Installation and Configuration

Maven Dependency

Parent tag inherits common spring dependencies from parent POM file. Parent tag provides inheritance between POM files t

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.6.RELEASE</version>
</parent>

JDK 1.8 and above is required by Spring boot support annotation
<!-- Spring boot need JDK Version 1.8 and above  -->
<properties>
    <java.version>1.8</java.version>
</properties>

Spring Boot Starters. Starters are a set of convenient dependency descriptors which we can
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>

<!--  Spring Boot Starter Security dependency is used for Spring Security  -->
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-security</artifactId>
</dependency>

<!-- Spring Boot Actuator provides secured endpoints for monitoring and managing App. To ge
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>

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

<!-- Spring JPA dependency -->
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

<!-- MySQL database driver dependency -->
<dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.12</version>
</dependency>

<!-- Spring mail dependency -->
<dependency>
        <groupId>org.springframework.boot</groupId>
```

```
        <artifactId>spring-boot-starter-mail</artifactId>
</dependency>

<!-- Used for logger dependency  -->
<dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
        <version>1.2.17</version>
</dependency>

<!-- Used to plug-in Tomcat -->
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <scope>provided</scope>
</dependency>
```

Start the application; Entry Point

Each application has one startup file that contains *main()* method. Startup class must be annotated by *@SpringBootApplicati*

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

Spring application is started by running following static method:

```
SpringApplication.run(Application.class, args);
```

Annotation @SpringBootApplication includes Auto-Configuration, Component Scan, and Spring Boot Configuration.

@ComponentScan

Spring Boot application scans all the beans and package declarations when the application initializes. By default *@SpringBo*

If you want to scan components from different then default packages then use *@ComponentScan* annotation.

```
@SpringBootApplication
@ComponentScan(basePackages = {
        "com.sunilos.utility",
        "com.sunilos.common
    }
)
public class DemoApplication {
   public static void main(String[] args) {
      SpringApplication.run(DemoApplication.class, args);
   }
   @Bean
   public RestTemplate getRestTemplate() {
      return new RestTemplate();
   }
}
```

Bean Configuration

You can configure custom beans using *@Configuration* and *@Bean* annotation in custom class. Keep new class in the same

```
@Configuration
public class AppConfig {

@Bean
 public UserDTO userBean() {
        UserDTO dto = new UserDTO();
        dto.setFirstName("Ram");
        dto.setLastName("Sharma");
        return dto;
 }
}
```

You can also define bean in startup file using @Bean tag. Separate configuration classes are used to group similar types of

Application properties

Spring boot keeps application configurable parameters into application.properties file. This file is located in root class-path.

Configurable parameters contain by this file are

- Server port number
- Data connection pool properties
- JPA properties
- Email server properties

- Custom properties

```
#Server info
server.port = 8080
spring.application.name = SunilOSApp

#Custom properties
page.size=10

##Data connection pool
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url = jdbc:mysql://localhost:3306/ORS_P10?useSSL=false
spring.datasource.username = ram
spring.datasource.password = ram

## Hibernate/JPA  Properties
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql = true
org.hibernate.dialect.MySQLDialect
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MariaDB53Dialect
spring.jpa.properties.hibernate.current_session_context_class=org.springframework.orm.hiber

#Email properties
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=my@gmail.com
spring.mail.password=mypass
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
```

You can add application context path using following predefined property.

```
server.servlet.context-path =/SunilOSApp
```

Copyright © Rays Technologies / All rights reserved.