# LAB- Spring boot basics

In this lab you will be understanding how to use Spring boot

Before you proceed, make sure that you have gone through **00-javaconfig document**

In this lab, you will be working on **00springboot-start** project

**Steps**

**1)** Open pom.xml and observe that it is having way2learnparent as the parent. Remove the parent tag now.

Configure pom as shown below :

```
  <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.2.5.RELEASE</version>
    </parent>

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

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

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

       <dependency>
                    <groupId>commons-dbcp</groupId>
                    <artifactId>commons-dbcp</artifactId>
              </dependency>

       <dependency>
              <groupId>mysql</groupId>
              <artifactId>mysql-connector-java</artifactId>
       </dependency>


 </dependencies>
```

```
<build>
            <plugins>
                  <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-compiler-plugin</artifactId>
                    <configuration>
                      <source>1.8</source>
                      <target>1.8</target>
                    </configuration>
                  </plugin>

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

2) The code given to you in this start project is same as the one in **01javaconfig-solution** project

3) Open ApplicationConfig.java and InfrastructureConfig.java and observe the bean definitions. We want to eliminate all these configurations and use Spring boot so that spring boot can automatically configure beans without declaration

4) Open JdbcAccountRepositoryImpl.java,JdbcTransactionRepositoryImpl.java, JdbcRewardRepositoryImpl.java, EmailServiceImpl.java and BankServiceImple.java and annotate them using @Component and and wire the dependencies using @Autowired.

5) Write A class BankApplication.java under the base package com.way2learnonline . **Make Sure that you create a class under base package otherwise you will face issues later.**

6) Annotate BankApplication class with @Configuration and @ComponentScan. For @ComponentScan , if you don't mention base package the current package in which it is declared will be considered as base package

7) Use `@PropertySource("classpath:db.properties")` on top of BankApplication class to specify the location of db.properties

8) Copy the all bean definitions from InfrastructureConfig.java in to Bank Application class.

**Now delete ApplicationConfig.java and InfrastructureConfig.java**

8) write main method inside BankApplication class as show below

```
public static void main(String[] args) throws Exception {

      ApplicationContext context=SpringApplication.run(BankApplication.class,args);
            BankService bankService=context.getBean(BankService.class);
            bankService.transfer(new Long(1), new Long(2),1000);
```

```
}
```

Run it and observe that it runs without any exceptions

9) Till now there is nothing special. We didnt use spring boot.

Now we want to enable auto configuration such that **SpringApplication.run()** class can configure required beans automatically .

So annotate BankApplication with `@EnableAutoConfiguration`

We dont want to configure bean for DataSourceInitializer. We want the Spring application to detect the .sql files and execute them automatically. So, remove `dataSourceInitializer()` AND `databasePopulator()` methods from BankApplication Class .

Also remove

```
@Value("classpath:dbscripts.sql")private Resource dbscript;
```

If there are files with name schema.sql and data.sql in classpath, SpringApplication.run() will automatically execute them using the dataSource Bean configured. So, rename dbscripts.sql to schema.sql .

Drop the database springdb and recreate it.

Run the class and make sure that it ran with out exception and amount got transferred. Go to database and observe that data is present as expected

**10) On BankApplication class u have following 3 annotations :**

```
@Configuration
@ComponentScan
@EnableAutoConfiguration
```

**These 3 annotations can be replaced with a Single convenience Annotation**

```
@SpringBootApplication
```

Replace those 3 annoatations with `@SpringBootApplication`

11) Run BankApplication.java and observe that it still runs the same

12) We dont want to specify the location of property file using

```
@PropertySource("classpath:db.properties")
```

If there is a file with name **aplication.properties** in class path, it will be automatically loaded in to Environment Object.

**SpringApplication** will load properties from **application.properties** files in the following locations and add them to the Spring **Environment**

- A /config subdir of the JVM's working directory.
- The JVM's working directory
- A classpath /config package
- The classpath root

So, remove `@PropertySource("classpath:db.properties")` and rename db.properties as application.properties

13)Run BankApplication.java and observe that it still runs as expected

14) If you dont want the file name to be application.properties, rename it to some name like myapplication.properties . Run the BankApplication.java and observe that it fails

- If you want to change the properties file name, u need to pass --**spring.config.name**=myapplication as command line argument

Run BankApplication.java and pass --**spring.config.name**=myapplication as command line argument. Observe that it works as expected with out exception.

15) Now revert back the file name to application.properties

16) If you dont want to use Environment object, you can configure the DataSource Bean as below :

```
	@Bean
	public DataSource dataSource( @Value("${db.driverclassname}")String
driverClassName,
			@Value("${db.url}")String url, @Value("${db.user}")String username,
			@Value("${db.password}")String password){
		BasicDataSource dataSource= new BasicDataSource();
		dataSource.setDriverClassName(driverClassName);
		dataSource.setUrl(url);
		dataSource.setUsername(username);
		dataSource.setPassword(password);
		return dataSource;
	}
```

Make the above changes to BankApplication and run it (**make sure to remove the command line argument u passed in earlier step. otherwise you will get exception** ). Observe that it runs

as expected.

17) Now, we dont want to set the values of datasource manually. we want them to be automatically populated. So, Modify the DataSource Bean as below :

```
@Bean
@ConfigurationProperties(prefix="db")
public DataSource dataSource( ){
	BasicDataSource dataSource= new BasicDataSource();

	return dataSource;
}
```

Make the above changes to BankApplication and run it. Observe that it runs as expected.

18) Now, we dont want to configure dataSource as a Bean. We want it to be automatically created

So, change the application.properties as shown below :

```
spring.datasource.url=jdbc:mysql://localhost/springdb
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=root
```

The default dataSource prefix is spring.datasource. If these properties are present, data source will be automatically created using these values. u need not configure explicitly.

Now remove dataSource bean configuration in BankApplication class and run it. Observe that it runs as expected.

19) If you dont provide the above properties, Spring application will try to create datasource for embedded database. The corresponding db (like hsqldb) jar has to be present in classpath.

**Congratulations !! You know how to use Spring Boot.**