Spring Core and REST Hands-on:

Hands-on 1:

Create a Spring Web Project using Maven Performed Steps:

Visited <https://start.spring.io>

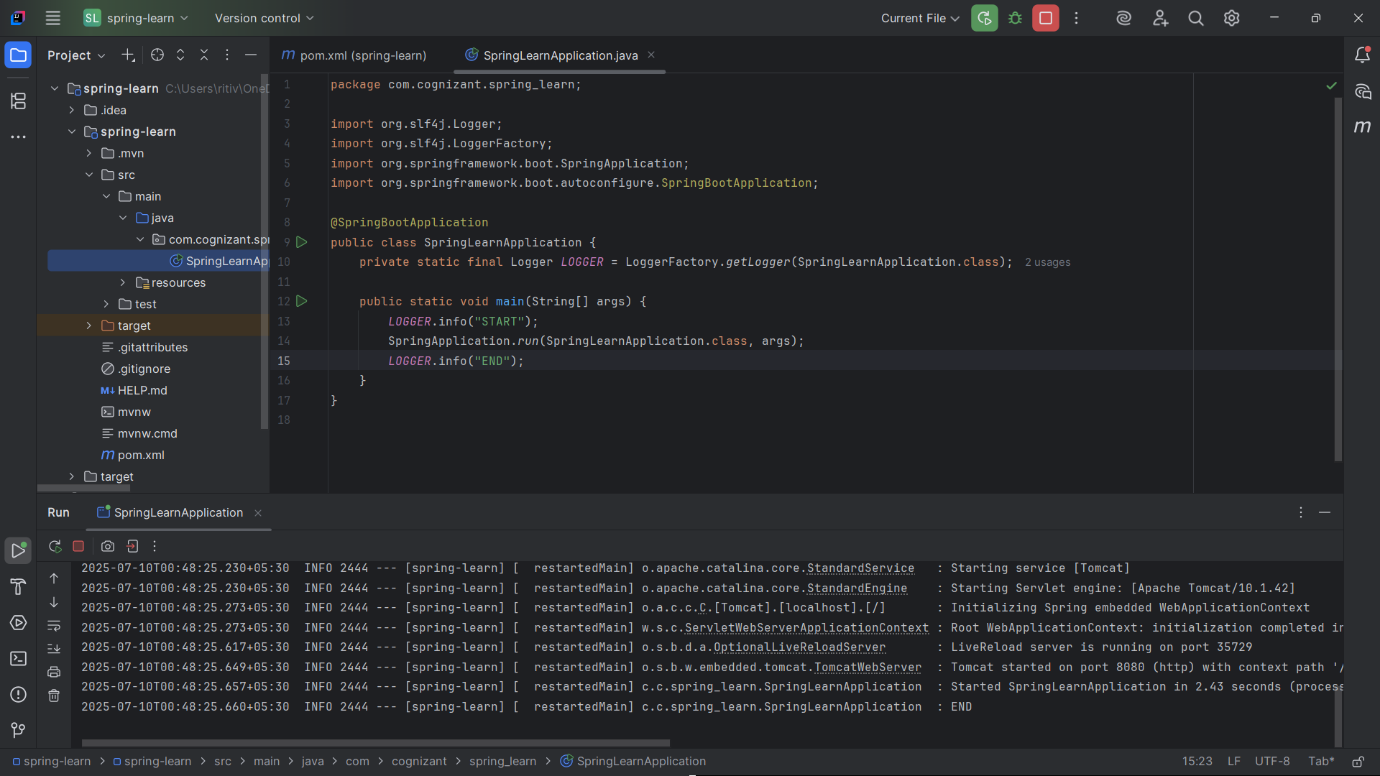
Group: com.cognizant ,

Artifact: spring-learn Dependencies: Spring Web, Spring Boot DevTools

Downloaded and extracted the ZIP

Ran mvn clean package with proxy configuration

Imported into IntelliJ and verified logs using LOGGER.info("START - main()")



Hands-on 2: Load SimpleDateFormat from Spring XML Created:( src/main/resources/date-format.xml)

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

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

</bean>

public static void displayDate() {

LOGGER.info("START - displayDate()");

ApplicationContext context = new ClassPathXmlApplicationContext("date

format.xml");

SimpleDateFormat format = context.getBean("dateFormat",

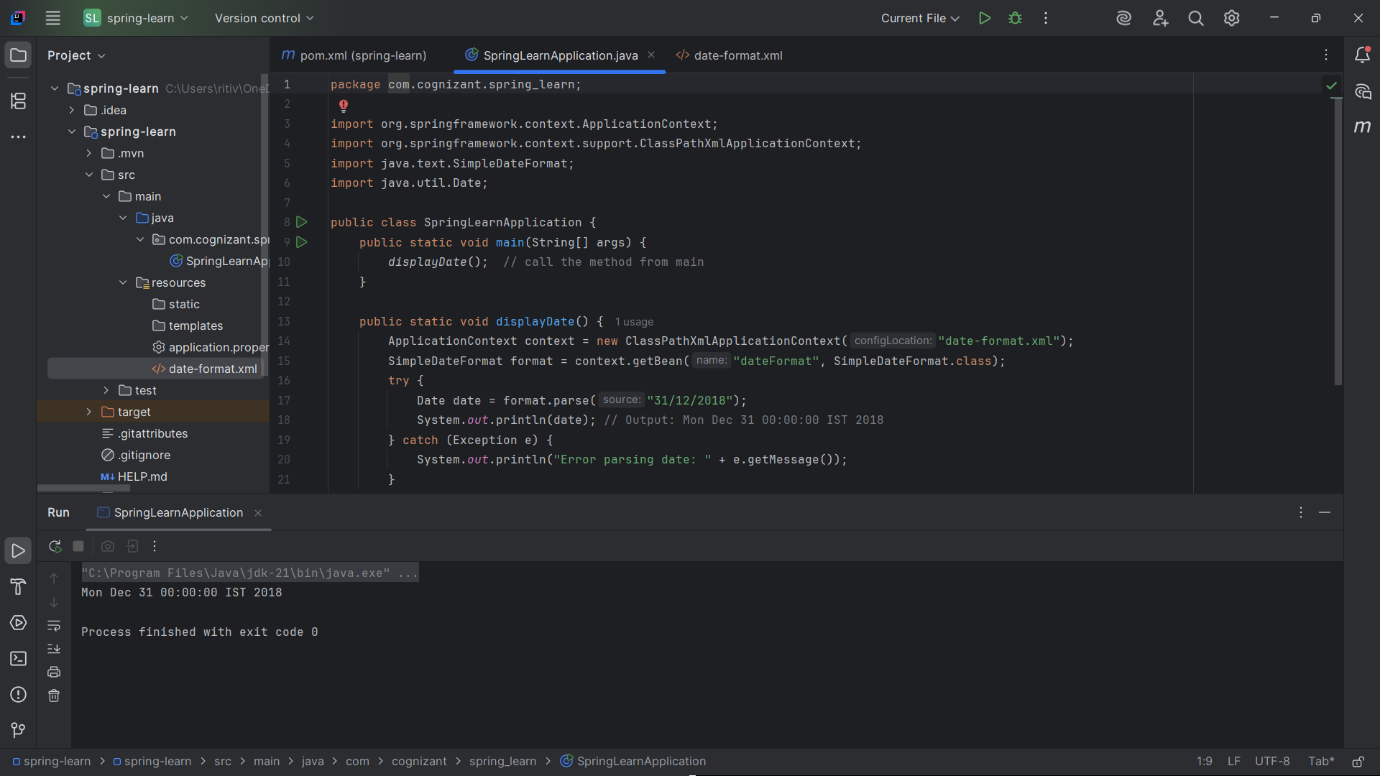
SimpleDateFormat.class);

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

LOGGER.debug("Date: {}", date);

LOGGER.info("END - displayDate()");

}

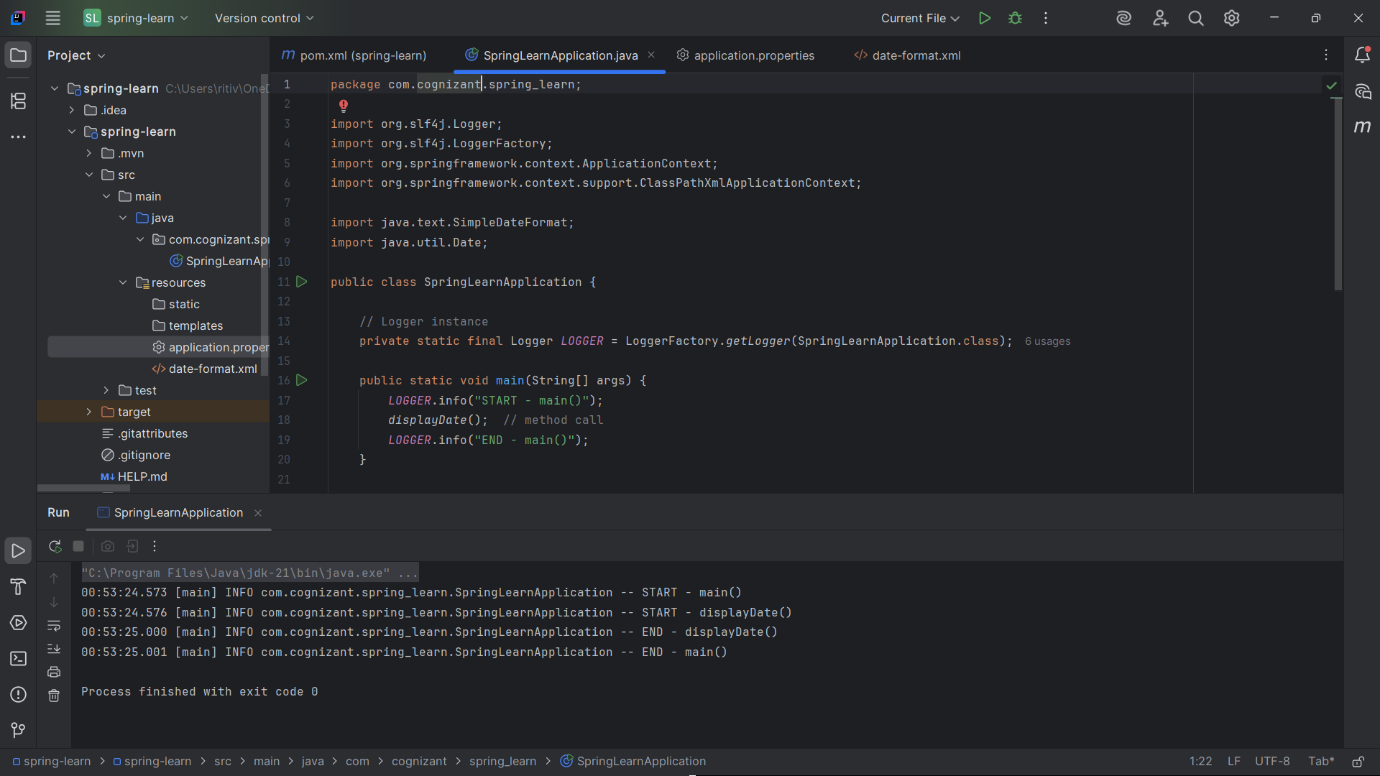


Hands-on 3: Logging in Spring Boot

logging.level.org.springframework=info logging.level.com.cognizant.spring\_learn=debug logging.pattern.console=%d{yyMMdd}|%d{HH:mm:ss.SSS}|%-20.20thread|%5p| %-25.25logger{25}|%25M|%m%n

private static final Logger LOGGER = LoggerFactory.getLogger(SpringLearnApplication.class); • Used LOGGER.info() at method start/end

• Used LOGGER.debug() for intermediate values



Hands-on 4:Load Country from XML

<bean id="country" class="com.cognizant.spring\_learn.Country">

<property name="code" value="IN" />

<property name="name" value="India" />

</bean>

public Country() {

System.out.println("INSIDE CONSTRUCTOR: Country()");

LOGGER.debug("Inside Country Constructor.");

}

public String toString()

{

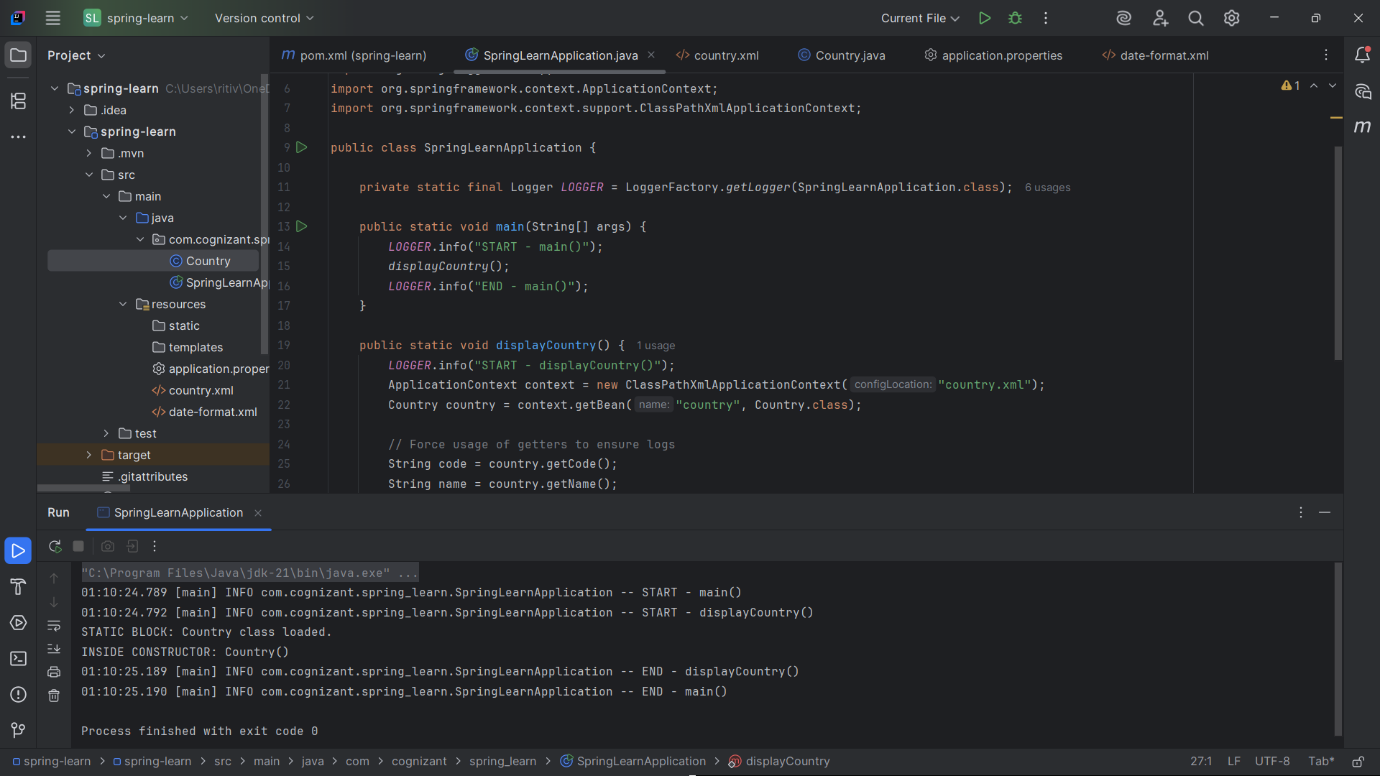
return "Country{" + "code='" + getCode() + '\'' + ", name='" + getName() + '\'' + '}';

}

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

Country country = context.getBean("country", Country.class);

LOGGER.debug("Country: {}", country);



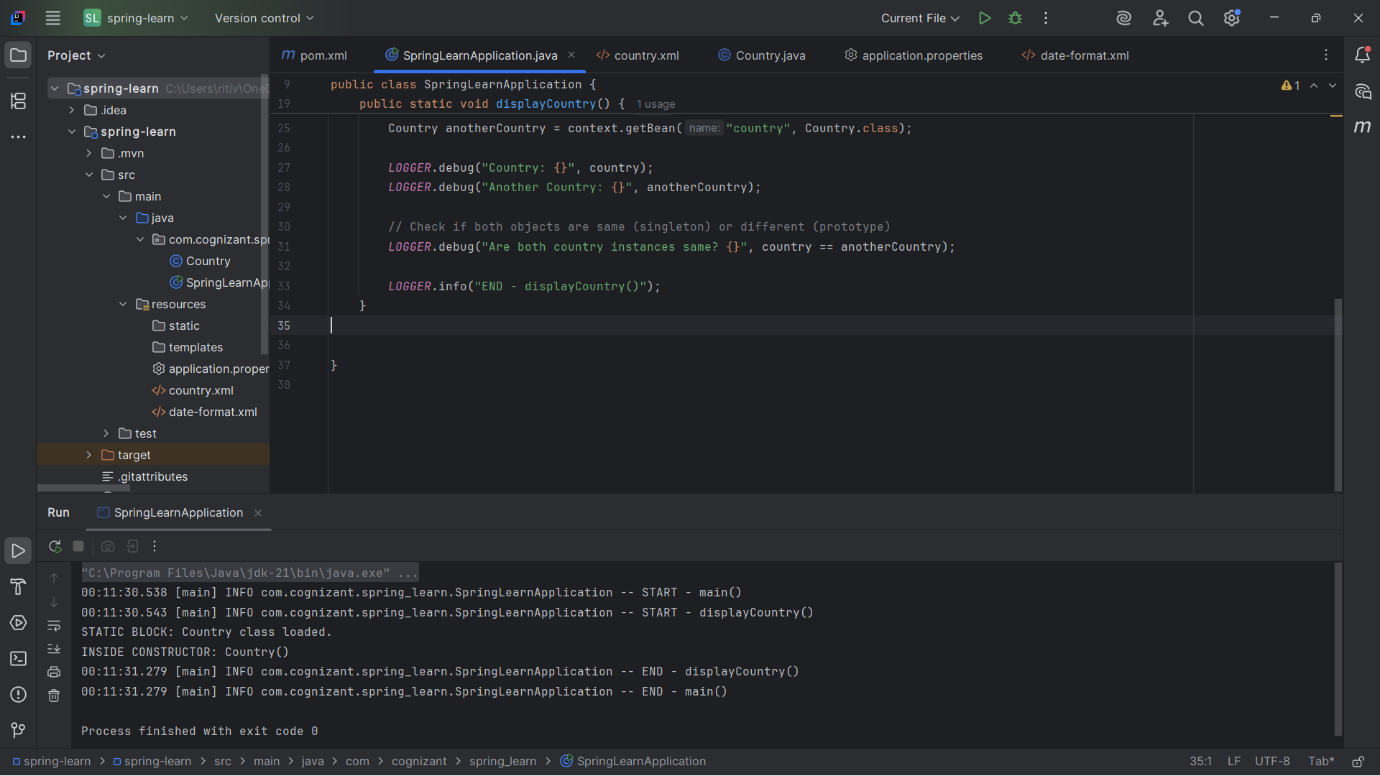
Hands-on 5: Singleton vs Prototype Scope

<bean id="country" class="com.cognizant.spring\_learn.Country" />

Country c1 = context.getBean("country", Country.class);

Country c2 = context.getBean("country", Country.class);

LOGGER.debug("Are both country instances same? {}", c1 == c2);



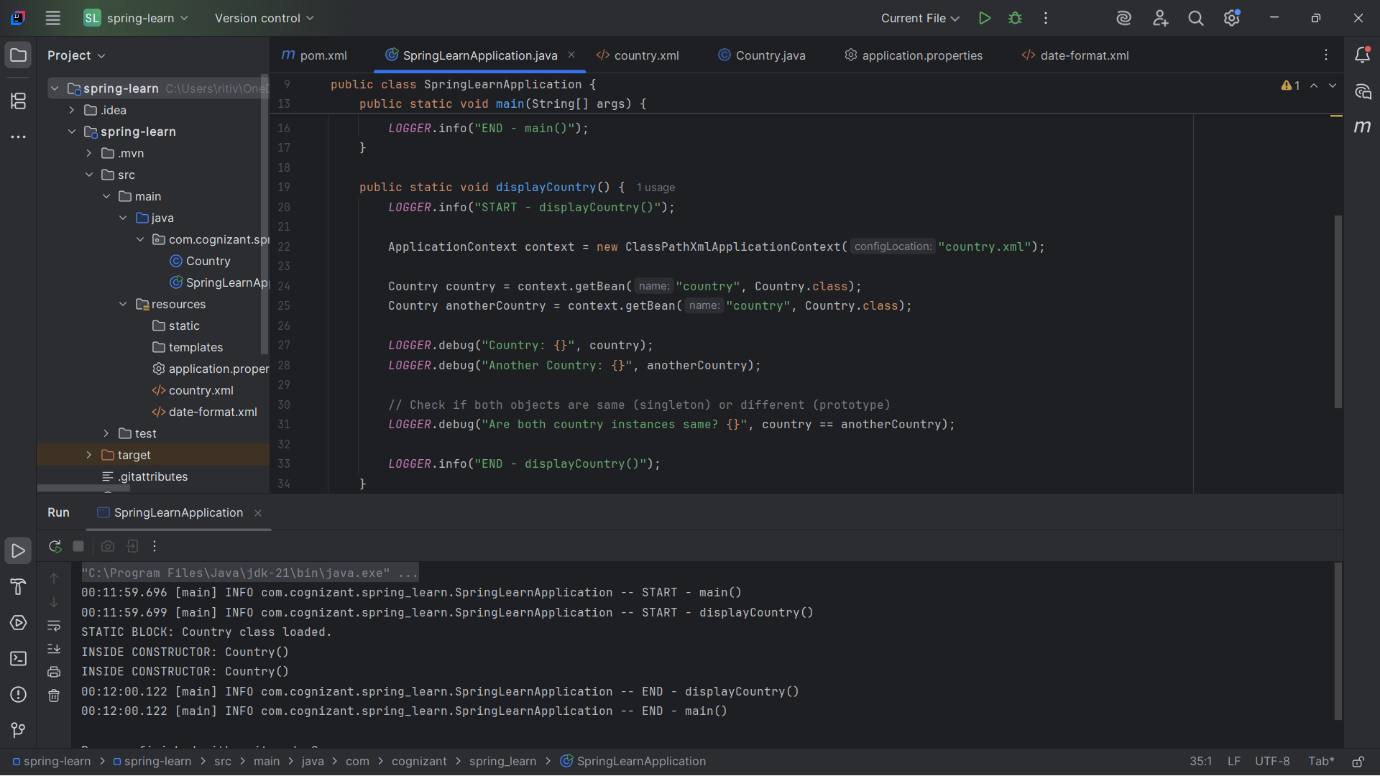
Constructor called once

Same instance: true

Prototype:

<bean id="country" class="com.cognizant.spring\_learn.Country"

scope="prototype" />



Constructor called twice

Different instances: false

Hands-on 6: Load List of Countries from XML

<bean id="in" class="com.cognizant.spring\_learn.Country">

<property name="code" value="IN" />

<property name="name" value="India" />

</bean>

<bean id="us" class="com.cognizant.spring\_learn.Country">

<property name="code" value="US" />

<property name="name" value="United States" />

</bean>

<bean id="de" class="com.cognizant.spring\_learn.Country">

<property name="code" value="DE" />

<property name="name" value="Germany" />

</bean>

<bean id="jp" class="com.cognizant.spring\_learn.Country">

3

<property name="code" value="JP" />

<property name="name" value="Japan" />

</bean>

<bean id="countryList" class="java.util.ArrayList">

<constructor-arg>

<list>

<ref bean="in" />

<ref bean="us" />

<ref bean="de" />

<ref bean="jp" />

</list>

</constructor-arg>

</bean>

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

List<Country> countries = context.getBean("countryList", List.class);

for (Country c : countries) {

LOGGER.debug("Country: {}", c);

}

SME Explanations:

<bean> : Defines a Spring-managed object with id and class attributes

<property> : Used for setter-based dependency injection

ApplicationContext : Spring container for managing beans

ClassPathXmlApplicationContext : Loads XML bean definitions

<list> : Represents a list collection

<ref bean="..."/> : Reference to another defined bean

scope="prototype" : Creates new bean instance on every

getBean() call

scope="singleton" : Creates only one shared instance (default)

LOGGER.debug(...) : Used to print debug-level logs (enabled via application.properties)

