

Spring Boot based REST API to Improve Data Quality Report Generation for Big Scientific Data: ARM Data Center Example

Kavya Guntupally
Environmental Sciences Division
Oak Ridge National Laboratory
 Oak Ridge, TN
 guntupallyk@ornl.gov

Ranjeet Devarakonda
Environmental Sciences Division
Oak Ridge National Laboratory
 Oak Ridge, TN
 devarakondar@ornl.gov

Kenneth Kehoe
University of Oklahoma
 Norman, OK
 kkehoe@ou.edu

Abstract— Web application technologies are growing rapidly with continuous innovation and improvements. This paper focuses on the popular Spring Boot [1] java-based framework for building web and enterprise applications and how it provides the flexibility for service-oriented architecture (SOA). One challenge with any Spring-based applications is its level of complexity with configurations. Spring Boot makes it easy to create and deploy stand-alone, production-grade Spring applications with very little Spring configuration. Example, if we consider Spring Model-View-Controller (MVC) framework [2], we need to configure dispatcher servlet, web jars, a view resolver, and component scan among other things. To solve this, Spring Boot provides several *Auto Configuration* options to setup the application with any needed dependencies. Another challenge is to identify the framework dependencies and associated library versions required to develop a web application. Spring Boot offers *simpler dependency management* by using a comprehensive, but flexible, framework and the associated libraries in one single dependency, which provides all the Spring related technology that you need for starter projects as compared to CRUD web applications. This framework provides a range of additional features that are common across many projects such as embedded server, security, metrics, health checks, and externalized configuration. Web applications are generally packaged as war and deployed to a web server, but Spring Boot application can be packaged either as war or jar file, which allows to run the application without the need to install and/or configure on the application server. In this paper, we discuss how Atmospheric Radiation Measurement (ARM) Data Center (ADC) at Oak Ridge National Laboratory, is using Spring Boot to create a SOA based REST [4] service API, that bridges the gap between frontend user interfaces and backend database. Using this REST service API, ARM scientists are now able to submit reports via a user form or a command line interface, which captures the same data quality or other important information about ARM data.

Index Terms— *auto configuration, CRUD, java framework, service-oriented architecture, REST, spring boot*

I. INTRODUCTION

The ADC currently uses a SOA based framework to develop any new applications and/or tools. One such application is the Data Quality Report (DQR) submission tool. The DQR tool is used by the Data Quality (DQ) Office to create report(s) about the quality of data within a particular data stream for a specific time range accompanied by a category of severity for the quality issue affecting the data. This tool allows the user to submit a report with required fields through the REST API, using Spring Boot, which saves the data to database. The API acts as a middle layer between the client User Interface (UI) and the database.

REST is an architectural style to design web services, that can be consumed from different clients using simple HTTP calls. Spring Boot makes developing the REST API service with minimum fuss. The API structure contains a Spring Boot Application class (acts as a launching point), a Java Persistence API (JPA) entity (Spring Boot Auto Configuration detects the database that is being used and entities to match the targeted tables), a controller (that handles all the services the client requires) and a repository (that is based on the CrudRepository [3] to perform database create, read, update, and/or delete [CRUD] operations).

II. IMPLEMENTATION

The application is designed using three main components (as shown in figure), a frontend UI form (to capture user entered data), API (which accepts HTTP/S requests either from a UI form or command call, like curl or Postman) and the database (where the data is stored). The UI is a simple form with a set of

fields that describes the data quality issue for the affected data. Once the form is submitted, the data is validated (check for required fields and expected formats). If valid, then it is sent as JavaScript Object Notation (JSON) request to the REST API using a resource URL.



Figure 1. Web service Architecture

The API validates the incoming request against the entity fields required to save, update, or delete the data in the database. If the data is valid and the operation is successful, the API returns a HTTP response back to UI with success message and standard status code [5] (2xx – *Success*). If invalid data or failed operation, return a failure message with exception encountered and corresponding standard status code [5] (4xx – client error or 5xx – server error). The API is the most important component, it is designed in such a way that it not only accepts HTTP request from the UI (i.e. form data), but also has the flexibility to accept JSON HTTP request through command line interface, which means the UI is not the only source to submit data. This allows ARM scientists to setup automated scripts to create DQ reports without being restricted to only using the UI form.

III. CONCLUSION

In conclusion, this paper highlights the SOA based REST API using Spring Boot Framework which does have definite advantages over other spring-based frameworks. It allows simpler dependency management, auto configuration, built-in CRUD handling, and flexible project startup (installed or local). In addition, we have discussed how we are using it to allow our scientists to submit reports using frontend UI and command line interfaces.

ACKNOWLEDGEMENT

The ARM is funded through the DOE Office of Science and is managed through the Biological and Environmental Research (BER) Division. Oak Ridge National Laboratory is managed by the UT-Battelle, LLC, for the U.S. Department of Energy under contract DE-AC05-000R22725.

REFERENCES

[1] “Spring Boot Reference Guide.” *Spring Boot Reference Guide*, docs.spring.io/spring-boot/docs/current/reference/htmlsingle.

[2] “Web on Servlet Stack.” *Web on Servlet Stack*, docs.spring.io/spring/docs/current/spring-framework-reference/web.html.

[3] “Working with Spring Data Repositories.” *1. Working with Spring Data Repositories*, docs.spring.io/spring-data/data-commons/docs/1.6.1.RELEASE/reference/html/repositories.html.

[4] “Representational State Transfer.” *Wikipedia*, Wikimedia Foundation, 16 Oct. 2018, en.wikipedia.org/wiki/Representational_state_transfer.

[5] “List of HTTP Status Codes.” *Wikipedia*, Wikimedia Foundation, 20 Oct. 2018, en.wikipedia.org/wiki/List_of_HTTP_status_codes.