Comparison of scripting languages

Ajan Jayant
University of California, Los Angeles

Abstract

The aim of this paper is to evaluate the pros and cons of using three frameworks written in different languages to build a web application that has many simultaneous interacting users that need lots of updates. The three frameworks are Node.js, written in Javascript, Twisted, written in Python, and EventMachine written in Ruby. The application is a web-based scheduling client. The paper focuses on the language features and the ease with which the application can be written in the respective language

1 Introduction: Scheduling Application

Though the frameworks themselves may be suited to solving specific internet problems, this paper evaluates the frameworks ability to implement a specific scheduling and collaboration application. The application allows managers to assign a task, a time frame and a workspace to a set of employees. These employees first construct a schedule in which each enters hours and divides the workspace according to their task, with the remaining workspace shared. They then proceed to sign in using the application, and constantly update the workspace. All updates in the shared workspace must must be sent to all employees. Thus, the application involves constant server-user communication, with lots of updates to servers, many updates sent to the clients and lots of simultaneous connections.

2 Python

Python is a multi-paradigm language, which supports object-oriented principles, as well as functional paradigms. All data in a Python program is represented by objects, allowing the passing of function without arguments to other functions; there are classes, though statements are allowed outside them, unlike Java. Python supports recursion and anonymous functions, though side effects can be implemented. Python is a strong, dynamically typed language. There are many built-in types, from numeric to iterator and mapping. Syntactically, it uses indentation to delimit blocks. Python uses reference counting to manage memory; if an object is dereferenced, its object count is set to 0 and its marked eligible for garbage collection. Python deals with exceptions using the try clause; it also allows users to raise an exception. Two unique features are its dynamic typing, enabling variable to be successively assignment to two different types, and mutability vs. immutability of its sequence types.

3 Javascript

Javascript is similar to Python in some ways, but very different in others. It is a Prototype-based programming language; as such there are no classes, and inheritance is performed via a process of cloning existing objects that serve as prototypes. It is far more functional than Python, being built primarily on scheme. It is dynamic, weakly typed, and has first-class functions. There are six types of literals: Array literals Boolean literals Floating-point literals Integers Object literals String literals, far fewer than in python. Its syntax is inspired by C and most similar to Java. Javascript uses a garbage-collector, but requires prototype designers to manage their own memory. The problems of circular references dont exist in Javascript, unlike Python, and it uses a mark and sweep algorithm. Uniquely, it has high levels of browser support and supports both client side and server side development.

4 Ruby

Ruby is a dynamic, reflective, general-purpose object-oriented programming language. It supports the functional style as in it has higher-order functions and First-class continuations. Ruby has classes and can have stand alone functions as well as methods. It supports duck typing and is strongly typed. Its syntax is inspired by Perl and it is garbage collected, like Python. Uniquely among the three languages studied, it allows users to redefine basic control features, like the string quality operator, and also allows the designers multiple ways of designing his own code. This contrasts with Python which forces users to design it a certain way.

5 Comparison of the Three Frameworks

- All three approaches share the event driven model. They have differing implementations with regards to how to write servers, clients and protocols, but can fully implement all these features. They all share the reactor model, for example. As regards to documentation, clearly Node.js has the upperhand. It has a very clear website with all functions (which are the building blocks of a Javascript application) well described with and implementation provided. Twisted has a very good tutorial on how to write basic servers, clients and web clients and differed objects, but for other less documented functions are also present. In particular, it is hard to work up the levels of inheritance and find attributes causing bugs in the program. EventMachine has a fairly strong API refereenc at http://eventmachine.rubyforge.org/EventMachine.html, but lacks the examples of Twisted and simplicity of design of Node.js.
- When it comes to writing protocols, Twisted is by the far the best of the three; includes lots and lots of protocol implementations, meaning that more likely than not there will be an API you can use to talk to some remote system (either client or server in most cases). Javascript uses objects and models each connection as heap allocation. HTTP is a first class protocol in Node. Node's HTTP library has grown out of the author's experiences developing and working with web servers.EventMachine has the EventMachine::Protocols; this supports protocol implementation, but there are less available protocols as compared to Twisted, and there is nothing that matches Node.js' HTTP library.
- When trying to write servers, all three show describe similar ways of of doing so, usually by invoking a version of a run command. EventMachines code is actually the shortest, while Twisteds is a bit more involved, giving you the option of building your own protocol using a definition that you yourself have provided.
- Writing simple client to these servers that are not tied to any specific functionality is easiest on Twisted; while javascript and EventMachine do support it, they are more usually specially optimized for a specific task. As such, Node.js net module provides you with an asynchronous network wrapper. It contains methods for creating both servers and clients (called streams). You can include this module with require('net'); this is an example of a server-client model optimised for the specific network. EventMachine uses connectedclients.map to implement clients in an asynchronous way.
- One of the main design differences is Node.js takes the event model a bit furtherit presents the event loop as a language construct instead of as a library. In other systems there is always a blocking call to start the event-loop. Typically one defines behavior through callbacks at the beginning

of a script and at the end starts a server through a blocking call like EventMachine::run(). In Node.js there is no such start-the-event-loop call. Node.js simply enters the event loop after executing the input script. Node.js exits the event loop when there are no more callbacks to perform. As such, the event loop is hidden from the user.

- Node.js provides support for streaming data constantly, instead of just sending data between clients and servers. It provides superior HTTP parsing tools as well as other functions. EventMachine and Twisted use a stream function and producers respectively, but don't have the same support for constant streaming of data.

- With reference to memory management, all three frameworks are implemented in languages that support automatic garbage collection. EventMachine uses a mark and sweep algorithm, while Python and Javascript use reference counting. As such EventMachine probably has the most efficient garbage-collection; in Python, if an object runs out of memory the user has to manually reclaim it. Javascript has similar problems, but has solved one: circular references no longer go uncollected.

- With regards to scalability, all three do reasonably well. Though there is much debate on the issue, it is assumed Node.js will fare slightly better than the other two due to a lack of blocking functions.

6 Evaluation with respect to the Application

For the chosen application, the best choice would be Node.js. Its superior documentation makes writing code much easier, and its functional style makes using protocols, servers and clients much cleaner. One drawback is that Javascript's built-in types are fewer than Python, so to replicate the functionality of tuples and lists would be difficult. On the other hand, Node.js support for streaming data would be of vital use to this application since the manager might want to view the schedule in real time when the employees are working on it. The main advantage, as mentioned previously, with Node.js is that it has no blocking functions. Not having to even worry about whether or not a library I want to use is blocking is a really useful feature. It also has an easier learning curve, and minor benefits with regards to scalability.

7 References

[1]http://programmers.stackexchange.com/questions/107950/differenc
between-javascript-and-python
[2]http://developers.slashdot.org/story/11/11/06/1526249/analyzing-
stackoverflow-users-programming-language-leanings
[3]http://nodejs.org/about/
[4]https://developer.mozilla.org/en-US/docs/JavaScript
[5]http://theadmin.org/articles/learning-eventmachine/
[6]http://stackoverflow.com/questions/5458631/whats-so-cool-about-twisted
http://eventmachine.rubyforge.org/EventMachine.html