**Objectives**

* Demonstrate creation of Spring Boot Application
  + Spring initializr, https://start.spring.io, @SpringBootApplication, SpringApplication.run()
    - Ref - https://start.spring.io
* Explain the need and benefits of Spring Boot
  + Makes Java development easy, avoids tedious development steps, reduces development time, avoids writing boilerplate code, provides embedded tomcat server, avoid XML configuration
    - Ref - https://www.journaldev.com/7969/spring-boot-tutorial
* Demonstrate loading bean from spring configuration file
  + Spring configuration xml, spring xml schema spring-beans.xsd, <bean>, id, class, <constructor-arg>, <property>, name, value, ClassPathXmlApplicationContext, ApplicationContext, context.getBean(), singleton scope, prototype scope
    - Ref - https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html
    - IoC Container - https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html#beans
    - Scopes - https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html#beans-factory-scopes
    - Constructor Injection - https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html#beans-constructor-injection
    - Setter method injection - https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html#beans-setter-injection
* Demonstrate inclusion of logging in Spring Boot Application
  + application.properties, logging.level, logging.pattern, server.port, LoggerFactory, Logger, log levels (trace, debug, info, warn, error)
    - Ref - https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-logging.html

**Hands on 1**

**Create a Spring Web Project using Maven**   
  
Follow steps below to create a project: 

1. Go to <https://start.spring.io/>
2. Change Group as “com.cognizant”
3. Change Artifact Id as “spring-learn”
4. Select Spring Boot DevTools and Spring Web
5. Create and download the project as zip
6. Extract the zip in root folder to Eclipse Workspace
7. Build the project using ‘mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456’ command in command line
8. Import the project in Eclipse "File > Import > Maven > Existing Maven Projects > Click Browse and select extracted folder > Finish"
9. Include logs to verify if main() method of SpringLearnApplication.
10. Run the SpringLearnApplication class.

SME to walk through the following aspects related to the project created:

1. src/main/java - Folder with application code
2. src/main/resources - Folder for application configuration
3. src/test/java - Folder with code for testing the application
4. SpringLearnApplication.java - Walkthrough the main() method.
5. Purpose of @SpringBootApplication annotation
6. pom.xml
   1. Walkthrough all the configuration defined in XML file
   2. Open 'Dependency Hierarchy' and show the dependency tree.

**Program:**

**pom.xml:**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>1.0.0</version>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.18</version>

<relativePath/>

</parent>

<properties>

<java.version>11</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**HelloController.java:**

package com.cognizant;

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

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

@RestController

public class HelloController {

@GetMapping("/")

public String hello() {

return "Welcome! Your Spring Boot app is working 🚀";

} }

**SpringLearnApplication.java:**

package com.cognizant;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

**.replit:**

language = "bash"

entrypoint = "run.sh"

run = "./run.sh"

[nix]

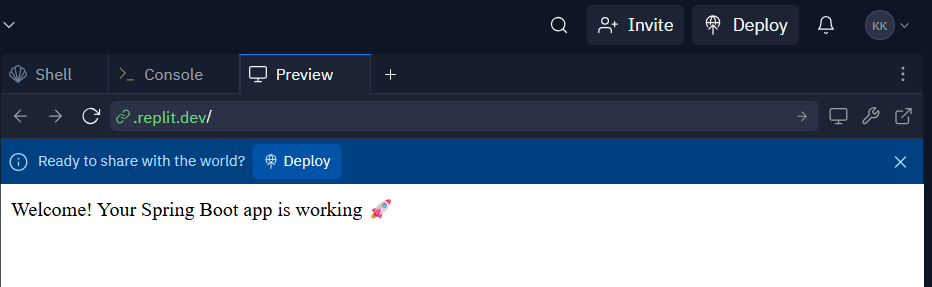
packages = ["maven", "openjdk"]

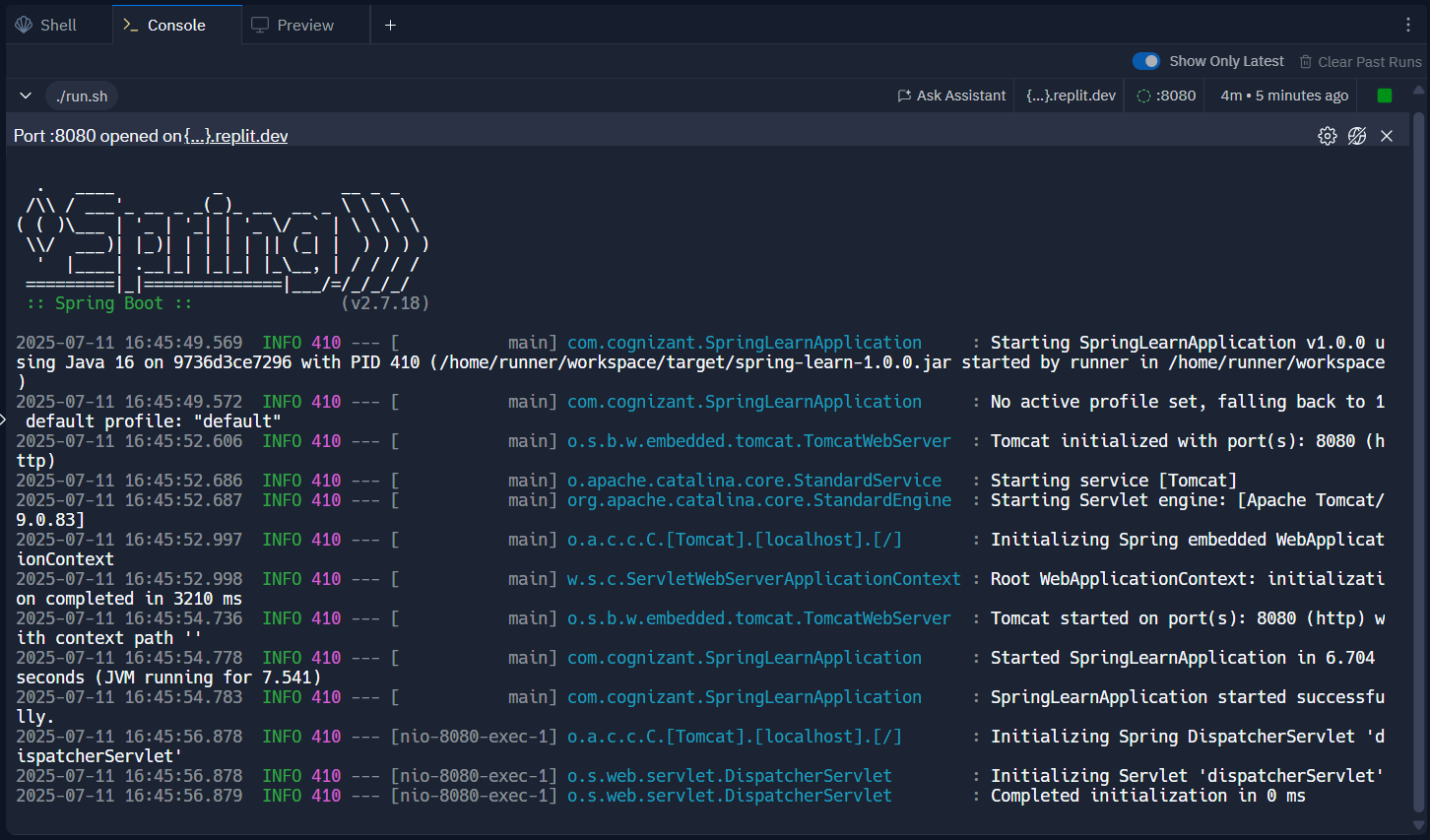
[[ports]]

localPort = 8080

externalPort = 80

**Output:**





**Hands on 2**

**Spring Core – Load SimpleDateFormat from Spring Configuration XML**   
  
SimpleDateFormat with the pattern ‘dd/MM/yyyy’ is created in multiple places of an application. To avoid creation of SimpleDateFormat in multiple places, define a bean in Spring XML Configuration file and retrieve the date.  
  
Follow steps below to implement:

* Create spring configuration file date-format.xml in src/main/resources folder of 'spring-learn' project
* Open https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html#beans-factory-metadata
* Copy the XML defined in the section of previous step URL and paste it into date-format.xml
* Define bean tag in the XML with for date format. Refer code below.

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://www.springframework.org/schema/beans

        https://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="dateFormat" class="java.text.SimpleDateFormat">

<constructor-arg value="dd/MM/yyyy" />

</bean>

</beans>

* Create new method displayDate() in SpringLearnApplication.java
* In displayDate() method create the ApplicationContext. Refer code below:

ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");

* Get the dateFormat using getBean() method. Refer code below.

SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);

* Using the format variable try to parse string '31/12/2018' to Date class and display the result using System.out.println.
* Run the application as 'Java Application' and check the result in console log output.

**Troubleshooting Tips**   
  
If the tomcat port has a conflict and the server is not starting include the below property in application.properties file in src/main/resources folder.

**Program:**

**pom.xml:**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>spring-learn</artifactId>

<version>1.0.0</version>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>6.1.4</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.10.1</version>

<configuration>

<source>${java.version}</source>

<target>${java.version}</target>

</configuration>

</plugin>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>3.1.0</version>

<configuration>

<mainClass>com.example.springlearn.SpringLearnApplication</mainClass>

</configuration>

</plugin>

</plugins>

</build>

</project>

**SpringLearnApplication.java:**

package com.example.springlearn;

import java.text.SimpleDateFormat;

import java.util.Date;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringLearnApplication {

public static void main(String[] args) {

displayDate();

}

public static void displayDate() {

ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");

SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);

try {

Date parsedDate = format.parse("31/12/2018");

System.out.println("Parsed Date: " + parsedDate);

} catch (Exception e) {

e.printStackTrace();

}

}

}

**.replit:**

compile = "javac -classpath .:target/dependency/\* -d . $(find . -type f -name '\*.java')"

run = "mvn compile exec:java -Dexec.mainClass=com.example.springlearn.SpringLearnApplication"

entrypoint = "src/main/java/Main.java"

hidden = ["\*\*/\*.class"]

[packager]

language = "java"

[packager.features]

packageSearch = true

[languages.java]

pattern = "\*\*/\*.java"

[languages.java.languageServer]

start = "jdt-language-server"

[unitTest]

language = "java"

[nix]

channel = "stable-22\_11"

[debugger]

support = true

[debugger.compile]

command = "javac -classpath .:/run\_dir/junit-4.12.jar:target/dependency/\* -g -d . $(find . -type f -name '\*.java')"

[debugger.interactive]

transport = "localhost:0"

connectTimeout = 60

startCommand = "java-debug"

[debugger.interactive.initializeMessage]

command = "initialize"

type = "request"

[debugger.interactive.initializeMessage.arguments]

adapterID = "cppdbg"

clientID = "replit"

clientName = "replit.com"

columnsStartAt1 = true

linesStartAt1 = true

locale = "en-us"

pathFormat = "path"

supportsInvalidatedEvent = true

supportsProgressReporting = true

supportsRunInTerminalRequest = true

supportsVariablePaging = true

supportsVariableType = true

[debugger.interactive.launchMessage]

command = "launch"

type = "request"

[debugger.interactive.launchMessage.arguments]

classPaths = ["."]

mainClass = "Main"

[deployment]

build = [

"javac",

"-classpath",

".:target/dependency/\*",

"-d",

".",

"$(find . -type f -name '\*.java')",

]

run = ["java", "-classpath", ".:target/dependency/\*", "Main"]

deploymentTarget = "cloudrun"

**Output:**

