Choosing a RESTful Framework for the RAPDB

Lance Tyler, Matt Morris

CSC: 400

Dr. Gene Fisher

December 5, 2014

Abstract

Choosing a framework for a new web service is difficult. There are many software architectural approaches for designing a framework. In this paper we explore what a RESTful framework is, and why you would choose a certain one over any other frameworks. It also discusses specific implementations of a Restful framework in the three different languages, Java, Javascript, and PHP.

Contents

1 Introduction

One of the most common questions asked by new developers when starting a new web service is what framework, and what language they should write it in. Like most programming projects it is often best to stick with what you know best, however there is a chance you could be creating a fancy new web service in an old, stagnant framework

2 RAPDB Requirements

For our senior project we created a social application development service known as RAPDB. The goal of this project was to make new social application development easier on new developers such as students. The service abstracts away concepts such as database design, web programming, and application security. In choosing a framework we had to balance scalability of the service, as well as have an easy method of interacting with the server. Based on these requirements we determined that a RESTful framework would satisfy most, if not all of our requirements.

3 What is RESTful?

Representational state transfer (REST) is a software architecture style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed hypermedia system. [1] The architectural properties of REST are defined by the following constraints: Client-server, stateless, cacheable, layered system, and code on demand. [1] The sections below illustrate how a select few of the REST constraints fit with RAPDB.

Client-server One of the main constraints, and benefits of the Restful architecture is the way it separates the client from the server. This fits in with our requirements of RAPDB, since the server does not care about the user interface or user state. This also allows the server to scale up easily, adding more servers, or adding in more public methods for the client to interface with. [1] The client-server constraint also allowed us to asynchronously develop our RAPDB demo application known as FoodApp.

Stateless Another constraint of REST is how it should not store the context of the client on the server between requests. [1] RAPDB has multiple users registered to multiple applications. Any request made to a resource must be independent, and not rely on any previous calls.

Uniform Interface A Restful architecture also has the constraint of using a uniform interface, that allows a client to easily access resources. HTTP verbs

such as GET, PUT, and DELETE are used in a meaningful way, and must follow a standard. [4] Sensible names must be used for resources, and developers should be able to understand the purpose of a resource just from looking at the name. [4] Lastly, response codes must be used precisely, allowing the application developer enough insight into the error in question. [4] For the RAPDB service, resource names correctly correspond to their container. For example the container for associations is named such, and the same goes for the data container. Application developers interact with these containers through the HTTP verb usage. A GET on the container will return a collection of the entire container, where a GET on the container/id will return just that single item. To create a new entry on the server the HTTP verb POST is used. [4] In our implementation, the server dictates how IDs are established, which is the standard way in a REST architecture. [4]

4 Choosing a Framework and Language

For newer developers looking to get some exposure to developing RESTful web services, two of the most relevant factors in deciding which framework they will use (assuming they aren't trying to reinvent the wheel) are both the overall learning curve, and the amount of support or examples that exist online from which they can learn. These are discussed below. In the sections below we will cover the learning curve and overall support for the following frameworks: Node.js, Jersey, and SlimPHP. We will also be including our experiences in setting up these frameworks.

4.1 Learning Curve

The learning curve associated with each framework will probably ultimately depend on your familiarity with the language it exists for. Assuming we are measuring this curve for those who already know the language in question, Jersey seems to require the least amount of initial configuration. After importing the necessary libraries, a method can be mapped to a URI pattern with a simple annotation, and as a result will have URI variables and request payloads (if applicable) succinctly passed into the method as arguments. This will satisfy the needs of a majority of users, who aren't seeking advanced features (although those exist too).

Two of the main potential learning curves associated with node.js are both the javascript syntax, and the fact that node.js is single threaded and completely asynchronous. While many developers might be familiar with the basic javascript syntax for basic web editing and DOM manipulation, it is a somewhat foreign concept for many developers to be using javascript as a server sided

language. For those especially unfamiliar with javascript's syntax, or even its odd quirks, this can present some challenges. The fact that node.js is completely asynchronous can also present somewhat of a paradigm shift to those unfamiliar with such an environment. For example, to make a call to the database, a callback must be used, instead of just receiving information back from a .get method. Understanding the use of closures and callbacks is paramount to effectively using the node.js platform. For example when trying to send the method of an object as a callback, we had to wrap the method in a closure, or else the callback would not work.

Once these potential problems are dealt with, adapting restify to fit your needs is relatively easy. Much like Jersey, you simply map a "route", or a URI, to some method which exists as a callback in this particular instance. SlimPHP acts similarly, providing a way to map string representation of URIs to anonymous functions. The biggest hindrance encountered with PHP was simply remembering to use the $ symbol to designate a variable, along with the fact that "." acts as a string concatenation operator.

4.2 Support

Of the three implementations, the Jersey Framework appears to be a clear winner in terms of the size and scope of both the developer community itself, and of the amount of online documentation for it. Jersey, which implements the JAX-RS API specification, has emerged as a contender for the throne as the dominant JAX-RS implementation, with only Spring as a real challenger. Jersey has an entire site dedicated to supporting and clarifying the use of its framework. There exists a user walk-through / entire overview of their API, which is really just the JAX-RS API. Between official and non-official sources, it is unlikely for new developers to run into any major issues that haven't been solved online already, which gives it a great level of support.

Node.js, on other hand, has plenty of official documentation, but is obviously much younger than Jersey. As such, there is a smaller online community supporting questions regarding the node.js language/framework, but that is quickly changing. As mentioned before, the specific framework we used for turning our node.js server into a RESTful service is the "restify" node module. This module yields full control over the HTTP protocol from within node, and actually has highly detailed and practical documentation. Between the native node language, and the additional API provided by restify, the code necessary to produce a simple server is probably less than any other language, but it also potentially comes with the cost of being more difficult to maintain in the future. This last point is probably highly subjective and will vary from

developer to developer(s). SlimPHP has documentation that appears at least on par with that of node's, but neither of the two have the vast community support that Jersey does.

5 Performance

In our attempts to test and compare the relative performance offered by each framework, we found that the results were fairly insignificant. The main attribute that we compared between the various implementations was request response times. However, this yielded highly varying results that yielded no obvious conclusions as to which framework actually performed better. In fact, it appeared that all we were really timing was network latency and possibly the extent of our server process's blocking while waiting to write to the disk for its database operations. However, this did at least allow us to reasonably conclude that none of our examined frameworks was objectively worse than the others. Since maximizing performance likely wouldn't be a pressing issue for a newer developer, the best bet in choosing a framework would likely be to choose based on the other factors discussed in this report and not worry about likely small differences in performance that would be seen at the users' application's scale.

6 Industry Opinion

Most of the software developer bloggers agree that Node.js is gaining the most momentum out of all the new frameworks that have been introduced. [3] Large companies such as Walmart, and Paypal have started using it, and in the process Node.js has made Javascript an even more highly marketable language to learn. Node.js also has less context switches, since the entire framework from frontend to database (MongoDB) can be written using Javascript. [3] PHP and Java developers are required to know their target language, and SQL extensively. This allows Node.js developer to produce working environments rather quickly, and with less training.

Paypal who recently just switched to Node.js from java reported that they have built their new infrastructure with less people, a third the lines of code, and 40 percent less files. [2] They have also reported that they can process double the requests in Node versus the Java application. [2] A considerable 35 percent decrease was also noticed in the average response time of users. [2] For a large enterprise such as Paypal, Node.js seems to be the best framework to go with. However the maturity of the framework is still in question, therefore Paypal has not entirely switched over to the Node.js framework. [2]

7 Conclusion

In conclusion when picking the correct Restful framework, you must consider the needs of your service first. If you anticipate a large amount of developers using your service, such as Paypal, then Node.js may be the best choice. If you require a plethora of experienced developers, and a finely tuned framework, then either Jersey, or SlimPHP may be the best choice. In the end we chose to go with Jersey for implementing our Rest framework. The ease of using a programming language we are already proficient with, combined with numerous API guides, and online tutorials made the development process of the RAPDB very smooth. Our Node.js implementation of the server managed to get far, however the asynchronous learning curve made it difficult to design, and to refactor code. The final deciding factor for Java was also the use of Eclipse, which added strong debugging support to the project. The Node.js debugger is relatively new and not as intuitive.

References

[1] R. T. Fielding, "Representational state transfer (rest)," 2000. [Online]. Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Information on REST

[2] J. Harrell, "Node.js at paypal," November 2013. [Online]. Available: https://www.paypal-engineering.com/2013/11/22/node-js-at-paypal/

Article describing Paypal's switch to Node.js from Java

[3] A. Mardan, "Php vs. node.js," August 2013. [Online]. Available: http://webapplog.com/php-vs-node-js/

Differences between Node and PHP

[4] RestApiTutorial.com, "Rest api quick tips." [Online]. Available: http://www.restapitutorial.com/lessons/restquicktips.html

Tips on setting up your Rest framework