Lab 4 - Google EndPoint API

What we mean by Google EndPoints?

Google Cloud Endpoints consists of tools, libraries and capabilities that allow you to generate APIs and client libraries from an App Engine application, referred to as an *API backend*, to simplify client access to data from other applications. Endpoints makes it easier to create a web backend for web clients and mobile clients such as Android or Apple's iOS.

What is Public API?

Public APIs are now considered to be the most effective mechanism to integrate two applications. Typically the APIs are hosted in a Server side application and exposed to the clients to consume. The clients could be other server side applications, native mobile applications or even browser applications (mobile or desktop).

When we speak about a Public API, few things are common to them:

- They are available over a popular and well defined protocol, usually HTTP.
- It uses JSON or XML as the data format.

What are the general verbs used in REST API?

Google Cloud Datastore is a NoSQL document database built for automatic scaling, high performance, and ease of application development. Learn more about Google DataStore in: https://cloud.google.com/datastore/docs/concepts/overview

Lab Objectives

- Build a Google app engine using java and javascript/html.
- Learn how to create endpoint and test it.
- Learn how you can communicate between 2 google apps using EndPoint API.

Lab Requirements

- Google App Engine account.
- 2 created projects in your Google app Engine developer console. Note that in Google free trial you can create more than 3 projects. So, you can reuse 2 projects we created in Lab 2.
- In the Lab specification, we assume the following Project Name and project id for these 2 projects, e.g.,
  - My API Project, project id: sxxxxxx-myapi
  - My API Client, project id: sxxxxxx-my-client
- CLI: Terminal (MacOS), Git Bash (Win) or PowerShell (Win)

1 Cloud SDK

a. Make sure that Cloud SDK is authorized to access your data and services on Google Cloud Platform:

```
gcloud auth login
```

b. Use Application Default Credentials:

```
gcloud auth application-default login
```

c. Install the Cloud SDK `app-engine-java` component:

```
gcloud components install app-engine-java
```

d. Update to the latest version of Cloud SDK and all components:

```
gcloud components update
```

2 Create an App Engine app

a. Set the API project as the default project (project id: sxxxxxx-myapi).

```
gcloud config set project [YOUR_PROJECT_ID]
```

Replace `[YOUR_PROJECT_ID]` with your Cloud project ID. If you have other Cloud Platform projects, and you want to use `gcloud` to manage them, see Managing Cloud SDK Configurations.

b. Create an App Engine app using the following command.

Replace `[YOUR_PROJECT_ID]` with your Cloud project ID and `[YOUR_REGION]` with the region that you want the App Engine app created in.

```
gcloud app create --project=[YOUR_PROJECT_ID]
```

3 Getting the sample code

To clone the sample API from GitHub:

a. Clone the sample repository to your local machine:

```
git clone https://github.com/GoogleCloudPlatform/java-docs-samples
```

b. Change to the directory containing the sample code:

```
cd java-docs-samples/appengine-java8/endpoints-v2-backend
```

4 Adding the project ID and version to the sample API Code

You must add the project ID obtained and version when you created your Cloud project to the sample's `pom.xml` before you can deploy the code.

To add the project ID:

a. Edit the file

`java-docs-samples/appengine-java8/endpoints-v2-backend/pom.xml`.

b. Replace YOUR_PROJECT_ID

Search for `<endpoints.project.id>` and `<deploy.projectId>` and replace the value `YOUR_PROJECT_ID` with your Cloud project ID.

For example:

```
<endpoints.project.id>YOUR_PROJECT_ID</endpoints.project.id>
<deploy.projectId>YOUR_PROJECT_ID</deploy.projectId>
```

c. Add PLUGIN.DEPLOY.VERSION property

Add the following property and value under `<deploy.projectId>YOUR_PROJECT_ID </deploy.projectId>` to have Google Cloud generate a version for you.

```
<deploy.version>GCLOUD_CONFIG</deploy.version>
```

d. Save your changes.

5 Building the sample project

To build the project:

a. Make sure you are in the directory
`java-docs-samples/appengine-java8/endpoints-v2-backend`.

b. Install Maven:

You will need to download Maven from Apache website.

c. Invoke the command:

```
mvn clean package
```

d. Wait for the project to build

When the project successfully finishes, you will see a message similar to this one:

```
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
--------
[INFO] Total time: 14.846s
[INFO] Finished at: Wed April 13 09:43:09 PDT 2016
[INFO] Final Memory: 24M/331M
```

e. TroubleShooting

```
error: error reading
/Users/NganHuynh/.m2/repository/com/google/endpoints/endpoints-
framework/2.1.0/endpoints-framework-2.1.0.jar; ZipFile invalid LOC header
(bad signature)

error: error reading
/Users/NganHuynh/.m2/repository/com/fasterxml/jackson/core/jackson-
databind/2.6.4/jackson-databind-2.6.4.jar; invalid CEN header (bad
signature)

error: error reading
/Users/NganHuynh/.m2/repository/com/google/appengine/appengine-api-1.0-
sdk/1.9.64/appengine-api-1.0-sdk-1.9.64.jar; ZipFile invalid LOC header
(bad signature
```

If you see these errors, please go to the directory and delete the whole version directory. For example “2.1.0” for endpoints-framework, “2.6.4” for jackson-databind, “1.9.64” for appengine-api-1.0-sdk

6 Generating the OpenAPI configuration file

You use a tool from the Frameworks library to generate an OpenAPI configuration file called `openapi.json`. This file describes the sample code's REST API.

To generate the OpenAPI configuration file:

a. Invoke the Endpoints Frameworks tool using this command:

```
mvn endpoints-framework:openApiDocs
```

b. Wait for the configuration spec to build.

When it finishes you'll see a message similar to this one:

```
OpenAPI document written to target/openapi-docs/openapi.json
```

Ignore any messages about failure to load a static logger class.

7 Deploying the Endpoints configuration

To deploy the Endpoints configuration, you use Google Service Management, an infrastructure service of Google Cloud Platform that manages other APIs and services, including services created using Cloud Endpoints.

To deploy the configuration file:

a. Make sure you are in the directory

`java-docs-samples/appengine-java8/endpoints-v2-backend`.

b. Deploy the OpenAPI configuration file that was generated in the previous section by invoking the following command:

```
gcloud endpoints services deploy target/openapi-docs/openapi.json
```

Note: Make sure your current project in the Google Cloud console is the project you deploy the Endpoints API.

This creates a new Cloud Endpoints service with the name in the format `YOUR_PROJECT_ID.appspot.com`. This name is configured in `pom.xml` and other configuration files included in the sample. Note that when you deploy the API on App Engine, a DNS record is created using the name format `YOUR_PROJECT_ID.appspot.com`, which is the fully qualified domain name (FQDN) that you use when you send requests to the API.

As it is creating and configuring the service, Service Management outputs a great deal of information to the terminal. You can safely ignore the warnings about the paths in `openapi.json` not requiring an API key. On successful completion, you will see a line like the following that displays the service configuration ID and the service name:

```
Service Configuration [2017-02-13-r2] uploaded for service [example-
project-12345.appspot.com]
```

In the above example, `2017-02-13-r2` is the service configuration ID and `example-project-12345.appspot.com` is the service name.

See gcloud endpoints services deploy in the Cloud SDK Reference documentation for more information.

8 Running the sample locally

After deploying the Endpoints configuration, you can run the sample locally.

a. Set the environment variable

Set the `ENDPOINTS_SERVICE_NAME` environment variable to: `[YOUR_PROJECT_ID].appspot.com`

In Linux or Mac OS:

```
export ENDPOINTS_SERVICE_NAME=[YOUR_PROJECT_ID].appspot.com
```

In Windows PowerShell:

```
$Env:ENDPOINTS_SERVICE_NAME="[YOUR_PROJECT_ID].appspot.com"
```

Replace `[YOUR_PROJECT_ID]` with your Cloud project ID.

b. Acquire new user credentials to use for Application Default Credentials.

```
gcloud auth application-default login
```

c. Run the development server:

```
mvn appengine:run
```

d. Send a request to the local instance

The local instance is reachable on http://localhost:8080/ as indicated by the logs printed by the `mvn appengine:run` command:

```
[INFO] GCLOUD: INFO: Module instance default is running at
http://localhost:8080/
```

To query the local instance:

```
curl \
    --header "Content-Type: application/json" \
    --request POST \
    --data '{"message":"hello world"}' \
    http://localhost:8080/_ah/api/echo/v1/echo
```

If you are on Windows, instead invoke the following PowerShell cmdlet:

```
(Invoke-WebRequest -Method POST -Body '{"message": "hello world"}' `
    -Headers @{"content-type"="application/json"} -URI `
    "http://localhost:8080/_ah/api/echo/v1/echo").Content
```

You should get a 200 response with the following data:

```
{
 "message": "hello world"
}
```

Note: This step can be skipped if an error is returned as it is not quite essential.

9 Deploying the API Backend

So far you have deployed the OpenAPI configuration to Service Management, but you have not yet deployed the code that will serve the API backend. This section walks you through deploying the sample API to App Engine.

To deploy the API backend:

a. Make sure you are in the directory

```
java-docs-samples/appengine-java8/endpoints-v2-backend
```

b. Deploy the API implementation code using Maven:

```
mvn appengine:deploy
```

The first time you upload a sample app, you may be prompted to authorize the deployment. Follow the prompts: when you are presented with a browser window containing a code, copy it to the terminal window.

c. Wait for the upload to finish.

We recommend that you wait a few minutes before sending requests to your API while App Engine completely initializes.

10 Sending a request to the API

After you deploy the API and its configuration file, you can send requests to the API

from your API Client (project id: sxxxxxx-my-client). To send a request to the API:

a. From a command line, invoke the following `curl` command:

```
curl \
    --header "Content-Type: application/json" \
    --request POST \
    --data '{"message":"hello world"}' \
    https://[YOUR_PROJECT_ID].appspot.com/_ah/api/echo/v1/echo
```

If you are on Windows, instead invoke the following PowerShell cmdlet:

```
(Invoke-WebRequest -Method POST -Body '{"message": "hello world"}' `
    -Headers @{"content-type"="application/json"} -URI `
    "https://[YOUR_PROJECT_ID].appspot.com/_ah/api/echo/v1/echo").Cont
ent
```

Replace `[YOUR_PROJECT_ID]` with your Cloud project ID.

You should get a 200 response with the following data

```
{
 "message": "hello world"
}
```