Hands-on 1: Create a Spring Web Project using Maven

```
package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import
org.springframework.boot.autoconfigure.SpringBootApplication;

@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");

   }

}
```

Sample output

123 INFO 12345 --- [main] com.cognizant.springlearn.SpringLearnApplication : START

234 INFO 12345 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer: Tomcat started on port(s)

245 INFO 12345 --- [main] o.s.boot.SpringApplication : Started SpringLearnApplication in 1.23 seconds (JVM running for 2.05)

246 INFO 12345 --- [main] com.cognizant.springlearn.SpringLearnApplication : END

Mon Dec 31 00:00:00 IST 2018

Hands-on 2: Load SimpleDateFormat from Spring XML Config:

```
<?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>
```

Update SpringLearnApplication.java

```
import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApplicationConte
xt;
import java.text.SimpleDateFormat;
import java.util.Date;
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);
      displayDate();
      LOGGER.info("END");
   }
   public static void displayDate() {
      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 (Exception e) {
```

```
            e.printStackTrace();
        }
    }
}
```

Sample Output:

123 INFO 12345 --- [main] com.cognizant.springlearn.SpringLearnApplication : START

456 INFO 12345 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer   : Tomcat started on port(s): 8080 (http)

456 INFO 12345 --- [main] o.s.boot.SpringApplication : Started SpringLearnApplication in 1.45 seconds (JVM running for 2.02)

Mon Dec 31 00:00:00 IST 2018

457 INFO 12345 --- [main] com.cognizant.springlearn.SpringLearnApplication : END

1.Hands-on: Create a Basic REST API with Spring Boot:

```
package com.cognizant.springlearn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

@RestController

public class HelloController {
```

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

Sample Output:

```
123  INFO 12345 --- [nio-8083-exec-1]
c.c.s.controller.HelloController : START
124  INFO 12345 --- [nio-8083-exec-1]
c.c.s.controller.HelloController : END
```

2.Country API (Load from XML Bean):

```
<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="country"
class="com.cognizant.springlearn.model.Country">
        <property name="code" value="IN"/>
```

```
        <property name="name" value="India"/>
    </bean>
</beans>
```

Country.java:

```
package com.cognizant.springlearn.model;
public class Country {
    private String code;
    private String name;
    // Getters and Setters
    public String getCode() { return code; }
    public void setCode(String code) { this.code = code; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}
```

CountryController.java

```
package com.cognizant.springlearn.controller;
import com.cognizant.springlearn.model.Country;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApplicationConte
xt;
```

```
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class CountryController {
   private static final Logger LOGGER =
LoggerFactory.getLogger(CountryController.class);
   @RequestMapping("/country")
   public Country getCountryIndia() {
      LOGGER.info("START");
      ApplicationContext context = new
ClassPathXmlApplicationContext("country.xml");
      Country country = context.getBean("country", Country.class);
      LOGGER.info("END");
      return country;
   }
}
```

Sample Output:

```
{
  "code": "IN",
  "name": "India"
}
```

MockMVC Testing (Basic Example):

```
package com.cognizant.springlearn.controller;

import org.junit.jupiter.api.Test;

import org.springframework.beans.factory.annotation.Autowired;

import
org.springframework.boot.test.autoconfigure.web.servlet.AutoConfig
ureMockMvc;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.web.servlet.MockMvc;

import static
org.springframework.test.web.servlet.request.MockMvcRequestBuild
ers.get;

import static
org.springframework.test.web.servlet.result.MockMvcResultMatchers
.*;

@SpringBootTest

@AutoConfigureMockMvc

public class HelloControllerTest {

    @Autowired

    private MockMvc mockMvc;

    @Test

    public void testSayHello() throws Exception {

        mockMvc.perform(get("/hello"))

                .andExpect(status().isOk())

                .andExpect(content().string("Hello World!!"));

    }

}
```

Sample Output:

345 INFO 12345 --- [main] c.c.s.controller.HelloController : START

346 INFO 12345 --- [main] c.c.s.controller.HelloController : END

. HelloControllerTest > testSayHello() PASSED

REST - Get country based on country code:

country.xml (in src/main/resources):

```
<?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="countryList" class="java.util.ArrayList">
      <constructor-arg>
         <list>
            <bean class="com.cognizant.springlearn.model.Country">
               <property name="code" value="IN" />
               <property name="name" value="India" />
            </bean>
            <bean class="com.cognizant.springlearn.model.Country">
               <property name="code" value="US" />
               <property name="name" value="United States" />
            </bean>
```

```
            <bean class="com.cognizant.springlearn.model.Country">
                <property name="code" value="CN" />
                <property name="name" value="China" />
            </bean>
        </list>
    </constructor-arg>
</bean>
</beans>
```

Country.java

```
package com.cognizant.springlearn.model;
public class Country {
    private String code;
    private String name;
    // Getters and Setters
    public String getCode() { return code; }
    public void setCode(String code) { this.code = code; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}
```

CountryService.java:

```
package com.cognizant.springlearn.service;

import com.cognizant.springlearn.model.Country;

import org.springframework.context.ApplicationContext;

import
org.springframework.context.support.ClassPathXmlApplicati
onContext;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CountryService {

   public Country getCountry(String code) {

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

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

      return countryList.stream()

            .filter(country ->
country.getCode().equalsIgnoreCase(code))

            .findFirst()

            .orElse(null); // or throw an exception if country not
found

   }

}
```

CountryController.java:

```
package com.cognizant.springlearn.controller;
import com.cognizant.springlearn.model.Country;
import com.cognizant.springlearn.service.CountryService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
@RestController
public class CountryController {
    private static final Logger LOGGER =
LoggerFactory.getLogger(CountryController.class);
    @Autowired
    private CountryService countryService;
    @GetMapping("/countries/{code}")
    public Country getCountry(@PathVariable String code) {
        LOGGER.info("START");
        Country country = countryService.getCountry(code);
        LOGGER.info("END");
        return country;
    }
}
```

Sample Output:


```
{
  "code": "IN",
  "name": "India"
}
```


3.Static Employee Data Using Spring XML:

```
<beans
xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:schemaLocation="http://www.springframework.org/schem
a/beans
https://www.springframework.org/schema/beans/spring-
beans.xsd">
   <bean id="skill1"
class="com.cognizant.springlearn.model.Skill">
       <property name="id" value="1" />
       <property name="name" value="Java" />
   </bean>
   <bean id="skill2"
class="com.cognizant.springlearn.model.Skill">
       <property name="id" value="2" />
```

```
        <property name="name" value="SQL" />
    </bean>
    <!-- Departments -->
    <bean id="dept1"
class="com.cognizant.springlearn.model.Department">
        <property name="id" value="1" />
        <property name="name" value="HR" />
    </bean>
    <bean id="dept2"
class="com.cognizant.springlearn.model.Department">
        <property name="id" value="2" />
        <property name="name" value="IT" />
    </bean>
    <!-- Employees -->
    <bean id="employeeList" class="java.util.ArrayList">
        <constructor-arg>
            <list>
                <bean
class="com.cognizant.springlearn.model.Employee">
                    <property name="id" value="1" />
                    <property name="name" value="Alice" />
                    <property name="salary" value="60000" />
                    <property name="permanent" value="true" />
                    <property name="department" ref="dept1" />
```

```
            <property name="skillList">
                <list>
                    <ref bean="skill1" />
                    <ref bean="skill2" />
                </list>
            </property>
        </bean>
        <!-- Add 3 more Employee beans similarly -->
    </list>
  </constructor-arg>
 </bean>
</beans>
```

DAO Layer:

```
package com.cognizant.springlearn.dao;
import com.cognizant.springlearn.model.Employee;
import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApplicati
onContext;
import org.springframework.stereotype.Repository;
import java.util.List;
@Repository
```

```
public class EmployeeDao {
    private static List<Employee> EMPLOYEE_LIST;
    public EmployeeDao() {
        ApplicationContext context = new
ClassPathXmlApplicationContext("employee.xml");
        EMPLOYEE_LIST = context.getBean("employeeList",
List.class);
    }
    public List<Employee> getAllEmployees() {
        return EMPLOYEE_LIST;
    }
}
```

Sample Output:

```
123  INFO 12345 --- [nio-8083-exec-1]
c.c.s.controller.EmployeeController : GET /employees called
```

Service Layer:

```
package com.cognizant.springlearn.service;
import com.cognizant.springlearn.dao.EmployeeDao;
import com.cognizant.springlearn.model.Employee;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```
import
org.springframework.transaction.annotation.Transactional;
import java.util.List;
@Service
public class EmployeeService {
    @Autowired
    private EmployeeDao employeeDao;
    @Transactional
    public List<Employee> getAllEmployees() {
        return employeeDao.getAllEmployees();
    }
}
```

Sample output:

```
[
  {
    "id": 1,
    "name": "HR"
  },
  {
```

```
    "id": 2,
    "name": "IT"
  }
]
INFO 12345 --- [nio-8083-exec-2]
c.c.s.controller.DepartmentController    : GET /departments
called
```

REST Controller:

```
package com.cognizant.springlearn.controller;
import com.cognizant.springlearn.model.Employee;
import com.cognizant.springlearn.service.EmployeeService;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.List;
@RestController
public class EmployeeController {
    @Autowired
    private EmployeeService employeeService;
    @GetMapping("/employees")
    public List<Employee> getAllEmployees() {
        return employeeService.getAllEmployees();
    }
}
```

Department REST Service:

```
package com.cognizant.springlearn.dao;

import com.cognizant.springlearn.model.Department;

import org.springframework.context.ApplicationContext;

import
org.springframework.context.support.ClassPathXmlApplicati
onContext;

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public class DepartmentDao {

   private static List<Department> DEPARTMENT_LIST;

   public DepartmentDao() {

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

      DEPARTMENT_LIST =
context.getBean("departmentList", List.class); // Ensure
departmentList bean exists

   }

   public List<Department> getAllDepartments() {

      return DEPARTMENT_LIST;

   }

}
```

```
package com.cognizant.springlearn.service;
import com.cognizant.springlearn.dao.DepartmentDao;
import com.cognizant.springlearn.model.Department;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class DepartmentService {
    @Autowired
    private DepartmentDao departmentDao
    public List<Department> getAllDepartments() {
        return departmentDao.getAllDepartments();
    }
}
```

```
package com.cognizant.springlearn.controller;
import com.cognizant.springlearn.model.Department;
import com.cognizant.springlearn.service.DepartmentService;
import
org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.web.bind.annotation.*;
import java.util.List;
@RestController
public class DepartmentController {
    @Autowired
    private DepartmentService departmentService
    @GetMapping("/departments")
    public List<Department> getAllDepartments() {
        return departmentService.getAllDepartments();
    }
}
```

Sample output:

```
getEmployees(): Observable<Employee[]> {
  return
this.http.get<Employee[]>('http://localhost:8083/employees');
}
GET http://localhost:8083/countries/in
{
  "code": "IN",
  "name": "India"
}
```

5. Create authentication service that returns JWT:

```
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.9.1</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

2. Security Configuration (SecurityConfig.java):

```
package com.cognizant.springlearn.config;
import
org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.web.builders.
HttpSecurity;
import
org.springframework.security.config.annotation.web.configur
ation.WebSecurityConfigurerAdapter;

@Configuration
```

```
public class SecurityConfig extends
WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws
Exception {
        http.csrf().disable()
.authorizeRequests().antMatchers("/authenticate").permitAll()
            .anyRequest().authenticated();
    }
}

package com.cognizant.springlearn.controller;
import com.cognizant.springlearn.util.JwtUtil;
import org.springframework.http.HttpHeaders;
import org.springframework.http.ResponseEntity;
import org.springframework.util.Base64Utils;
import org.springframework.web.bind.annotation.*;
@RestController
public class AuthenticationController {
    @GetMapping("/authenticate")
    public ResponseEntity<?>
authenticate(@RequestHeader(HttpHeaders.AUTHORIZATI
ON) String authHeader) {
```

```
        if (authHeader != null && authHeader.startsWith("Basic
")) {
            String base64Credentials =
authHeader.substring("Basic ".length());
            String credentials = new
String(Base64Utils.decodeFromString(base64Credentials));
            String[] values = credentials.split(":", 2);
            String username = values[0];
            String password = values[1];
            if ("user".equals(username) &&
"pwd".equals(password)) {
                String token = JwtUtil.generateToken(username);
                return ResponseEntity.ok().body("{\"token\":\"" +
token + "\"}");
            } else {
                return ResponseEntity.status(401).body("Invalid
credentials");
            }
        }
        return ResponseEntity.status(400).body("Missing
Authorization Header");
    }
}
```

```
package com.cognizant.springlearn.util;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import java.util.Date;
public class JwtUtil {
    private static final String SECRET_KEY =
"your_secret_key";
    private static final long EXPIRATION_TIME = 1000 * 60
* 60; // 1 hour
    public static String generateToken(String username) {
        return Jwts.builder()
                .setSubject(username)
                .setIssuedAt(new
Date(System.currentTimeMillis()))
                .setExpiration(new
Date(System.currentTimeMillis() + EXPIRATION_TIME))
                .signWith(SignatureAlgorithm.HS256,
SECRET_KEY)
                .compact();
    }
}
```

Sample Output:

{
  "token": "eyJhbGciOiJIUzI1NiJ9..."
}

```
private static final Logger LOGGER =
LoggerFactory.getLogger(AuthenticationController.class);
LOGGER.info("Authenticating user: {}", username);
```