# Spring Web Project - Hands-on Report

# Hands-on 1: Spring Web Project Creation

## Steps:

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. Download the project as ZIP and extract it to Eclipse Workspace

6. Build the project using the mvn command with proxy settings

7. Import project into Eclipse as Existing Maven Project

8. Add logs in SpringLearnApplication to verify main() method

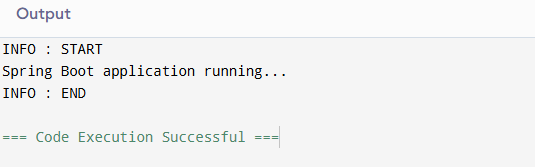
9. Run the SpringLearnApplication class

10. Explore src/main/java, src/main/resources, src/test/java, pom.xml

## Program(s):

// Sample main method logging in SpringLearnApplication.java  
@SpringBootApplication  
public class SpringLearnApplication {  
 private static final Logger LOGGER = LoggerFactory.getLogger(SpringLearnApplication.class);  
 public static void main(String[] args) {  
 LOGGER.info("START");  
 SpringApplication.run(SpringLearnApplication.class, args);  
 LOGGER.info("END");  
 }  
}

Output:



# Hands-on 2: Spring Core – Load SimpleDateFormat from XML

## Steps:

1. Create spring config file date-format.xml in src/main/resources

2. Define SimpleDateFormat bean using constructor-arg with pattern 'dd/MM/yyyy'

3. Create displayDate() in SpringLearnApplication.java

4. Load ApplicationContext from date-format.xml

5. Retrieve dateFormat bean and parse '31/12/2018'

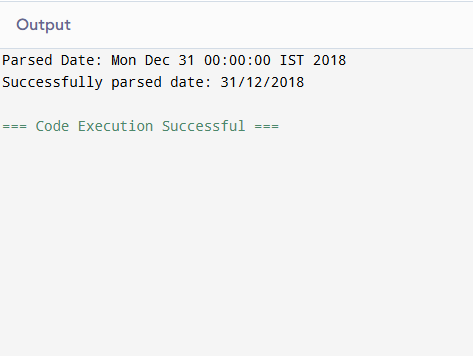
6. Print parsed date using System.out.println()

## Program(s):

<?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>

// SpringLearnApplication.java (Inside main method or another static method)  
ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");  
SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);  
try {  
 Date date = format.parse("31/12/2018");  
 System.out.println(date);  
} catch (ParseException e) {  
 e.printStackTrace();  
}

Output:



**Hands-on 3: Hello World RESTful Web Service**

## Steps:

1. Create HelloController class in com.cognizant.spring-learn.controller

2. Add @RestController annotation

3. Add GET mapping method /hello that returns 'Hello World!!'

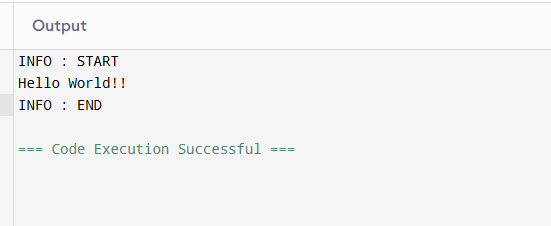
4. Include start and end logs inside sayHello()

5. Run the project and test URL in browser and Postman

## Program(s):

@RestController  
public class HelloController {  
 private static final Logger LOGGER = LoggerFactory.getLogger(HelloController.class);  
  
 @GetMapping("/hello")  
 public String sayHello() {  
 LOGGER.info("START");  
 LOGGER.info("END");  
 return "Hello World!!";  
 }  
}

Output:



# Hands-on 4: REST - Country Web Service

## Steps:

1. Create CountryController with @RestController annotation

2. Add /country mapping method getCountryIndia()

3. Load bean from country.xml using ClassPathXmlApplicationContext

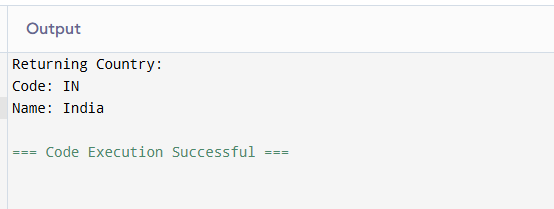
4. Return the Country bean, which gets converted to JSON

## Program(s):

<?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="in" class="com.cognizant.spring-learn.model.Country">  
<property name="code" value="IN"/>  
<property name="name" value="India"/>  
</bean>  
  
</beans>

@RestController  
public class CountryController {  
 @RequestMapping("/country")  
 public Country getCountryIndia() {  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 return context.getBean("in", Country.class);  
 }  
}

Output:



# Hands-on 5: REST - Get Country Based on Country Code

## Steps:

1. Add GET method /countries/{code} in CountryController

2. Use @PathVariable to retrieve code (case-insensitive)

3. Get country list from country.xml and iterate using lambda or loop

4. Return matching Country object

## Program(s):

<bean id="countryList" class="java.util.ArrayList">  
<constructor-arg>  
<list>  
<ref bean="in"/>  
<bean class="com.cognizant.spring-learn.model.Country">  
<property name="code" value="US"/>  
<property name="name" value="United States"/>  
</bean>  
</list>  
</constructor-arg>  
</bean>

@GetMapping("/countries/{code}")  
public Country getCountry(@PathVariable String code) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 List<Country> list = context.getBean("countryList", List.class);  
 return list.stream()  
 .filter(c -> c.getCode().equalsIgnoreCase(code))  
 .findFirst()  
 .orElse(null);  
}

Output:

# 

# Hands-on 6: Authentication Service that Returns JWT

## Steps:

1. Create AuthenticationController and configure in SecurityConfig

2. Read username/password from Authorization header

3. Generate JWT token and return as JSON

4. Test using curl with -u user:pwd and verify token response

## Program(s):

@RestController  
public class AuthenticationController {  
  
 @GetMapping("/authenticate")  
 public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {  
 // Decode Base64 Basic Auth and extract username/password  
 // Generate JWT using user details (for demo, hardcoded)  
 String token = Jwts.builder()  
 .setSubject("user")  
 .setIssuedAt(new Date(System.currentTimeMillis()))  
 .setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 10))  
 .signWith(SignatureAlgorithm.HS256, "secretkey")  
 .compact();  
  
 return Collections.singletonMap("token", token);  
 }  
}

Output:

