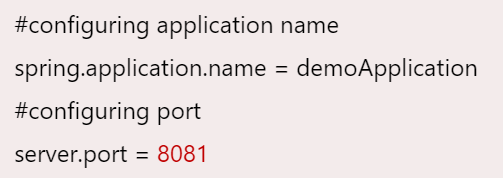
1. **Spring Boot** is a module of Spring Framework. It is used to create stand-alone, production-grade Spring Based Applications with minimum efforts. It is better to use if we want to develop a simple Spring-based application or RESTful services.
2. Spring Boot built-in **starters** make development easier and rapid.
3. Spring Boot Framework comes with a built-in mechanism for application configuration using a file called **application.properties**. The properties have default values. We can set a property(s) for the Spring Boot application. Spring Boot also allows us to define our own property if required.



1. **YAML** Properties File
   1. Spring Boot provides another file to configure the properties is called yml file. The Yaml file works because the Snake YAML jar is present in the classpath. Instead of using the application.properties file, we can also use the application.yml file, but the Yml file should be present in the classpath.
2. Spring Boot **auto-configuration** automatically configures the Spring application based on the jar dependencies that we have added.
3. The annotation @SpringBootApplication is the combination of three annotations: @ComponentScan, @EnableAutoConfiguration, and @Configuration.
4. Spring Boot DevTools
   1. DevTools stands for Developer Tool. The aim of the module is to try and improve the development time while working with the Spring Boot application. Spring Boot DevTools pick up the changes and restart the application.
5. Spring Boot AOP
   1. It provides a solution to implement cross-cutting concerns.
      1. Implement the cross-cutting concern as an aspect.
      2. Define pointcuts to indicate where the aspect has to be applied.
6. Spring Boot Thymeleaf
   1. It is a server-side Java template engine for both web (servlet-based) and non-web (offline) environments. It is perfect for modern-day HTML5 JVM web development. It provides full integration with Spring Framework.
7. Spring Boot Annotations
   1. **@Required**: It applies to the bean setter method. It indicates that the annotated bean must be populated at configuration time with the required property, else it throws an exception BeanInitilizationException.
   2. **@Autowired**: It is used to autowire spring bean on setter methods, instance variable, and constructor. When we use @Autowired annotation, the spring container auto-wires the bean by matching data-type.
   3. **@Configuration**: It is a class-level annotation. The class annotated with @Configuration used by Spring Containers as a source of bean definitions.
   4. **@ComponentScan**: It is used when we want to scan a package for beans. It is used with the annotation @Configuration.
   5. **@Bean**: It is a method-level annotation. It is an alternative of XML <bean> tag. It tells the method to produce a bean to be managed by Spring Container.
   6. **@Component**: It is a class-level annotation. It is used to mark a Java class as a bean. A Java class annotated with @Component is found during the classpath. The Spring Framework pick it up and configure it in the application context as a Spring Bean.
   7. **@Controller**: The @Controller is a class-level annotation. It is a specialization of @Component. It marks a class as a web request handler. It is often used to serve web pages. By default, it returns a string that indicates which route to redirect.
   8. **@Service**: It is also used at class level. It tells the Spring that class contains the business logic.
   9. **@Repository**: It is a class-level annotation. The repository is a DAOs (Data Access Object) that access the database directly. The repository does all the operations related to the database.
   10. **@EnableAutoConfiguration**: It auto-configures the bean that is present in the classpath and configures it to run the methods.
   11. **@RequestMapping**: It is used to map the web requests. It has many optional elements like consumes, header, method, name, params, path, produces, and value. We use it with the class as well as the method.
   12. **@RestController**: It can be considered as a combination of @Controller and @ResponseBody annotations.
   13. **@ResponseBody**: It binds the method return value to the response body. It tells the Spring Boot Framework to serialize a return an object into JSON and XML format.
   14. **@RequestBody**: It is used to bind HTTP request with an object in a method parameter.
   15. **@PathVariable**: It is used to extract the values from the URI.
   16. **@RequestParam**: It is used to extract the query parameters form the URL.
   17. **@RequestHeader**: It is used to get the details about the HTTP request headers.
   18. **@RequestAttribute**: It binds a method parameter to request attribute.