Literature Review + Outline: Object Shells

DeDominic, Anthony
Eastern Connecticut State University
Willimantic, USA
dedominica@my.easternct.edu

Abstract

This is merely the background on my subject, tying together research garnered over the weeks. Below I will discuss the importance of a shell with modern data structures and object like constructs I will also discuss problems that are being solved with such shells. I will also talk about processes that will be used in my project.

1. Introduction

1.1 Related Works

Object Oriented shells isn't an etirely new concept. As others have shown, many projects, some more serious than others, attempted to solve this issue [1] [2].

The power of shells is well understood. Most programming languages have some kind of system() or shell invocation mechanism. Some, such as the developers of shcaml attempted to take it further by including functional combinators; these combinators allowed for slipping in native ocaml code and objects. As a result these code pieces could be parsers, that could take known shell commands, and structure their data into key-value trees [1].

Purely object oriented attempts ultimately result in new instances of bash and explicit message passing with named pipes [2].

1.1 SOA

Service Oriented Architectures, or SOA, are the modern way to construct, highly available, data rich web applications. In short, a service oriented architecture is a way for applications to share state and provide services over middleware and structured, serialized objects. [3] One way of transacting states between services is through Representational state transfer, or REST. With REST, services are able to share well structured data through document structures like JSON, XML, etc. [4]

Currently popular shells have no built-in feature complete way of transacting or using structured documents. As a result, working with web oriented services becomes a hassle; A hassle that involves very long and contrived text pipelines full of tools like: sed, grep, awk, xargs, tr, paste, cut, tee, and so on. What results is some sort of string tokenized parser.

In order to unify the web and systems management, tools like ansible were created. Ansible, ends up using a general purpose language like python which can natively handle these structures. Projects like powershell are built on the .NET runtime and can thus utilize class like features and functions. The streaming pipelines in Powershell are merely .NET classes which allow for some structural correctness and parsing [5]. As a result, Microsoft prefers to call more than just a shell, but a whole "automation and configuration management framework." One that is capable of handling structured data such as JSON, XML, CSV, etc, REST APIs, and object models. As a result, the conventional UNIX shell is becoming marginalized by various general purpose languages and domain specific languages.

1.2 Process

Citations

[1] A. Heller and J. Tov, "Caml-Shcaml: An OCaml Library for UNIX Shell Programming," 2008. DOI: 10.1145/1411304.1411316

[2] J. Haemer, "A New Object-Oriented Programming Language: sh," 1994 [Online]. Available: http://https://www.usenix.org/legacy/publications/library/proceedings/bos94/full_papers/haemer.ps

[3] E. Newcomer and G. Lomow, "Understanding SOA with Web services." Addison-Wesley, 2005.

[4] R. Battle and E. Benson, "Bridging the semantic Web and Web 2.0 with representational state transfer (REST)." Elsevier, 2008. DOI: 10.1016/j.websem.2007.11.002

[5] J. Snover, "Monad Manifesto." Microsoft, Aug-2002 [Online]. Available: http://www.jsnover.com/Docs/MonadManifesto.pdf