International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-7, Issue-3, May-2019
DOI: 10.21276/ijircst.2019.7.3.14

Rapid Web Application Development Using Spring Framework: A Case Study

Ravichandra Wali, Prof. Suresh Kumar M

Abstract **- In the agile world, the web application development has to be accurate and economical. One of the best ways to achieve the productivity is by reducing the complexity. The complexity can be reduced by the separation of concerns. If there is a reduction of concerns of the configurations and the boilerplate code. One approach is to use spring framework for development. It provides a decoupled way of developing web applications. Web application development becomes simple in spring with its various features. The underlying spring architecture and a case study using spring have been presented in the paper [1].**

Keywords **– Spring Framework, Java, REST Services, Python**

I. INTRODUCTION

Spring is a popular framework for application development in Java. It provides many features to simplify the development and "Inversion of Control" (Dependency Injection) is one of the best features. It allows the development of applications with loose coupling and the loosely coupled applications are best for unit testing and reuse [2].

One challenge with the applications developed using spring framework is its configurations complexity. To reduce this complexity, Spring Boot makes development and deployment of stand-alone applications, backend services as easy as possible [7]. It also helps to overcome the challenge of finding the right framework dependencies and library versions needed to develop an application, by providing a simple dependency management. It also provides a wide range of features like embedded server, security, externalized configuration, since spring framework focuses on Aspect Oriented Programming more than Object Oriented Programming.

Manuscript received May 22, 2019

Ravichandra Wali, Dayananda Sagar College of Engineering, Dept of ISE (e-mail: raviwali3@gmail.com).

Prof. Suresh Kumar M, Dayananda Sagar College of Engineering, Dept of ISE

For instance, if security, logging etc are needed in the project, then AOP can be used, the developer can focus of business logic and the spring framework handles everything else. Any of these actions can be easily performed before or after a method call, after the methods returns or throws an exception. The architecture chosen for application development plays an important role in productivity. The architecture should be simple and meet the requirements of an Enterprise application. The three fundamental blocks of any J2EE application are the User Interface layer, the services layer (business logic) and the Data access layer.

II. ARCHITECTURE

Spring can use other frameworks such as Struts, Web Work etc as its architecture. Lightweight container architecture can also be used which integrates easily with business objects. The Plain Old Java Objects (POJO) represents these business objects in the lightweight container.

Another popular architecture is Micro service architecture. The micro services focus on a single business logic which is common and can be reused in the backend. There is a tool called Swagger, which simplifies the task of developers to view and analyze the micro services at one place. The developer does not have to go through any program source code, application documents or the configurations. The controller classes created in spring boot will get automatically mapped by the Swagger to their micro services [3]. The methods in the controllers can be made REST [6] API. The GET, POST, DELETE and PUT functionalities of REST API can be accessed using Swagger. The REST API serves the requirements of the front-end application. Spring also provides embedded Tomcat server which starts to run with our application on port 8080

III. METHODOLOGY

In this paper, it is shown how a simple web service can be created that executes python scripts when a user makes http request. Conventionally, the application structure consists of the starting point, controller, entity, service and repository [4].

The project is created by clicking

1. File > New > Maven Project

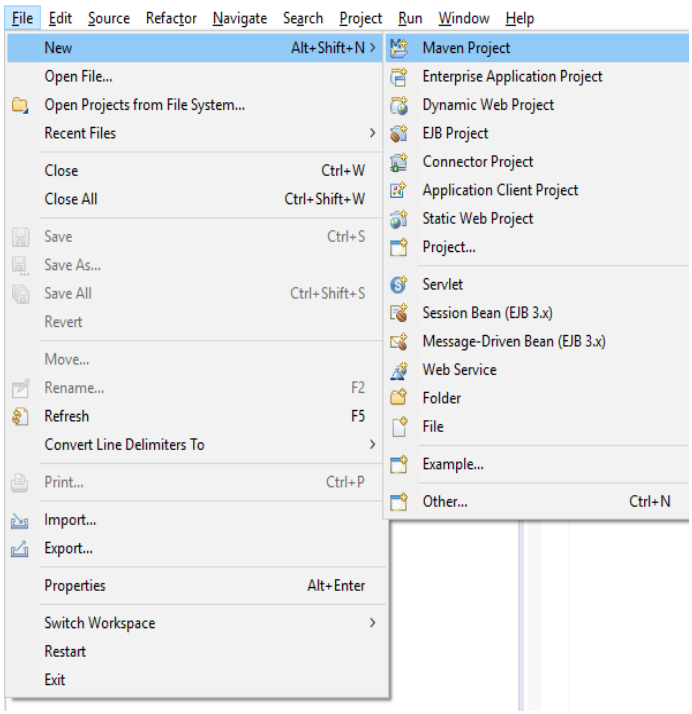

Fig.1: File menu

2. Select the archetype as quick start

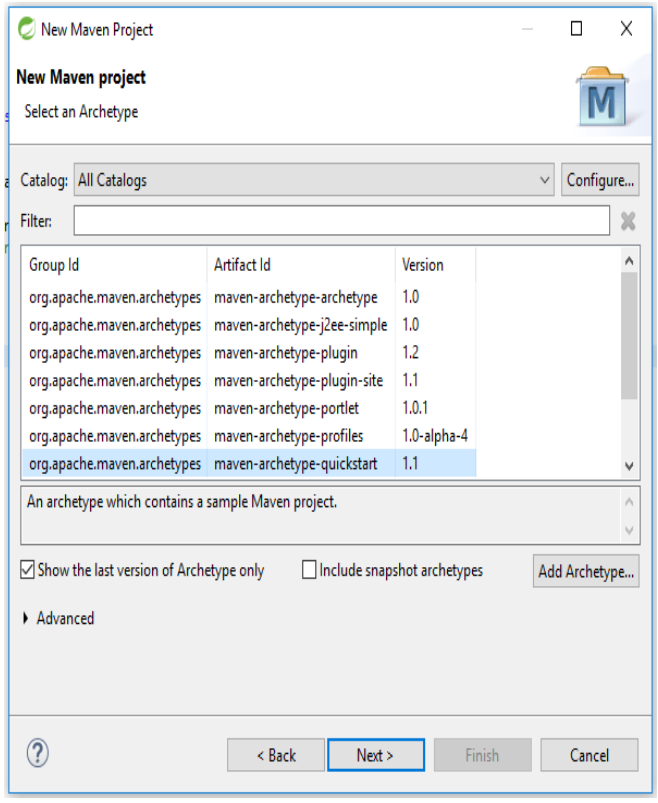

Fig. 2: Maven project menu

3. Choose the archetype parameters

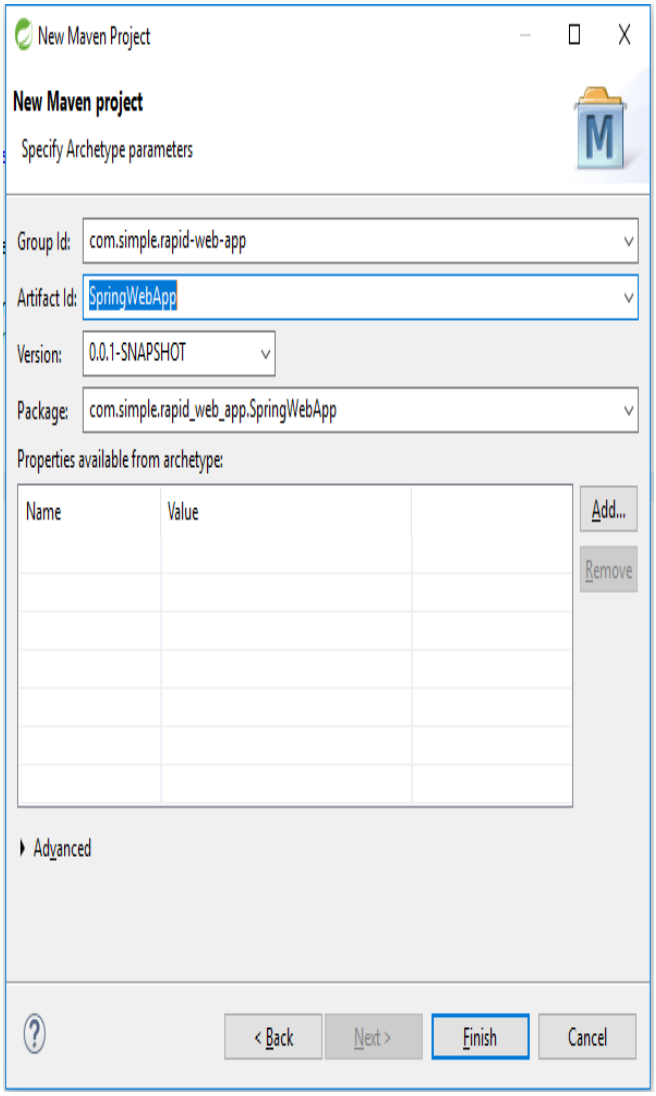

Fig. 3: Archetype menu

4. Add the spring-boot-starter-parent parent tag and spring-boot-starter-web dependency in pom.xml

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.4.2.RELEASE</version></parent>
  <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-
  web</artifactId>
  </dependency>
```

5. Edit build configurations by setting the goals as "clean install"

International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-7, Issue-3, May-2019
DOI: 10.21276/ijircst.2019.7.3.14

Edit configuration and launch.

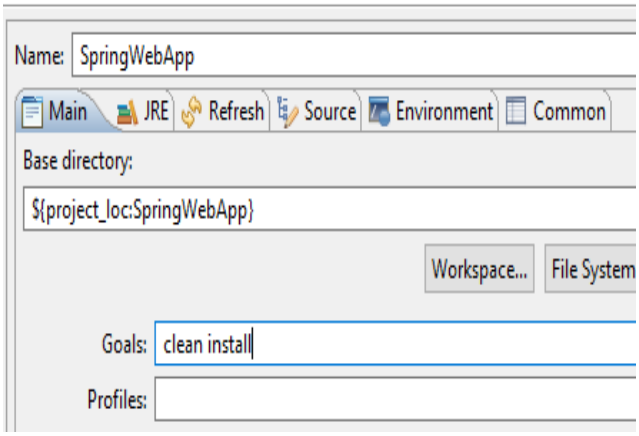

Fig. 4: Project build menu

The boilerplate code is generated by spring boot and gives the starting point for the application in the file App.java that can be used to run the application. Use the @Spring Boot Application annotation before the class definition. This enables the 3 features- auto configuration, component scan and extra configuration. Make a resources directory and add the python files to be executed in it. Create a service package and create a java class that contains a method to run the python script (.py file) using command line interface. The instance of Process Builder is used to run commands. The service is class is annotated with @Service to denote that the class provides a specific service.

The service classes are used to communicate with the database, run a process on a separate thread, perform business logic, etc. The POJO[5] classes are all created in entity package and are annotated with @Repository. They only contain properties, constructors, getters and setters but no business logic.Create a controller packager with the annotation @RestController.

Now with the @RequestMapping at the method definition, we can provide the GET, POST, PUT, DELETE functionalities. In our case, @ResquestMapping("/runPython") is the annotation and the method can be called using localhost:8080/runPython. GET is the default functionality. The service class that executes python scripts is autowired in the controller using @Autowired. The beans can also be configured using XML[8] but springboot automatically gets the job done. This annotation dynamically instantiates the variables and provides loose coupling.

The controllers are used to provide the functionality at the backend needed by the front end (client side). The four functionalities are implemented as shown below and "plots" (graph object) is the POJO with id, name, yAxis amd xAxis as its attributes-

1. **PUT**

   It uses @Request Mapping (value = "/plots/{id}", method=Request Method. PUT) as annotation and the method takes the parameters (@Path Variable ("id") String id, @Request Body Plot plot)

   The plot object can be added to the database or to a static repository variable.

2. **POST**

   It uses @Request Mapping (value = "/plots", method=RequestMethod.POST) as annotation and the method takes the parameters (@Request Body Plot[] plots)

   The operations can be performed on the array of plot objects.

3. **GET**

   It uses @Request Mapping (value = "/plots", method=Request Method. GET) as annotation and the method does not need any parameters. It can fetch data from the database or the repository and return in JSON format with the Response Entity object.

4. **DELETE**

   It uses @Request Mapping (value = "/plots/{id}", method=Request Method. DELETE) as annotation and the method takes the parameters (@Path Variable ("id") String id). It deletes the plot object that matches the id.

The corresponding Response Entity can be returned at the end of each method. All methods update the existing data except POST, which creates new data.

IV. CONCLUSION

Spring framework is an efficient framework oriented towards development of commercial applications using Java. Spring boot makes it easier for developers to get everything out of the box and focus only on the business logic. Due to the embedded server and loose coupling among the classes, it is easy for unit testing and module testing. The division of an application as Model View Controller and Dependency Injection simplifies the task of the developer.

V. ACKNOWLEDGEMENT

I would like to thank Mr. Suresh Kumar M, Assistant Professor in the Department of ISE, Dayananda Sagar College of Engineering.

REFERENCES

[1] Mrs. Rasika Khandal, Meenal Meshram, Shubham Mahatme, "Study of Spring Framework", International Conference On Emanations in Modern Engineering Science and Management (ICEMESM-2017)

[2] "Spring Boot Reference Guide." Spring Boot Reference Guide, docs.spring.io/spring-boot/docs/current/reference/htmlsingle.

[3] Chunsheng Zhao, Mai Jiang, Zhiyong He," The Design of E-Commerce System Architecture Based on Struts2, Spring and Hibernate", IEEE Transaction Paper Dated 2010.

[4] Rod Johnsonet," Professional Java Development with the Spring Framework", Publications John Wiley & Sons 2005.

[5] Praveen Gupta, Prof. M.C. Govil, "Spring Web MVC Framework for rapid open source J2EE application development: a case study", International Journal of Engineering Science and Technology, Vol. 2(6), 2010, 1684-1689.

[6] "Representational State Transfer." Wikipedia, Wikimedia Foundation, 16 Oct. 2018, en. wikipedia.org/wiki/Representational_state_transfer.

[7] Rod Johnsonet, "Professional Java Development with the Spring Framework", Publications John Wiley & Sons 2005.

[8] Wang HaiTao, Jia BaoXian, "Research Based on Web Development of Spring Integration Framework", 2010 International Forum on Information Technology and Applications

