Automatically Generated REST Services

(Proposal for UC Davis ECS 193 - Winter/Spring 2018)

Keywords: Microservices, Swagger, OpenAPI, Open Source

Background/Introduction/Business Value:

Developing REST services for modern cloud applications is a problem that affects tech companies from Google down to tiny start-ups. Writing REST at industrial scale is hard: services use different programming languages, often expose complex interfaces, and must change quickly to meet new requirements. Our project will solve this problem by delivering a lightweight code generation tool that enables developers to create initial REST service code easily and change it quickly afterwards.

Existing efforts to make REST development easier provide an obvious starting point. OpenAPI--also known as Swagger--is an industry-supported consortium that offers popular source tools to document REST interface contracts as well as generate server and client code. Unfortunately the code generation tool, Swagger Codegen, is complex, buggy, and hard to change. We will focus on an open source replacement for Swagger Codegen that solves these problems. In addition, we will make it easy to extend code generation to cover as much of the service code as possible--even database schema will be included. The result will be a welcome addition we can offer to the OpenAPI community.

Description/Design Issues/Project scope:

Solution:

A small, Python-based codegen tool that scans OpenAPI 3.0 specifications and generates REST server as well as client code from built-in templates for Java, Python, Golang, and SQL. Ease-of-use is a principal design value--users can download the tool and start generating code within minutes. Anybody who can master simple Python programming can modify code generation templates or create their own within an hour.

User Story:

Alistair and Sue design a REST-based microservice with a command line client using an OpenAPI 3.0 specification. They generate initial server and client code in Python using the codegen built-in templates, then hack in a prototype implementation they can show to users to get feedback. After the demo, they decide to write the client in Java, keep the server in Python, and add SQL schema for PostgreSQL. Sue quickly adapts the Java

templates to generate code for a command-line interpreter that submits REST requests, while Bob tweaks the Python templates to generate a Python/Flask service that adds custom security auditing on each request. Sue also generates PostgreSQL schema to store data in a shared database server used by all micro-services. The microservice interface changes several times over the month or so of development. Each time Bob and Sue update the OpenAPI spec and regenerate the REST boilerplate code and SQL. The project finishes on time and also incorporates lots of great user feedback prior to launch.

Deliverables:

1. Design documents including an analysis of existing OpenAPI codegen tools and alternatives to implement more simply and efficiently.
2. Apache 2.0 licensed Github project for codegen from OpenAPI 3.0 specifications.
    1. Compact Python-based code generation engine.
    2. Templates for a selection of languages including Java, Python, and SQL.
    3. Sample REST application that illustrates how to run codegen as well as develop new templates.
3. Time permitting, develop a website with tutorial documentation and announce the project formally to the OpenAPI community. Extra credit if you publish on pypi.org!

Critical issues:

Designing a Python codegen core that scans specifications and creates code from easy-to-understand templates. If the core is more than 2 to 3,000 lines you are probably doing it wrong. ;)

Giving users control over the codegen process. This is perhaps the hardest issue, because to solve it you need insight into how REST services work across different languages and REST processing frameworks as well as what parts of the code are practical to generate.

Contact:

The sponsors will meet with the students virtually 1-2X per week and once a month at the UC Davis Campus. Guidance will including feedback from testing codegen on industrial REST applications.

Sponsors:

Robert Hodges berkeleybob2105@gmail.com 510-501-3728

(Others TBD)