



RV Educational Institutions®  
RV College of Engineering®

Autonomous  
Institution Affiliated  
to Visvesvaraya  
Technological  
University, Belagavi

Approved by AICTE,  
New Delhi

*Go, change the world*

**Technical Seminar Report**  
**18MCA62**  
**on**  
**“Building An Application Using Spring Boot Framework”**

**Submitted by**  
**Chaitra B V**  
**1RV19MCA21**

**Under the Guidance**  
**of**

**Dr. S Anupama Kumar**

**Associate Professor**

**Department of MCA**

**RV College of Engineering®**

**Bengaluru-59**

*Submitted in partial fulfillment of the requirements for the award of degree  
of*

**MASTER OF COMPUTER APPLICATIONS**  
**2022**

# RV COLLEGE OF ENGINEERING®

(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)

## DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

Bengaluru– 560059



### CERTIFICATE

Certified that the seminar titled **“Build An Application Using Spring Boot Framework”** carried out by **Chaitra B V, USN: 1RV19MCA21**, a bonafide student of **RV College of Engineering®, Bengaluru** submitted in partial fulfillment for the award of **Master of Computer Applications** of **RV College of Engineering®, Bengaluru** affiliated to **Visvesvaraya Technological University, Belagavi** during the year **2022**. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the departmental library. The report has been approved as it satisfies the academic requirement in respect of the technical seminar prescribed for the said degree.

<b>Dr. S Anupama Kumar</b> Associate Professor Department of MCA RVCE, Bengaluru –59		<b>Dr. Andhe Dharani</b> Professor and Director Department of MCA RVCE, Bengaluru–59
---	--	---

# **RV COLLEGE OF ENGINEERING®**

(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)

## **DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS**

**Bengaluru– 560059**

### **DECLARATION**

I, **Chaitra B V**, student of Sixth Semester MCA, **Department of Master of Computer Applications**, RV College of Engineering®, Bengaluru declare that the Seminar titled **“Build An Application Using Spring Boot Framework”** has been carried out by me. It has been submitted in partial fulfillment of the course requirements of Seminar-II 18MCA62 for the award of degree in **Master of Computer Applications** of RV College of Engineering®, Bengaluru affiliated to Visvesvaraya Technological University, Belagavi during the academic year **2022**. The matter embodied in this report has not been submitted to any other university or institution for the award of any other degree or diploma.

**Date of Submission:**

**Signature of the Student**

Student Name: Chaitra B V

USN: 1RV19MCA21

Department of Master of Computer  
Applications

RV College of Engineering®

Bengaluru-560059.

## **ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompany the success of any work would be incomplete unless I mention the name of the people, who made it possible, whose constant guidance and encouragement served a beacon light and served our effort with success.

I express my wholehearted gratitude to Dr. **Subramanya K N**, Principal, RV College of Engineering® for providing me an opportunity.

I express my special thanks to **Dr. Andhe Dharani**, Professor, and Director, Department of MCA, RV College of Engineering®, Bengaluru for her constant support and guidance.

I express my sincere thanks and wholehearted credit to my Internal guide **Dr. S Anupama Kumar**, Associate Professor, Department of MCA, RV College of Engineering®, Bengaluru for her constant encouragement, support, and guidance during the project work.

I am also thankful to lab in-charge staff and all faculty of the department for their help and support during the seminar. On a moral personal note, my deepest appreciation and gratitude to my beloved family, who have been a fountain of inspiration and have provided unrelenting encouragement and support.

On a moral personal note, my deepest appreciation and gratitude to my beloved family, who have been a fountain of inspiration and have provided unrelenting encouragement and support.

**Chaitra B V**

Department of MCA  
RV College of Engineering®  
Bengaluru-59

## ABSTRACT

Spring Boot is an open-source Java-based framework used to create a Micro Service. It is developed by Pivotal Team. It is easy to create stand-alone and production ready spring applications using Spring Boot. Spring Boot contains a comprehensive infrastructure support for developing a micro service and enables users to develop enterprise-ready applications that can “just run”. It can be started with minimum configurations without the need for an entire Spring configuration setup.

Spring boot provides a flexible way to configure Java Beans, XML configurations, and Database Transactions. It provides a powerful batch processing and manages REST endpoints. Handling dependency management is a difficult task for big projects. Spring Boot resolves this problem by providing a set of dependencies for developer’s convenience. Spring boot starter includes the web starter, test starter, data JPA starter, mail starter etc. Building An Application using Spring Boot Framework includes four steps which is Creation of a new Maven Project in any IDE, Adding the Camunda & Spring framework dependencies, Adding the web.xml file for bootstrapping the Spring container and adding a Spring Java configuration to set up the application context

Thus, using spring boot and its starters helps in fast and easy development of spring-based applications and no need for the deployment of war files. Spring boot provides the ability to create standalone applications and the spring boot starters used in production-ready, tested & supported dependency configurations and helps to decrease the overall configuration time for the project.

## Table of Contents

<b>PARTICULARS</b>	<b>PAGE NO.</b>
Title page	i
College Certificate	Ii
Declaration by student	iii
Acknowledgement	iii
Abstract	iv
Table of Contents	vi
List of Tables	vii
List of Figures	vii
<b>Chapter 1: Introduction</b>	<b>1-9</b>
1.1 Introduction to the Seminar Title	1
1.2 Description of the Seminar Concept	4
1.3 Applications of Concept	5
1.4 Architecture Diagram	9
<b>Chapter 2: Literature Review</b>	<b>10-12</b>
2.1 Literature Survey	10
2.2 Summary of the literature Survey	12
<b>Chapter 3: Technical Significance</b>	<b>13-41</b>
3.1 Technological Developments	13
3.2 Tools and Technologies	16
3.3 Sustainability and Societal Concern	37
3.3 Conclusion	41
<b>Bibliography</b>	<b>42-43</b>

## List of Tables

TABLE NO.	PARTICULARS	PAGE NO.
1.1	Acronym and Abbreviation	3

## List of Figures

FIGURE NO.	PARTICULARS	PAGE NO.
1.1	Diagram Of Spring Core	3
1.2	Architecture Diagram of Spring Boot	9
3.1	Spring Boot Features	15
3.2	Directory structure of Spring boot framework	25
3.3	File structure of Spring Boot Framework	27
3.4	Code of pom.xml	28
3.5	Code of Java Main Application	29
3.6	UI file Directory	30
3.7	HTML code	31
3.8	Javascript code	32
3.9	CSS3 code	33
3.10	Final CSS3 code	34
3.11	UI of calculator	35
3.12	UI of calculator	35
3.13	UI of calculator	36

---

## Chapter 1: Introduction

**This chapter gives brief introduction to spring boot framework**

### 1.1 Introduction to Spring Boot Framework

Spring Boot is an open Java-based framework used to create a micro Service. It is developed by Pivotal Team and is used to build stand-alone and production ready spring applications. It is easy to create stand-alone and production ready spring applications using Spring Boot. Spring Boot contains infrastructure support for developing a micro service and enables developers to develop enterprise-ready applications that can be just run.

Spring Boot provides several Auto Configuration options to set up the application with any needed dependencies. 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 users 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. 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

Spring Boot is just an extension of the already existing and expansive Spring frameworks, but it has some specific features that make the application easier for working within the developer ecosystem. That extension includes pre-configurable web starter kits that help facilitate the responsibilities of an application server that are required for other Spring projects. The Spring Boot framework creates a fully production-ready environment that is completely configurable using its prebuilt code within its codebase. The microservice architecture provides developers with a fully enclosed application, including embedded application servers.

The Spring framework provides a rich ecosystem of projects to address modern application needs, like security, simplified access to relational and NoSQL datastores, batch processing, integration with social networking sites, large volume of data streams processing, etc. As Spring is a very flexible and customizable framework, there are usually multiple ways to

---



configure the application. Spring Boot addresses this "Spring applications need complex configuration problems by using its powerful autoconfiguration mechanism.

The main goal of Spring Boot is to quickly create Spring-based applications without requiring the developers to write the same boilerplate configuration again and again.

In recent years, the microservices architecture has become the preferred architecture style for building complex enterprise applications. Spring Boot is a great choice for building microservices-based applications using various Spring Cloud modules.

Spring Boot offers a fast way to build applications. It looks at the classpath and at the beans users have configured, makes reasonable assumptions about what users are missing, and adds those items. With Spring Boot, users can focus more on business features and less on infrastructure.

## ACRONYM AND ABBREVIATION:

CRUD	Create, Read, Update, Delete
RMI	Remote Method Invocation
AMQP	Advanced Message Queuing Protocol
SpEL	Spring Expression Language
MVC	Model-View-Controller
HTTP	HyperText Transfer Protocol
XML	Extensible Markup Language
JDBC	Java Database Connectivity
ORM	Object-relational mapping
CLI	Command Line Interface
IDEs	Integrated Development Environment
STS	Spring Tool Suite
HTML	HyperText Markup Language
CSS	Cascading Style Sheets

Table 1.1: Acronym And Abbreviation:

## 1.2 Description of Spring Boot Framework:

Microservices are small functions where each function of the application operates as an independent service. This architecture allows for each service to scale or update without disrupting other services in the application. It allows the developers to develop and deploy services independently. Each service running has its own process.

Spring Boot Starters are dependency provides production-ready features to help developers to monitor and manage their application. The starters provide the required dependencies needed for the build according to the type of project chosen. For example, a Web project has certain dependencies that may be included simply by stating a “Web project.”

As capable and comprehensive as Spring Framework is, it still requires significant time and knowledge to configure, set up, and deploy Spring applications. Spring Boot mitigates this effort with three important capabilities.

### Autoconfiguration

Autoconfiguration means that applications are initialized with pre-set dependencies that users don't have to configure manually. As Java Spring Boot comes with built-in autoconfiguration capabilities, it automatically configures both the underlying Spring Framework and third-party packages based on the settings (and based on best practices, which helps avoid errors). Even though users can override these defaults once the initialization is complete, Java Spring Boot's auto configuration feature enables users to start developing the Spring-based applications fast and reduces the possibility of human errors.

### Opinionated approach

Spring Boot uses an opinionated approach to adding and configuring starter dependencies, based on the needs of the project. Following its own judgment, Spring Boot chooses which packages to install and which default values to use, rather than requiring users to make all those decisions and set up everything manually.

Users can define the needs of the project during the initialization process, during which users choose among multiple starter dependencies—called Spring Starters—that cover typical use cases. Users run Spring Boot Initializr by filling out a simple web form, without any coding.

For example, the ‘Spring Web’ starter dependency allows users to build Spring-based web applications with minimal configuration by adding all the necessary dependencies—such as the Apache Tomcat web server—to the project. ‘Spring Security’ is another popular starter dependency that automatically adds authentication and access-control features to the application. Spring Boot includes over 50 Spring Starters, and many more third-party starters are available.

### Standalone applications

Spring Boot helps developers create applications that just run. Specifically, it lets users create standalone applications that run on their own, without relying on an external web server, by embedding a web server such as Tomcat or Netty into the app during the initialization process. As a result, users can launch the application on any platform by simply hitting the Run command. (Users can opt out of this feature to build applications without an embedded Web server.)

## 1.3 Applications of Spring Boot :

Spring Framework offers a dependency injection feature that lets objects define their own dependencies that the Spring container later injects into them. This enables developers to create modular applications consisting of loosely coupled components that are ideal for microservices and distributed network applications.

Spring Framework also offers built-in support for typical tasks that an application needs to perform, such as data binding, type conversion, validation, exception handling, REST and event management, internationalization, and more. It integrates with various Java EE technologies such as RMI (Remote Method Invocation), AMQP (Advanced Message Queuing Protocol), Java Web Services, and others. In sum, Spring Framework provides developers with all the tools and features they need to create loosely coupled, cross-platform Java EE applications that run in any environment.

---

Spring Boot is a Framework from the Spring Team to ease the bootstrapping and development of new Spring Applications. Spring Boot helps users to accelerate and facilitate application development.

Spring Boot reduces lots of development time and increases productivity.

When the software development domain extended, and more and more applications started to be published on the market, it became easier to observe that many of these apps had similar requirements. Let's name a few of them:

Logging error, warning, and info messages happen in every app. Most applications use transactions to process data changes. Transactions represent an important mechanism that takes care of data consistency. Most applications use protection mechanisms against the same common vulnerabilities. Most applications use similar ways to communicate with each other. Most applications use similar mechanisms to improve their performance, like caching or data compression. And the list continues. It turns out actually that the business logic code implemented in an app is significantly smaller than the wheels and belts that make the engine of the application (also often referred to as “the plumbing”).

When developers say “business logic code,” developers refer to the code that implements the business requirements of the application. This code is what implements the user's expectations in an application. For example, “clicking on a specific link will generate an invoice.” is something the users expect to happen. Some code of the application users develop implements this functionality, and this part of code is what developers call the business logic code. However, besides implementing the business logic, any app takes care of lots more aspects: security, logging, data consistency, and so on

Moreover, the business logic code is what makes an application different from another from the functionality point of view. If users take two different apps, say a ridesharing system and a social networking app, they have different use cases.

when developers refer to the Spring framework, they refer to a part of the software capabilities that includes:

**Spring Core** – is one of the fundamental parts of Spring that includes foundational capabilities. One of these features users find within Spring Core is the Spring context. The Spring context is a fundamental capability of the Spring framework that enables Spring to manage instances of your app. Also, as part of Spring Core, users find the Spring aspects functionality. Aspects help Spring intercept and manipulate methods users define in your app. The Spring Expression Language (SpEL) is another capability users will find as part of Spring Core, which enables users to describe configurations for Spring using a specific language. All of these are new notions, and developers don't expect users to know them yet. But shortly, users will understand that Spring Core holds the mechanisms that Spring uses to integrate into your app.

**Spring Data Access** – is also one of the fundamental parts of Spring. It provides basic tools users can use to connect to SQL databases to implement the persistence layer of your app.

**Spring Model-View-Controller (MVC)** – is the part of the Spring framework that enables users to develop web applications that serve HTTP requests.

**Spring Testing** – is the part holding the tools users need to write tests for your Spring application. Users can initially imagine Spring Framework as a solar system, where Spring Core represents the star in the middle, which holds all the framework together

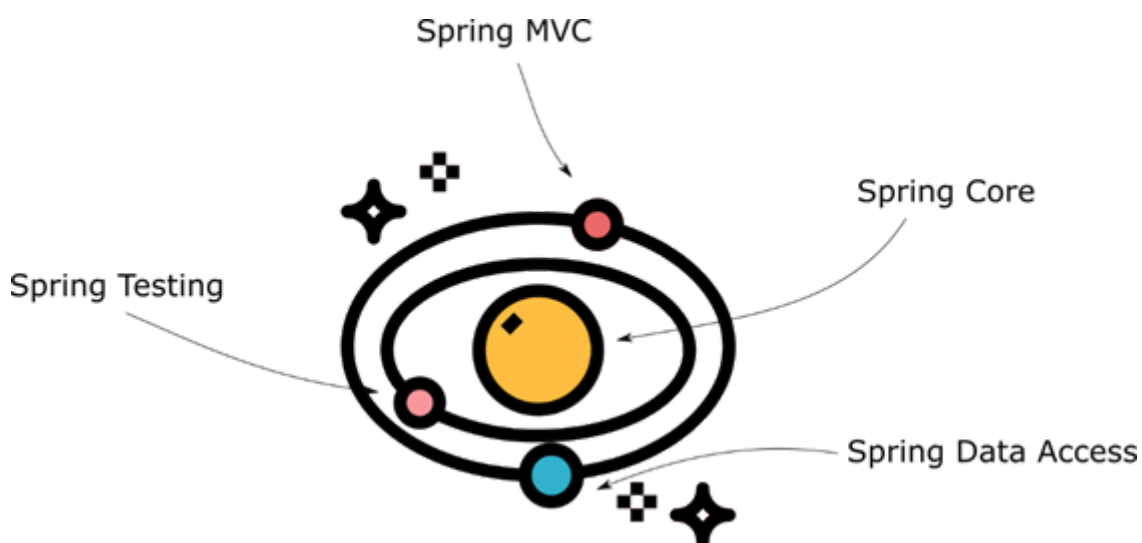


Fig 1.1 Diagram of spring core

Other application of Spring Boot and its starters are:

- It is very easy to develop Spring Based applications with Java or Groovy.
- It reduces lots of development time and increases productivity.
- It avoids writing lots of boilerplate Code, Annotations and XML Configuration.
- It is very easy to integrate Spring Boot Application with its Spring Ecosystem like Spring JDBC, Spring ORM, Spring Data, Spring Security etc.
- It follows “Opinionated Defaults Configuration” Approach to reduce Developer effort
- It provides Embedded HTTP servers like Tomcat, Jetty etc. to develop and test the web applications very easily.
- It provides CLI (Command Line Interface) tool to develop and test Spring Boot(Java or Groovy) Applications from command prompt very easily and quickly.
- It provides lots of plugins to develop and test Spring Boot Applications very easily using Build Tools like Maven and Gradle
- It provides lots of plugins to work with embedded and in-memory Databases very easily.

## 1.4 Architecture Diagram of Spring Boot:

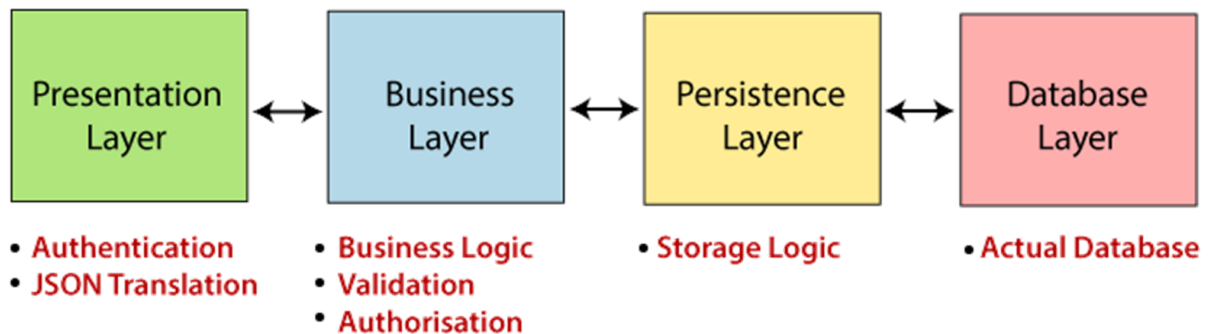


Fig 1.2 Architecture Diagram of Spring Boot

1. **Presentation/Display Layer:** It's the front layer or upper layer of this structure, as it is made up of viewpoints. It's used to interpret the JSON areas to items and vice-versa and handles authentication and HTTP requests. After finishing the authentication, it moves into the business layer for additional processes.
2. **Business Layer:** It manages all of the business logic and performs validation and consent since it's part of business logic. By way of instance, only admins are permitted to alter the consumer's account.
3. **Persistence Layer:** It comprises all of the storage logic, such as database questions of this program. Additionally, it translates the company items from and to database rows.
4. **Database Layer:** It could include many databases. The execution of the aforementioned layered structure is performed in this way: The HTTP request or internet requests are managed by the Controllers from the demonstration layer, the providers control the company logic, as well as the repositories handle persistence (storage logic). A control can manage numerous providers, a service may manage many repositories, and also a repository may manage many databases



---

## Chapter 2: Literature Review

**The purpose of a literature review is to gain an understanding of the existing research and debates relevant to spring boot Framework**

### 2.1 Literature Survey

JIBRIL ADAMU in “Descriptive Analysis of Built in Security Features in Web Development Frameworks” [1] speaks about an important form of Spring application that provides two attack protection mechanisms: The Synchronizer Token Pattern and the Same Site Attribute. Spring Security offers built-in users authentication support

Theofanis Vassiliou, Gioles in “Quality Assurance of Microservices - When to trust the micro-service test results?”[2] speak about Microservice architecture that has become a standard software architecture style, loosely coupled. Spring boot provides facilities in unit testing

Zhe Wang, Feng Tang in “Design and Implementation of a Health Status Reporting System Based on Spring Boot”[3] speaks about the Spring Boot and JPA frameworks which are used on the server side to simplify development steps and quickly carry out background development.

The automatic configuration function provided by it greatly improves the efficiency of background development.

Ashraful Haque, Rasheq Rahman in “Microservice-based Architecture of a Software as a Service (SaaS) Building Energy Management Platform”[4] presents a microservice based approach to building energy management systems that can be used to provide multiple services to end users. A working front end interface has been developed using the spring staters and microservice based approach

Mykhajlo Klymash, Ihor Tchaikovskiy, Olena Hordiichuk-Bublivska, Yulia in “Research of Microservices Features in Information Systems Using Spring Boot”[5] proposes the article which considers an example of creating a microservice using Spring Boot. This method allows users to quickly deploy the microservice and perform data processing.

Yuan Zuo , Yulei Wu , Senior Member, IEEE, Geyong Min , Chengqiang Huang, and Ke Pei in “An Intelligent Anomaly Detection Scheme for Microservices Architectures With Temporal and Spatial Data Analysis”[6] proposes a general-purpose anomaly detection framework targeting at microservices architectures. The framework is comprised of informative data representation, temporal logs modelling and temporal-spatial joint analysis

Liu xuchen, Li chaoyu in “Design and Implementation of a Spring Boot-Based Data Collection System ”[7] speaks about the basic features and concepts of the Spring Boot framework,

Author also speaks about the design of each functional module of the data collection system based on Spring Boot, and the specific implementation method

Michal Gajewski, Michal Gajewski in “Analysis and Comparison of the Spring Framework and Play Framework Performance, Used to Create Web Applications in Java”[8] speaks about the Service-Based Application which may create a solution that is suitable for micro finance. The system uses the spring starters for computing services

Du Ying-kui ,Guan Ping in “Cloud Data Monitoring Management and Visual Application System Based on Spring Boot”[9] proposes a system which develops cloud-based data monitoring management and visualization applications based on Spring Boot. The spring boot microservice in Cloud facilitates users to remotely access web projects. The combination of Spring Boot framework and AJAX has achieved complete separation of front and backend

Yasith Jayawardana, Randil Fernando, Gavindya Jayawardena, Dileka Weerasooriya, Indika Perera in “A Full Stack Microservices Framework with Business Modelling” [10] presents a novel approach which uses boilerplate code compliant with microservices architecture. Author also presents Mstack a full-stack framework for creating business process pipelines using microservices architecture

## 2.2 Summary of the literature Survey:

1. Spring application helps in attack protection mechanisms
2. Spring Security offers built-in users authentication support
3. Spring boot provides facilities in unit testing
4. The automatic configuration function provided by spring starters greatly improves the efficiency of application development
5. Creating a microservice using Spring Boot allows users to quickly deploy the microservice and perform data processing.
6. The spring boot microservice in Cloud facilitates users to remotely access web projects.
7. The combination of Spring Boot framework and AJAX has achieved complete separation of front and backend

## Chapter 3: Technical Significance

### 3.1 Technological Developments:

Spring Boot allows easy setup of standalone Spring-based applications. It's ideal for pulling up new microservices and easy to deploy. It also makes data access less of a pain, thanks to the JPA mappings through Spring Data. Spring Boot helps developers create applications that just run. Specifically, it lets users to create standalone applications that run on their own, without relying on an external web server, by embedding a web server such as Tomcat or Netty into developers app during the initialization process

Spring Web MVC is the original web framework built on the Servlet API and included in the Spring Framework from the very beginning. The formal name “Spring Web MVC” comes from the name of its source module spring-webmvc but it is more commonly known as “Spring MVC”. Spring MVC, like many other web frameworks, is designed around the front controller pattern where a central Servlet, the DispatcherServlet, provides a shared algorithm for request processing while actual work is performed by configurable, delegate components. This model is flexible and supports diverse workflows.

The DispatcherServlet, as any Servlet, needs to be declared and mapped according to the Servlet specification using Java configuration or in web.xml. In turn the DispatcherServlet uses Spring configuration to discover the delegate components it needs for request mapping, view resolution, exception handling, and more.

DispatcherServlet expects a `WebApplicationContext`, an extension of a plain `ApplicationContext`, for its own configuration. `WebApplicationContext` has a link to the `ServletContext` and Servlet it is associated with. It is also bound to the `ServletContext` such that applications can use static methods on `RequestContextUtils` to look up the `WebApplicationContext` if they need access to it.

---

In the 1.3.0 release of Spring Boot and new module is available called Spring Boot Developer Tools. This new Spring Boot module is aimed at improving developer productivity in building Spring Web Applications.

When user is developing a web application in Java, or really any programming language, a common workflow is to code, compile, deploy, and then test in the browser. In scripting languages, such as PHP, there is no compile / deploy phase. The script is evaluated by the server at run time, thus negating the need for a compile / deploy phase.

In the world of Java web development, developers don't have this luxury. Our Java code is compiled down to Java byte code, then deployed to an application server such as Tomcat. The compile, deploy, test phase is a common step in the process of writing software. The longer it takes, the greater the impact it has on the productivity

Spring Boot Developer Tools takes a different approach, it does a restart, not a reload. BUT – under the covers, it is using two class loaders. One for all jar classes in the project, and one for the project classes. Thus on a 'restart', only the project classes are reloaded. The 10's of thousands of classes contained in jar files in the typical Java Spring project are not reloaded. By doing this, restarting Tomcat and the Spring context become VERY fast. Since the Spring context is being restarted, it addresses issues found with the approach used in Spring Loaded.

**Build Tools:** The automatic restart is triggered when changes on the classpath are detected. Thus if users build with Maven or Gradle class files in the target directory will change and an automatic build will be triggered.

**IDEs:** IntelliJ and Eclipse are the two most popular IDEs for Java development. There are some notable differences in use between the two IDEs.

Eclipse is the foundation for the Spring Tool Suite (aka STS). Development of the Spring Boot Developer Tools seems biased towards STS. Which is to be expected. Both are Pivotal products. An automatic restart in Eclipse is triggered with the save action. In Eclipse, this triggers a recompile of the change classes, which triggers the automatic restart.

With IntelliJ the process is slightly different. IntelliJ does not recompile on save, but unlike Eclipse, it does perform automatic file saves for you. IntelliJ can be configured to compile on

save, but this gets disabled when an application is running. Thus in IntelliJ, users need to trigger the build manually, which will in turn fire off the automatic restart. So with the extra step, the developer experience in IntelliJ is not quite as smooth.

Some Features of Spring boot framework includes:



Fig 3.1 Spring Boot Features

- Creates stand-alone spring application with minimal configuration needed.
- It has embedded tomcat, jetty which makes it just code and run the application.
- Provide production-ready features such as metrics, health checks, and externalized configuration.
- Absolutely no requirement for XML configuration.

---

## 3.2 Tools and Technologies :

Spring Tools provides world-class development support for developers of Spring applications. Spring Tools can be used in various coding environments, ranging from Eclipse as a full-featured integrated development environment to Visual Studio Code and Theia as lightweight code editors.

The new generation of Spring Tools is largely built from scratch, incorporating modern technologies and developer tooling architectures.

It runs in separate processes, is built with performance in mind from the start, and knows about the latest Spring technologies

Technical Relevance of spring boot and its staters

1. Web Development
2. Spring Application
3. Application events and listeners
4. Admin features
5. Security

### Web Development

It is well suited for the Spring module for web application development. Users can easily create a self-contained HTTP server using embedded Tomcat, Jetty or Undertow. Users can use the spring-boot- starter-web module to start and run applications quickly.

In Spring's approach to building web sites, HTTP requests are handled by a controller. users can easily identify these requests by the `@Controller` annotation. In the following example, the `HelloController` handles GET requests for `/hello` by returning the name of a View, in this case, "hello". A View is responsible for rendering the HTML content:

#### *HelloController.java*

```
package hello;
```

```
import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

@Controller

public class HelloController {

    @RequestMapping("/hello")

    public String hello(@RequestParam(value="user", required=false, defaultValue="Mr")
String user, Model model) {

        model.addAttribute("user", user);

        return "hello";

    }

}
```

This controller is concise and simple, but there's plenty going on. Let's break it down step by step. The `@RequestMapping` annotation ensures that HTTP requests to `/hello` are mapped to the `hello()` method. The above example does not specify GET vs. PUT, POST, and so forth, because `@RequestMapping` maps all HTTP operations by default. Use `@RequestMapping(method=GET)` to narrow this mapping. `@RequestParam` binds the value of the query String parameter `user` into the `user` parameter of the `hello()` method. This query String parameter is not required; if it is absent in the request, the `defaultValue` of "Mr" is used. The value of the `user` parameter is added to a `Model` object, ultimately making it accessible to the view template.



## SpringApplication

It is a class which provides a convenient way to bootstrap a spring application which can be started from the main method. Users can start their application just by calling a static run() method.

## Application Events and Listeners

Spring Boot uses events to handle a variety of tasks. It allows users to create factory files that are used to add listeners. developers can refer to it by using the ApplicationListener key.

An event can have multiple listeners doing different work based on application requirements.

There are two ways to define a listener. developers can either use the @EventListener annotation or implement the ApplicationListener interface. In either case, the listener class has to be managed by Spring.

## Annotation-Driven

Starting with Spring 4.1 it's now possible to simply annotate a method of a managed bean with @EventListener to automatically register an ApplicationListener matching the signature of the method:

```
@Component
class UserRemovedListener {

    @EventListener
    ReturnedEvent handleUserRemovedEvent(UserRemovedEvent event) {
        // handle UserRemovedEvent ...
        return new ReturnedEvent();
    }

    @EventListener
    void handleReturnedEvent(ReturnedEvent event) {
        // handle ReturnedEvent ...
    }

    ...
}
```

No additional configuration is necessary with annotation-driven configuration enabled. Our method can listen to several events or if developers want to define it with no parameter at all, the event types can also be specified on the annotation itself.

Example: `@EventListener({ContextStartedEvent.class, ContextRefreshedEvent.class})`.

For the methods annotated with `@EventListener` and defined as a non-void return type, Spring will publish the result as a new event for us. In the above example, the `ReturnedEvent` returned by the first method will be published and then handled by the second method.

Spring allows our listener to be triggered only in certain circumstances if developers specify a SpEL condition:

```
@Component
class UserRemovedListener {

    @EventListener(condition = "#event.name eq 'reflectoring'")
    void handleConditionalListener(UserRemovedEvent event) {
        // handle UserRemovedEvent
    }
}
```

The event will only be handled if the expression evaluates to true or one of the following strings: “true”, “on”, “yes”, or “1”. Method arguments are exposed via their names. The condition expression also exposes a “root” variable referring to the raw `ApplicationEvent` (`#root.event`) and the actual method arguments (`#root.args`)

In the above example, the listener will be triggered with `UserRemovedEvent` only when the `#event.name` has the value 'reflectoring',

## Admin Support

Spring Boot provides the facility to enable admin related features for the application. It is used to access and manage applications remotely. Users can enable it by simply using `spring.application.admin.enabled` property.

### Admin Server Setup:

```
<dependency>
  <groupId>de.codecentric</groupId>
  <artifactId>spring-boot-admin-starter-server</artifactId>
  <version>2.4.1</version>
</dependency>
```

After this, the `@EnableAdminServer` will be available, so we'll be adding it to the main class, as shown in the example below:

```
@EnableAdminServer
```

```
@SpringBootApplication
```

```
public class SpringBootAdminServerApplication(exclude =
AdminServerHazelcastAutoConfiguration.class) {
```

```
    public static void main(String[] args) {
        SpringApplication.run(SpringBootAdminServerApplication.class, args);
    }
}
```

### Setting Up a Client:

Now, after we've set up our admin server, developers can register our first Spring Boot application as a client. developers must add the following Maven dependency:

```
<dependency>  
  <groupId>de.codecentric</groupId>  
  <artifactId>spring-boot-admin-starter-client</artifactId>  
  <version>2.4.1</version>  
</dependency>
```

Next, developers need to configure the client to know about the admin server's base URL. For this to happen, developers just add the following property:

```
spring.boot.admin.client.url=http://localhost:8080
```

Starting with Spring Boot 2, endpoints other than health and info are not exposed by default.

### Security Configuration:

The Spring Boot Admin server has access to the application's sensitive endpoints, so it's advised that developers add some security configuration to both admin and client application.

## Security:

Spring Boot applications are spring based web applications. So, it is secure by default with basic authentication on all HTTP endpoints. A rich set of Endpoints are available for developing a secure Spring Boot application.

Spring Security is a framework which provides various security features like: authentication, authorization to create secure Java Enterprise Applications.

It is a sub-project of Spring framework which was started in 2003 by Ben Alex. Later on, in 2004, It was released under the Apache License as Spring Security 2.0.0.

It overcomes all the problems that come during creating non spring security applications and manages a new server environment for the application.

This framework targets two major areas of application are authentication and authorization. Authentication is the process of knowing and identifying the user that wants to access.

Authorization is the process to allow authority to perform actions in the application. We can apply authorization to authorize web requests, methods and access to individual domains.

Technologies that support Spring Security Integration. Spring Security framework supports a wide range of authentication models. These models are either provided by third parties or the framework itself.

Spring Security supports integration with all of these technologies:

- o HTTP BASIC authentication headers
  - o HTTP Digest authentication headers
  - o HTTP X.509 client certificate exchange
  - o LDAP (Lightweight Directory Access Protocol)
  - o Form-based authentication
  - o OpenID authentication
  - o Automatic remember-me authentication
  - o Kerberos
  - o JOSSO (Java Open Source Single Sign-On)
-

- 
- o AppFuse
  - o AndroMDA
  - o Mule ESB
  - o DWR(Direct Web Request)

The beauty of this framework is its flexible authentication nature to integrate with any software solution. Sometimes, developers want to integrate it with a legacy system that does not follow any security standard, there Spring Security works nicely

#### Property Defaults:

Spring-boot does a lot of auto-configurations, including enabling caching by default to improve performance. One such example is caching of templates used by template engines, e.g. *thymeleaf*. But during development, it's more important to see the changes as quickly as possible.

The default behavior of caching can be disabled for *thymeleaf* using the property *spring.thymeleaf.cache=false* in the *application.properties* file. developers do not need to do this manually, introducing this *spring-boot-devtools* does this automatically for us.

#### Automatic Restart:

In a typical application development environment, a developer would make some changes, build the project and deploy/start the application for new changes to take effect, or else try to leverage *JRebel*, etc.

Using *spring-boot-devtools*, this process is also automated. Whenever files change in the classpath, applications using *spring-boot-devtools* will cause the application to restart.

Live Reload: *spring-boot-devtools* module includes an embedded LiveReload server that is used to trigger a browser refresh when a resource is changed.

For this to happen in the browser developers need to install the LiveReload plugin one such implementation is Remote Live Reload for Chrome.

Global Settings: *spring-boot-devtools* provides a way to configure global settings that are not coupled with any application. This file is named as *.spring-boot-devtools.properties* and it located at \$HOME.

Remote Applications: *spring-boot-devtools* provides out of the box remote debugging capabilities via HTTP, to have this feature it is required that *spring-boot-devtools* are packaged as part of the application. This can be achieved by disabling *excludeDevtools* configuration in the plugin in maven.

Remote Update: The remote client monitors the application classpath for changes as is done for remote restart feature. Any change in the classpath causes the updated resource to be pushed to the remote application and a restart is triggered.

---

## Consider the example of Building A simple Calculator web Application using Spring Application

For creating a web application with Spring Boot, following tools are needed

- IDE like visual studio or any other IDE
- Maven
- JDK 1.8+

Setting Up:

### 1 Installation instructions for the Java developer

users can use Spring Boot in the same way as any standard Java library. Simply include the appropriate spring-boot-\*.jar files on the classpath. Spring Boot does not require any special tools integration, so users can use any IDE or text editor; and there is nothing special about a Spring Boot application, so users can run and debug as users would any other Java program.

Although users could just copy Spring Boot jars, developers generally recommend that users use a build tool that supports dependency management (such as Maven or Gradle).

### Maven installation

Spring Boot is compatible with Apache Maven 3.0 or above. If users don't already have Maven installed, users can follow the instructions at [maven.apache.org](https://maven.apache.org).

On many operating systems Maven can be installed via a package manager. If the user is an OSX Homebrew user try `brew install maven`. Ubuntu users can run `sudo apt-get install maven`.

The spring-boot-starter-parent is a great way to use Spring Boot, but it might not be suitable all of the time. Sometimes users may need to inherit from a different parent POM, or users might just not like our default settings. See Section 12.1.2, “Using Spring Boot without the parent POM” for an alternative solution that uses an import scope.



### 9.1.2 Gradle installation

Spring Boot is compatible with Gradle 1.6 or above. If users don't already have Gradle installed, users can follow the instructions at [www.gradle.org/](http://www.gradle.org/).

Spring Boot dependencies can be declared using the `org.springframework.boot` group. Typically the project will declare dependencies to one or more "Starter POMs". Spring Boot provides a useful Gradle plugin that can be used to simplify dependency declarations and to create executable jars.

### Project Structure

Spring Boot does not require any specific code layout or structure. We can always follow some of the best practices suggested by Spring Boot team, however, final structure will be driven by the project requirement.

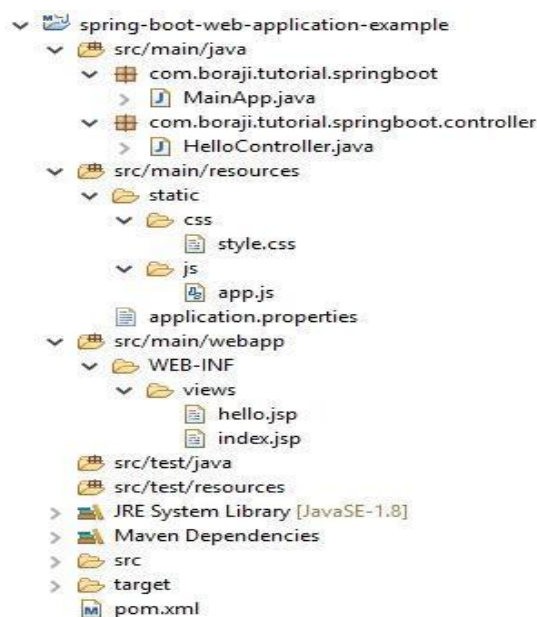
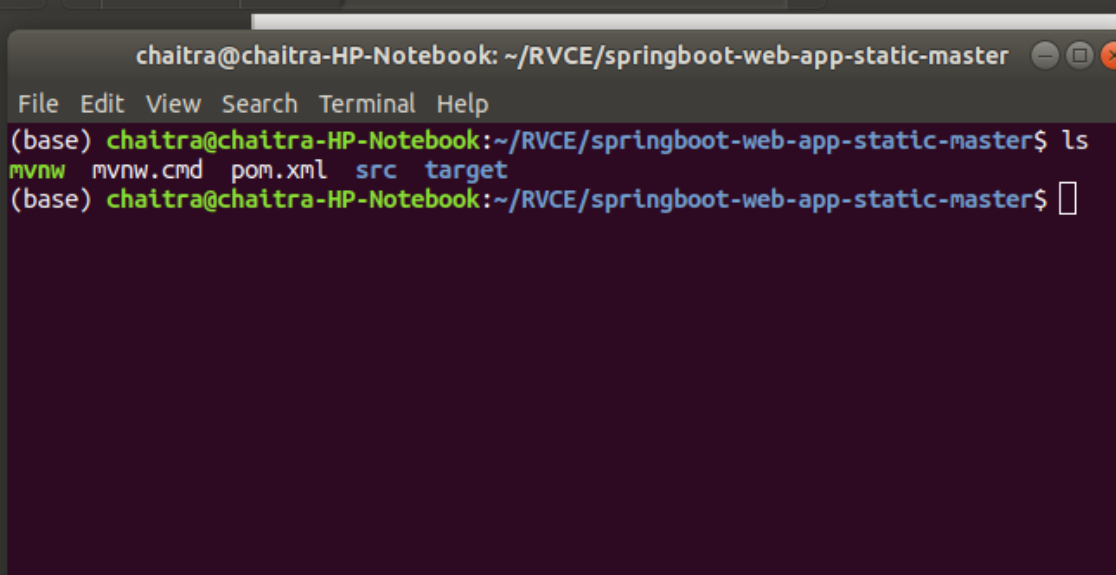


Fig 3.2 Directory structure of Spring boot framework

Here is the layout of the calculator web application

A screenshot of a terminal window titled "chaitra@chaitra-HP-Notebook: ~/RVCE/springboot-web-app-static-master". The terminal shows the command "ls" being executed, which lists the files "mvnw", "mvnw.cmd", "pom.xml", "src", and "target". The prompt "(base) chaitra@chaitra-HP-Notebook:~/RVCE/springboot-web-app-static-master\$" is visible at the bottom of the terminal.

```
chaitra@chaitra-HP-Notebook: ~/RVCE/springboot-web-app-static-master
File Edit View Search Terminal Help
(base) chaitra@chaitra-HP-Notebook:~/RVCE/springboot-web-app-static-master$ ls
mvnw mvnw.cmd pom.xml src target
(base) chaitra@chaitra-HP-Notebook:~/RVCE/springboot-web-app-static-master$
```

Fig 3.3 File structure of Spring Boot Framework

Mvn.cmd :

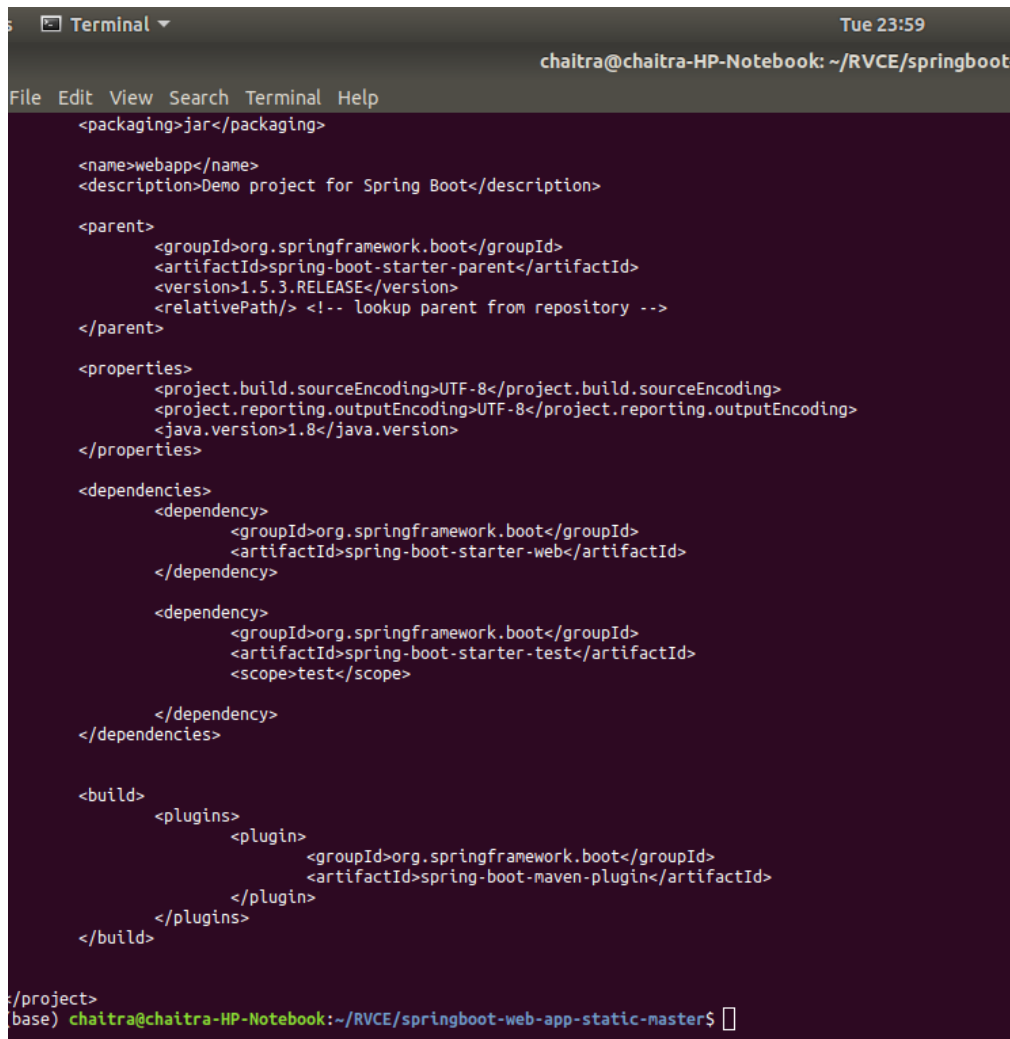
These files are from Maven wrapper. It works similarly to the Gradle wrapper.

This allows users to run the Maven project without having Maven installed and present on the path. It downloads the correct Maven version if it's not found (as far as developers know by default in the user home directory).

The mvnw file is for Linux (bash) and the mvnw.cmd is for the Windows environment

## Pom.xml

Let's start looking into the pom.xml file to understand Spring Boot configurations in more detail. developers will be covering only Spring Boot related changes in pom.xml. Here is the pom.xml file from the sample project.



```
<packaging>jar</packaging>

<name>webapp</name>
<description>Demo project for Spring Boot</description>

<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.5.3.RELEASE</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>

<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
  <java.version>1.8</java.version>
</properties>

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>

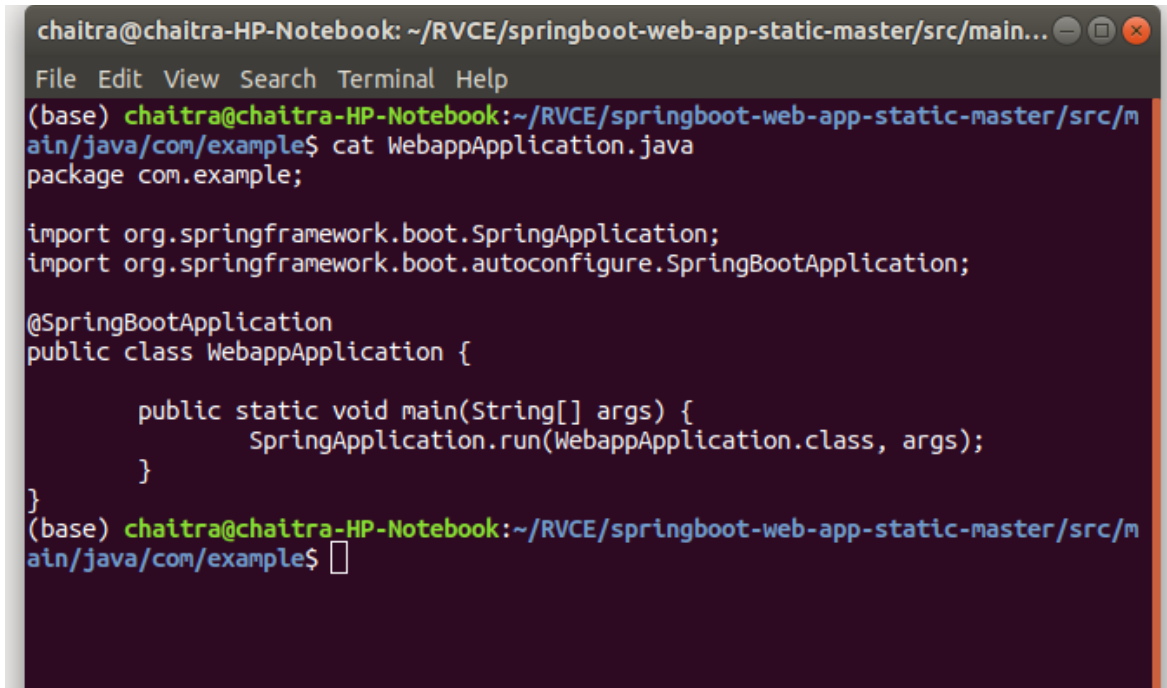
</project>
(base) chaitra@chaitra-HP-Notebook:~/RVCE/springboot-web-app-static-master$
```

Fig 3.4 pom.xml

One of the main features of Spring Boot is the “Starters”, they are an easy way to add required dependencies (jars) in the class path. When using Spring Boot, developers don't have to add jar/dependencies in the class path (In case a starter is not available, users might have to add these dependencies in the pom.xml or can create their own custom starter). developers just need to add correct “Starters” in the pom.xml file and Spring Boot will make sure to add those dependencies automatically.

## Main Application

Here is the main Spring Boot application class, this is a Spring Configuration class. The annotation `@SpringBootApplication` enables the Spring Context and all the startup magic of Spring Boot.

A screenshot of a terminal window titled "chaitra@chaitra-HP-Notebook: ~/RVCE/springboot-web-app-static-master/src/main...". The terminal shows the command to view the file "WebappApplication.java" and its contents. The code is as follows:

```
(base) chaitra@chaitra-HP-Notebook:~/RVCE/springboot-web-app-static-master/src/main/java/com/example$ cat WebappApplication.java
package com.example;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class WebappApplication {

    public static void main(String[] args) {
        SpringApplication.run(WebappApplication.class, args);
    }
}
(base) chaitra@chaitra-HP-Notebook:~/RVCE/springboot-web-app-static-master/src/main/java/com/example$
```

Fig 3.5 Main application

`@SpringBootApplication` is equivalent to using `@Configuration`, `@EnableAutoConfiguration` and `@ComponentScan` with their default values. If users are starting the project, it's recommended to use annotation.

`@Configuration` annotation indicates that a class declares one or more `@Bean` methods and may be processed by the Spring container to generate bean definitions and service requests for those beans at runtime

`@EnableAutoConfiguration` of the Spring Application Context, attempting to guess and configure beans that users are likely to need. Auto-configuration classes are usually applied based on the classpath and what beans users have defined. For example, if users have `tomcat-embedded.jar` on the classpath users are likely to want a `TomcatServletWebServerFactory` (unless users have defined their own

ServletWebServerFactory bean). to automatically pick up all Spring components, including @Configuration classes.

Another interesting feature of the main class is the main method. This is a standard method that will follow standard Java workflow. The main class will pass on control to Spring Boot SpringApplication class.

One of the most important annotations in spring is @ComponentScan which is used along with the @Configuration annotation to specify the packages that developers want to be scanned. @ComponentScan without arguments tells Spring to scan the current package and all of its sub-packages.

UI Template :

Here is the simple UI File Directory of a simple calculator.

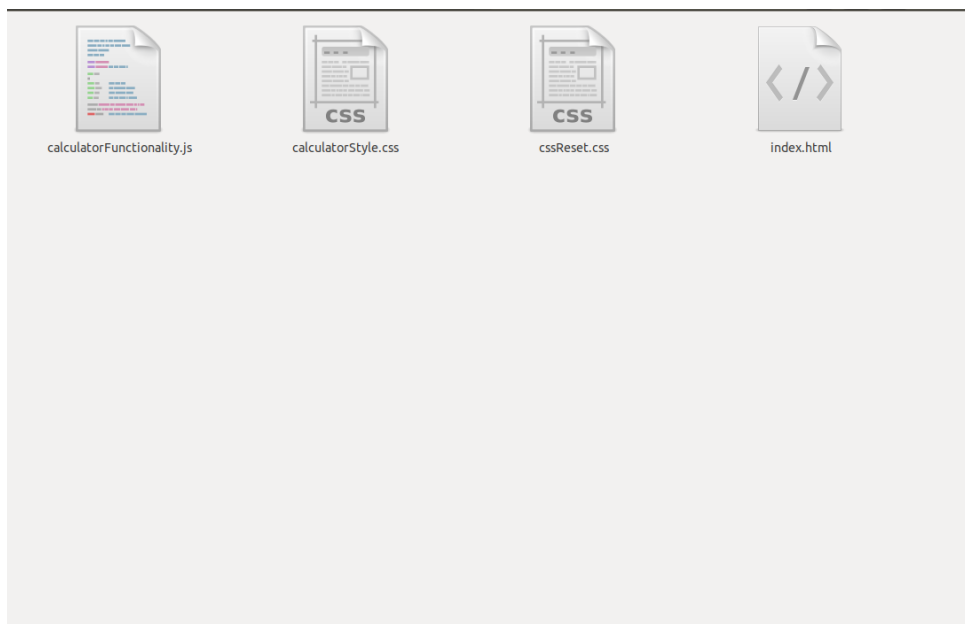


Fig 3.6 UI file directory

The UI directory includes html5, css3 and javascript file for functionality of the calculator

The HTML5 code:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Simple Web Calculator</title>
  <link type="text/css" rel="stylesheet" href="calculatorStyle.css">
  <link type="text/css" rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
  <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
  <link type="image/png" rel="shortcut icon" href="imgs/favicon.png">
</head>
<body>
  <div id="container">
    <section id="main">
      <header><h1 id="title">Simple Web Calculator</h1></header>
      
      <div id="calculator">
        <div id="calculator-interface">
          <div id="calculator-screen">
            <div id="screen-top">
              <p id="nums-top"></p>
            </div>
            <div id="screen-bottom">
              <p id="nums-bottom">0</p>
            </div>
          </div>
          <div id="functional-container">
            <div id="negate" class="button">+/-</div>
            <div id="pi" class="button">π</div>
            <div id="euler" class="button">e</div>
            <div id="factorial" class="button">n!</div>
            <div id="ce" class="button">CE</div>
          </div>
        </div>
      </div>
    </section>
  </div>

```

Fig 3.7 Html Code

Calculator Title: This is the title at the top of our application, “Spring Boot Calculator”.

Output Screen: This will be our output screen, where all text will be shown. Like the input that the user will type and the answer calculated from the user input.

Question Output: This will be the input given by the user.

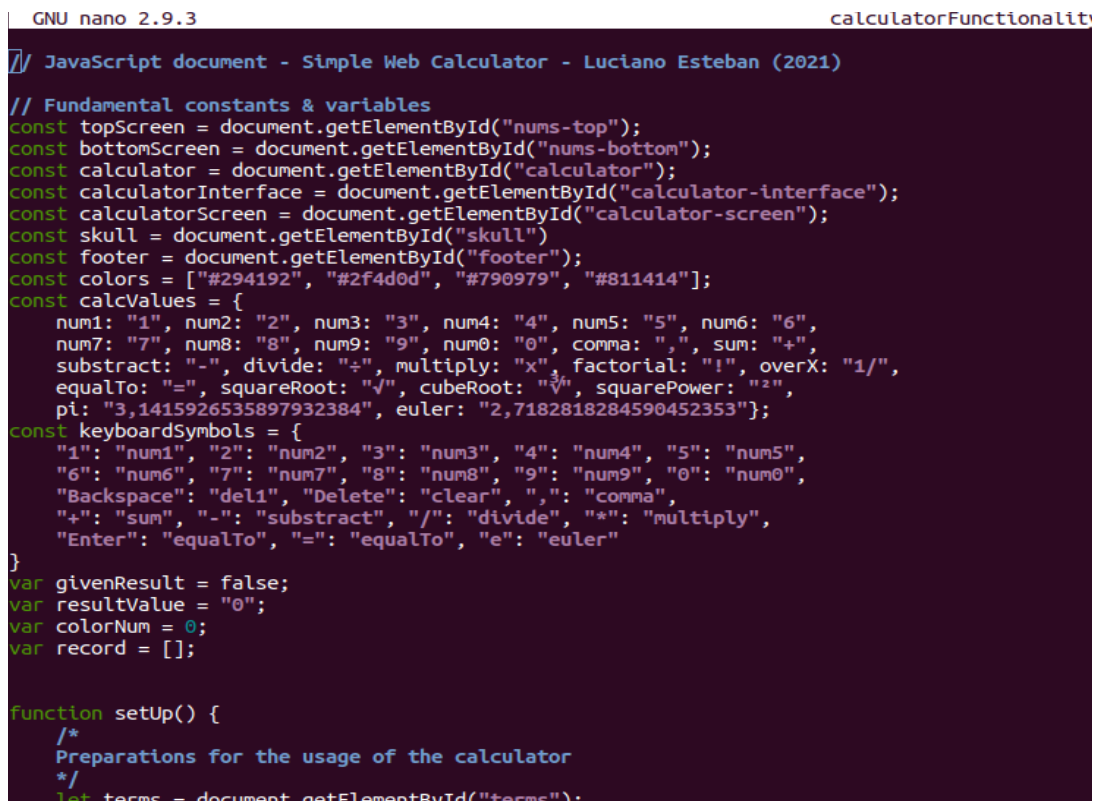
Answer Output: This will be the result calculated from user input.

## JavaScript Code:

First, developers should select all our calculator's buttons and operations. developers can do this by putting some classes in the HTML file. However, since developers do not want to mix CSS classes with JavaScript classes, developers can use data attributes to select them instead.

data-operation to represent the operation buttons, data-numbers to represent the number buttons, data-all-clear to represent the All-Clear button and data-delete to represent the Delete button. developers can add these classes to the previous-operand and current-operand.

Here's how it would look like in the code:



```

GNU nano 2.9.3 calculatorFunctionality.js
// JavaScript document - Simple Web Calculator - Luciano Esteban (2021)

// Fundamental constants & variables
const topScreen = document.getElementById("nums-top");
const bottomScreen = document.getElementById("nums-bottom");
const calculator = document.getElementById("calculator");
const calculatorInterface = document.getElementById("calculator-interface");
const calculatorScreen = document.getElementById("calculator-screen");
const skull = document.getElementById("skull");
const footer = document.getElementById("footer");
const colors = ["#294192", "#2f4d0d", "#790979", "#811414"];
const calcValues = {
  num1: "1", num2: "2", num3: "3", num4: "4", num5: "5", num6: "6",
  num7: "7", num8: "8", num9: "9", num0: "0", comma: ".", sum: "+",
  subtract: "-", divide: "÷", multiply: "x", factorial: "!", overX: "1/",
  equalTo: "=", squareRoot: "√", cubeRoot: "∛", squarePower: "²",
  pi: "3,1415926535897932384", euler: "2,7182818284590452353"};
const keyboardSymbols = {
  "1": "num1", "2": "num2", "3": "num3", "4": "num4", "5": "num5",
  "6": "num6", "7": "num7", "8": "num8", "9": "num9", "0": "num0",
  "Backspace": "del1", "Delete": "clear", ",": "comma",
  "+": "sum", "-": "subtract", "/": "divide", "*": "multiply",
  "Enter": "equalTo", "=": "equalTo", "e": "euler"
}

var givenResult = false;
var resultValue = "0";
var colorNum = 0;
var record = [];

function setUp() {
  /*
   Preparations for the usage of the calculator
  */
  let terms = document.getElementById("terms");

```

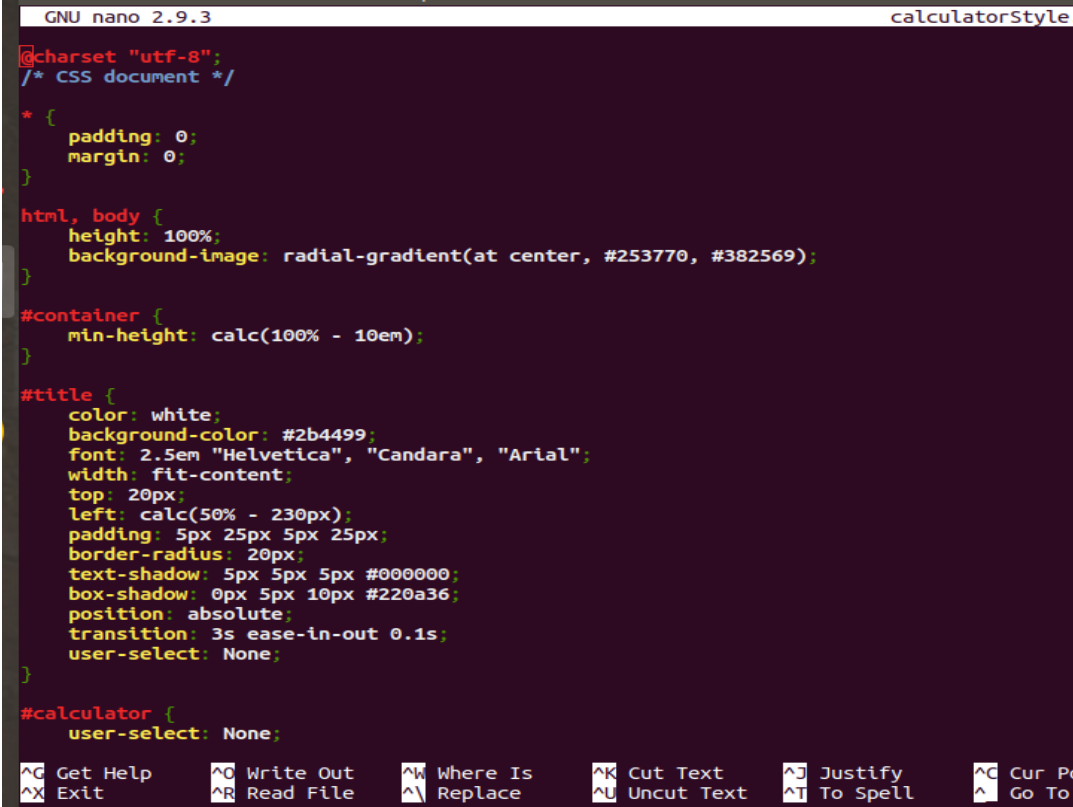
Fig 3.8 Javascript code

In the JavaScript file, define some constant variables which will represent the number buttons. developers will then perform a query using document.querySelectorAll(). This function will allow users to get all elements that match a certain string.

### Styling the calculator

Next, developers need to style the calculator using CSS. First, select all the elements, including the before and after elements. developers can then apply the box-sizing attribute and set it as border-box.

Css file:



```
GNU nano 2.9.3 calculatorStyle
@charset "utf-8";
/* CSS document */

* {
  padding: 0;
  margin: 0;
}

html, body {
  height: 100%;
  background-image: radial-gradient(at center, #253770, #382569);
}

#container {
  min-height: calc(100% - 10em);
}

#title {
  color: white;
  background-color: #2b4499;
  font: 2.5em "Helvetica", "Candara", "Arial";
  width: fit-content;
  top: 20px;
  left: calc(50% - 230px);
  padding: 5px 25px 5px 25px;
  border-radius: 20px;
  text-shadow: 5px 5px 5px #000000;
  box-shadow: 0px 5px 10px #220a36;
  position: absolute;
  transition: 3s ease-in-out 0.1s;
  user-select: None;
}

#calculator {
  user-select: None;
}
```

Fig 3.9 CSS3 code

### Styling the calculator

Next, developers need to style the calculator using CSS. First, select all the elements, including the before and after elements. developers can then apply the box-sizing attribute and set it as border-box.

The first thing developers can do is to set the output to span across the entire width of the calculator. developers can do this by using the grid-column attribute again and setting it to span from column 1 to -1, essentially just the last column.

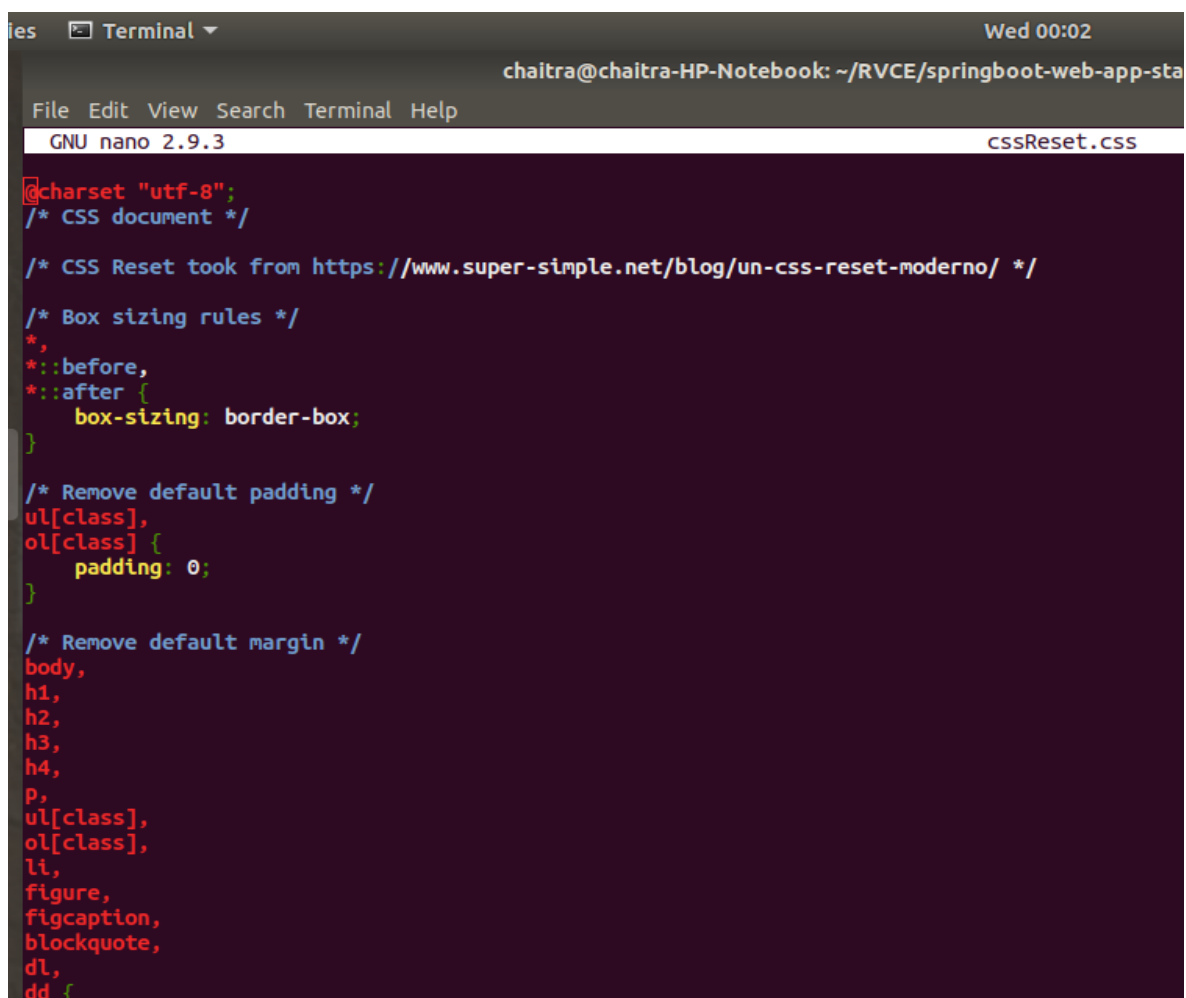


Next, developers will change the background-color to black with transparency of 75%. Then, developers will align all the elements inside the container.

The easiest way is by using flex. Therefore, set the display attribute to flex and align-items attribute to flex-end. The output elements will be positioned at the right side of the calculator. To space them out, developers can use the justify-content attribute and set it to space-around. developers can also change the flex-direction and set it to column to align the output elements vertically.

Next, developers can set the padding to any desired value. Also, to make the output elements wrap when they get too long, developers can use the word-wrap attribute to choose where the words should break. Besides, developers can add a word-break and set it to break-all. developers should style the previous and current operands in the output class.

The final CSS code comes out like this:

A screenshot of a terminal window with a dark background. At the top, it shows 'Terminal' and 'Wed 00:02'. Below that, the user 'chaitra' is at a terminal on a machine named 'chaitra@chaitra-HP-Notebook' with the working directory '~/RVCE/springboot-web-app-sta'. The terminal is running 'GNU nano 2.9.3' and editing a file named 'cssReset.css'. The code displayed is a CSS reset, starting with '@charset "utf-8";' and '/\* CSS document \*/'. It includes a comment about the CSS Reset source and then defines rules for box sizing, removing default padding for lists, and removing default margins for various HTML elements like body, h1-h4, p, ul, ol, li, figure, figcaption, blockquote, dl, and dd.

```
ies Terminal ▾ Wed 00:02
chaitra@chaitra-HP-Notebook: ~/RVCE/springboot-web-app-sta
File Edit View Search Terminal Help
GNU nano 2.9.3 cssReset.css
@charset "utf-8";
/* CSS document */

/* CSS Reset took from https://www.super-simple.net/blog/un-css-reset-moderno/ */

/* Box sizing rules */
*,
*::before,
*::after {
  box-sizing: border-box;
}

/* Remove default padding */
ul[class],
ol[class] {
  padding: 0;
}

/* Remove default margin */
body,
h1,
h2,
h3,
h4,
p,
ul[class],
ol[class],
li,
figure,
figcaption,
blockquote,
dl,
dd {
```

Fig 3.10 CSS3 code

User Interface of simple Calculator:

Fig 1.9 UI of Simple Calculator

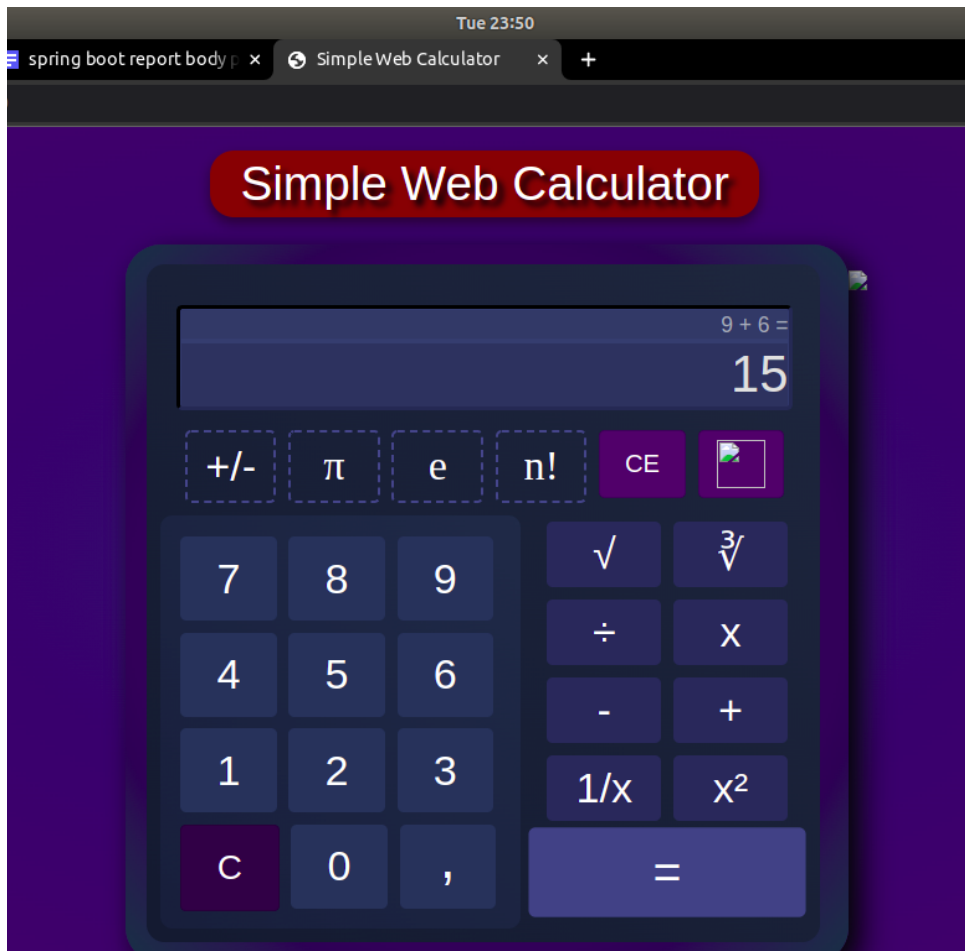


Fig 3.11

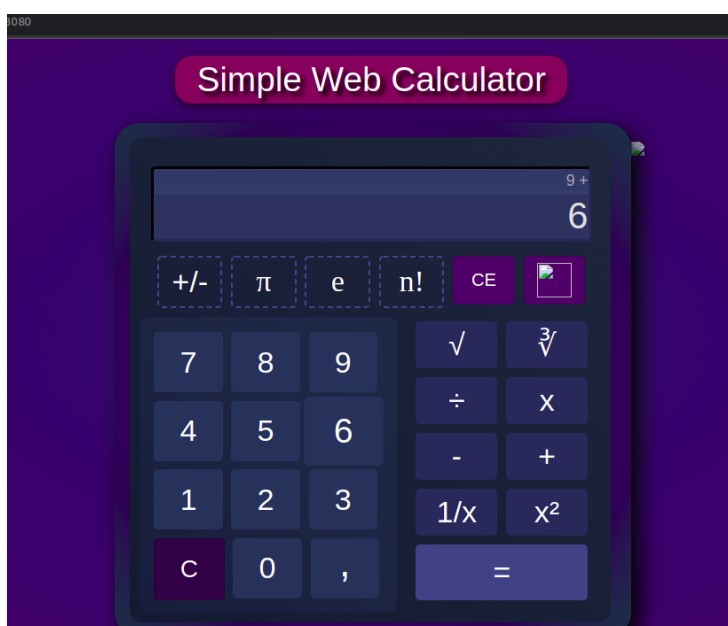


Fig 3.12

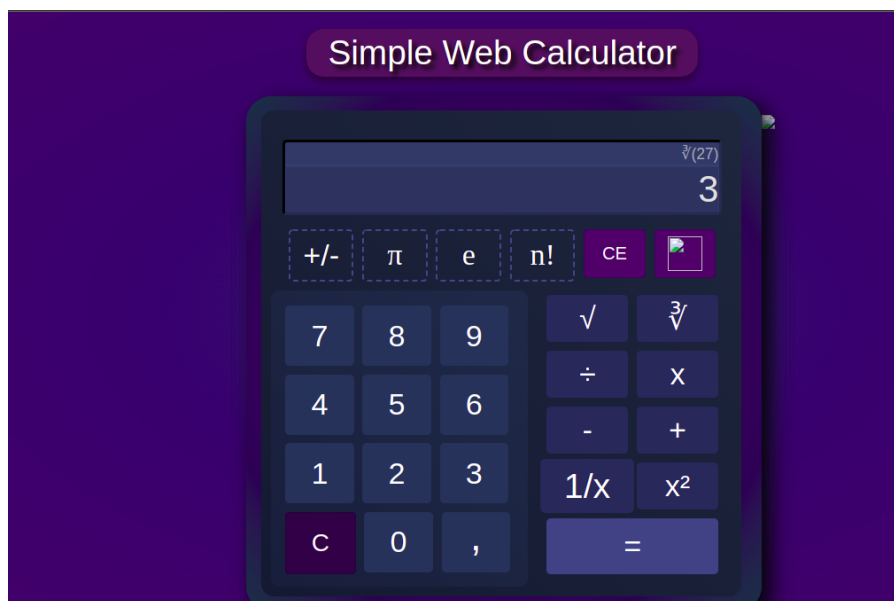


Fig 3.13

While using Thymeleaf as the template engine, Spring Boot will look for resources by surrounding the view name with a prefix and suffix (externalized to `spring.thymeleaf.prefix` and `spring.thymeleaf.suffix`, defaults `'classpath:/templates/'` and `' .html'` respectively).

### Run Application

We are done with the simple web application, it's time to run the application. Although it is possible to package this service as a traditional WAR file for deployment to an external application server, the simpler approach demonstrated is to create a standalone application. To run the application from IDE, developers need to run the web application as a standalone java application.

With Maven, developers can run the application using `mvn spring-boot:run` command.

We can build the JAR file with the `mvn clean package` command and run the jar by using `java -jar target/demo-app-0.1.0.jar`.

Now the site is up and running, visit, <http://localhost:8080/welcome> and if everything is in place, the user should have the following output on the web browser.

---

### 3.3 Sustainability and Societal Concern:

Spring Framework has a better future ahead. Spring is now the de facto standard when it comes to the Java world. Spring Boot is a refined, updated and modern version of Spring. It is now production ready. Companies have started choosing it for backend development, web apps and enterprise applications. There is little hassle involved in setting up applications in comparison with Spring Framework which involves lots of xml configuration. It's a next generation framework and quite popular. Most of the other frameworks are built on top of or provide support for spring boot.

It's an excellent choice for microservices architecture. Fast, easy and developer productive framework like Grails, Play etc. It's backed by strong spring framework communities which are being adopted by many enterprises for their software development.

#### Common Concerns #1: Going Too Low Level

While understanding the internals of a particular library and its implementation is for the most part good and necessary (and can be a great learning process as well), it's detrimental to your development as a software engineer to be constantly tackling the same low-level implementation details. There is a reason why abstractions and frameworks such as Spring exist, which is precisely to separate users from repetitive manual work and allow users to concentrate on higher level details— your domain objects and business logic.

So embrace the abstractions - the next time users are faced with a particular problem, do a quick search first and determine whether a library solving that problem is already integrated into Spring; nowadays, chances are you'll find a suitable existing solution. As an example of a useful library, I'll be using Project Lombok annotations in examples for the remainder of this article. Lombok is used as a boilerplate code generator and the lazy developer within users hopefully shouldn't have a problem acquainting themselves with the library.

#### Common Concerns #2: 'Leaking' Internals

Exposing your internal structure is never a good idea because it creates inflexibility in service design and consequently promotes bad coding practices. 'Leaking' internals are manifested by making database structure accessible from certain API endpoints

---

### Common Concerns #3: Lacking Separation of Concerns

As your application grows, code organization increasingly starts becoming an ever more important matter. Ironically, most of the good software engineering principles start to break down at scale – especially in cases where not much thought has been given to the application architecture design. One of the most common Concerns developers then tend to succumb to is mixing code concerns, and it's extremely easy to do!

What usually breaks separation of concerns is just 'dumping' new functionality into existing classes. This is, of course, a great short-term solution (for starters, it requires less typing) but it inevitably becomes a problem further down the road, be it during testing, maintenance, or somewhere in between.

### Common Concerns #4: Inconsistency and Poor Error Handling

The topic of consistency is not necessarily exclusive to Spring (or Java, for that matter), but still is an important facet to consider when working on Spring projects. While coding style can be up for debate (and is usually a matter of agreement within a team or within an entire company), having a common standard turns out to be a great productivity aid. This is especially true with multi-person teams; consistency allows hand-off to occur without many resources being spent on hand-holding or providing lengthy explanations regarding the responsibilities of different classes

Consider a Spring project with its various configuration files, services and controllers. Being semantically consistent in naming them creates an easily searchable structure where any new developer can manage his way around the code; appending Config suffixes to your configuration classes, Service suffixes to your services and Controller suffixes to your controllers, for example.

Closely related to the topic of consistency, error handling on the server-side deserves a specific emphasis. If users ever had to handle exception responses from a poorly written API, users probably know why– it can be a pain to properly parse exceptions, and even more painful to determine the reason for why those exceptions occurred in the first place.

As an API developer, you'd ideally want to cover all user-facing endpoints and translate them into a common error format. This usually means having a generic error code and description

---

rather than the cop-out solution of a) returning a “500 Internal Server Error” message, or b) just dumping the stack trace to the user (which should actually be avoided at all costs since it exposes your internals in addition to being difficult to handle client-side).

### Common Concerns #5: Improperly Dealing with Multithreading

Regardless of whether it is encountered in desktop or web apps, Spring or no Spring, multithreading can be a tough nut to crack. Problems caused by parallel execution of programs are nerve-racking elusive and often times extremely difficult to debug - in fact, due to the nature of the problem, once users realise users are dealing with a parallel-execution issue users are probably going to have to forego the debugger entirely and inspect your code “by hand” until users find the root error cause. Unfortunately, a cookie-cutter solution does not exist for solving such issues; depending on your specific case, users are going to have to assess the situation and then attack the problem from the angle users deem is best.

here are some practical considerations for debugging and preventing multithreading errors:

#### Avoid Global State

First, always remember the “global state” issue. If users are creating a multithreaded application, absolutely anything that is globally modifiable should be closely monitored and, if possible, removed altogether. If there is a reason why the global variable must remain modifiable, carefully employ synchronization and track your application’s performance to confirm that it’s not sluggish due to the newly introduced waiting periods.

#### Avoid Mutability

This one comes straight from functional programming and, adapted to OOP, states that class mutability and changing state should be avoided. This, in short, means foregoing setter methods and having private final fields on all your model classes. The only time their values are mutated is during construction. This way users can be certain that no contention problems arise and that accessing object properties will provide the correct values at all times.

#### Log Crucial Data

Assess where your application might cause trouble and preemptively log all crucial data. If an error occurs, users will be grateful to have information stating which requests were received and have better insight into why your application misbehaved. It’s again necessary to note that

logging introduces additional file I/O and should therefore not be abused as it can severely impact your application's performance.

### Reuse Existing Implementations

Whenever users are in need of spawning their own threads (e.g. for making async requests to different services), reuse existing safe implementations rather than create your own solutions. This will, for the most part, mean utilizing `ExecutorServices` and Java 8's neat functional-style `CompletableFuture`s for thread creation. Spring also allows asynchronous request processing via the `DeferredResult` class.

### Common Concerns #6: Not Employing Annotation-Based Validation

Let's imagine our `TopTalent` service from earlier requires an endpoint for adding new `TopTalents`. Furthermore, let's say that, for some really valid reason, every new name needs to be exactly 10 characters long.

---

### 3.4 Conclusion

Spring Boot has become an integral part of the Java ecosystem, offering an efficient and scalable toolbox for building Spring applications with a microservices architecture. It speeds up the development and deployment processes by using intuitive default settings for unit and integration tests. What's more, Spring Boot helps developers build robust applications with clear and secure configurations without spending a lot of time and effort on figuring out the intricacies of Spring.

Spring is an excellent choice for developers to build enterprise Java applications. However, it is highly beneficial when used alongside Spring Boot. While Spring offers developers flexibility and versatility, Spring Boot focuses on reducing code length and configuration, thereby providing developers with the easiest and simplest method to build an application. The added advantages of Spring Boot are of great value as they reduce development time and effort by a considerable margin.

The most notable features properties were Auto-configuration, Standalone, Opinionated all in all, the above serves one purpose to get the program up and running as quick as possible

Think of Spring Boot as another programmer who does all the dirty work of configuring and managing dependencies by intelligently sensing the direction of the project. It embeds the Tomcat server, configures servlet containers, and bootstraps REST APIs if it senses that the application is a Web project. If it is a JPA or JDBC project that accesses relational databases, it provides the necessary boilerplate code and configures it automatically. As a programmer, one needs to be aware of what is going behind the scenes and focus on the business aspect of the project because managing and monitoring are already taken care of. Unless there is a specific need, tweaking the project structure provided by Spring Boot is almost unnecessary. Perhaps, programming cannot be simpler than this.



---

## BIBLIOGRAPHY

1. JIBRIL ADAMU, Descriptive Analysis of Built-in Security Features in Web Development Frameworks, 2022 16th International Conference on Ubiquitous Information Management and Communication (IMCOM) | 978-1-6654-2678-7/22/\$31.00 ©2022 IEEE | DOI: 10.1109/IMCOM53663.2022.972175
2. Theofanis Vassiliou-GiolesQuality ,Assurance of Microservices - When to trust the microservice test results, 2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C) | 978-1-6654-7836-6/21/\$31.00 ©2021 IEEE | DOI: 10.1109/QRS-C55045.2021.0002
3. Zhe Wang, Feng Tang Design and Implementation of a Health Status Reporting System Based on Spring Boot, 2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE) | 978-1-7281-9146-1/20/\$31.00 ©2020 IEEE | DOI: 10.1109/ICAICE51518.2020.0009
4. Ashraful Haqu, Rasheq Rahman, Saifur Rahman, Microservice-based Architecture of a Software as a Service (SaaS) Building Energy Management Platform
5. Mykhajlo Klymash, Ihor Tchaikovskiy, Olena Hordiichuk-Bublivska, Yulia , Research of Microservices Features in Information Systems Using Spring Boot, 2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T) | 978-1-7281-9177-5/20/\$31.00 ©2020 IEEE | DOI: 10.1109/PICST51311.2020.9467
6. Yuan Zuo , Yulei Wu , Senior Member, IEEE, Geyong Min , Chengqiang Huang, and Ke Pei, An Intelligent Anomaly Detection Scheme for Microservices Architectures With Temporal and Spatial Data Analysis, IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING, VOL. 6, NO. 2, JUNE 2020
7. Liu xuchen, Li chaoyu, Design and Implementation of a Spring Boot-Based Data Collection System , 2020 12th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)

8. Michal Gajewski, Michal Gajewski, Analysis and Comparison of the Spring Framework and Play Framework Performance, Used to Create Web Applications in Java, 978-1-7281-4029-2/19/\$31.00 ©2019 IEEE
9. Du Ying-kui<sup>1</sup>, Wang Yang<sup>1</sup>, Guan Ping<sup>2</sup>, Peng Y, Cloud Data Monitoring Management and Visual Application System Based on Spring Boot, 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC 2019)
10. Kavya Guntupally, Ranjeet Devarakonda, Kenneth Kehoe, Spring Boot based REST API to Improve Data Quality Report Generation for Big Scientific Data: ARM Data Center Example, 2018 IEEE International Conference on Big Data (Big Data)