Washington University
mmburke@gwu.edu

After more semesters than I care to admit of creating HTML-formatted schedules for my course syllabi by hand, I finally came to my senses and started designing a software solution. Given that there are a number of ways I need to format and use the information, I decided that the best approach was to design a Domain Specific Language (DSL); hence, I embarked on designing Gregor.

The main focus of this paper is an evaluation of Tcl as a tool for building DSLs. This includes a discussion of several techniques for using Tcl to implement DSLs, as well as an examination of Tcl's suitability in this context as compared to several other languages, namely Lua and Ruby.

Summary

Putting together a schedule for a course (listing which topics are to be covered on what days, when the examinations are, due dates for assignments, listing holidays, etc.) is tedious and error-prone. Moreover, even when an instructor diligently revises her course between semesters, the topics covered and the order in which they are addressed is often quite similar, yet the dates will all need to be updated. Clearly, there is a lot of work that can be automated in this process.

The first step in designing a solution for this problem was to decide on a data representation. I decided on designing a DSL to describe the information for several reasons: much of the information is pulled from sources where it is already represented in some form of markup langugage (such as semester dates from the academic calendar (HTML), class dates from the schedule of courses (HTML)); ease of insertion into a version control system; and ease of manipulation using command line tools, pipes and filters. Given the decision to implement a DSL, there remained the question of how best to implement it. Although Ruby is currently experiencing a great deal of popularity in this context, I chose Tcl because of my greater familiarity and experience with the langauge.

Once the DSL approach was decided upon, there were still a number of decisions concerning "style," such as whether to design a so-called "internal" or an "external" DSL, and how best to take advantage of Tcl in the implementation. Several iternations of Gregor are described along with an analysis of what's gone well and what hasn't. Finally, we will discuss some of the pros and cons to using Tcl for DSL implementation as compared to other languages, specifically Lua and Ruby.