[Introduction](javascript:void(0);)

**Objectives**

* Explain the need and benefit of ORM
  + ORM (Object-Relational Mapping), makes it easier to develop code that interacts with database, abstracts the database system, transactionality
    - ORM Pros and Cons - https://blog.bitsrc.io/what-is-an-orm-and-why-you-should-use-it-b2b6f75f5e2a
    - What is ORM? - https://en.wikipedia.org/wiki/Object-relational\_mapping

* Demonstrate the need and benefit of Spring Data JPA
  + Evolution of ORM solutions, Hibernate XML Configuration, Hibernate Annotation Configuration, Spring Data JPA, Hibernate benefits, open source, light weight, database independent query
    - With H2 in memory database - https://www.mkyong.com/spring-boot/spring-boot-spring-data-jpa/
    - With MySQL - https://www.mkyong.com/spring-boot/spring-boot-spring-data-jpa-mysql-example/
    - XML Configuration Example -https://www.tutorialspoint.com/hibernate/hibernate\_examples.htm
    - Hibernate Configuration Example -https://www.tutorialspoint.com/hibernate/hibernate\_annotations.htm

* Explain about core objects of hibernate framework
  + Session Factory, Session, Transaction Factory, Transaction, Connection Provider
    - Hibernate Architecture Reference - https://www.tutorialspoint.com/hibernate/hibernate\_architecture.htm

* Explain ORM implementation with Hibernate XML Configuration and Annotation Configuration
  + XML Configuration - persistence class, mapping xml, configuration xml, loading hibernate configuration xml file; Annotation Configuration - persistence class, @Entity, @Table, @Id, @Column, hibernate configuration xml file Loading hibernate configuration and interacting with database get the session factory, open session, begin transaction, commit transaction, close session
    - XML Configuration Example - https://www.tutorialspoint.com/hibernate/hibernate\_examples.htm
    - Hibernate Configuration Example - https://www.tutorialspoint.com/hibernate/hibernate\_annotations.htm

* Explain the difference between Java Persistence API, Hibernate and Spring Data JPA
  + JPA (Java Persistence API), JPA is a specification (JSR 338), JPA does not have implementation, Hibernate is one of the implementation for JPA, Hibernate is a ORM tool, Spring Data JPA is an abstraction above Hibernate to remove boiler plate code when persisting data using Hibernate.
    - Difference between Spring Data JPA and Hibernate - https://dzone.com/articles/what-is-the-difference-between-hibernate-and-sprin-1
    - Intro to JPA - https://www.javaworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html

* Demonstrate implementation of DML using Spring Data JPA on a single database table
  + Hibernate log configuration and ddl-auto configuration, JpaRepsitory.findById(), defining Query Methods, JpaRespository.save(), JpaRepository.deleteById()
    - Spring Data JPA Ref Repository methods - https://docs.spring.io/spring-data/jpa/docs/2.2.0.RELEASE/reference/html/#repositories.core-concepts
    - Query methods - https://docs.spring.io/spring-data/jpa/docs/2.2.0.RELEASE/reference/html/#repositories.query-methods

**Spring Data JPA - Quick Example**   
  
**Software Pre-requisites**

* MySQL Server 8.0
* MySQL Workbench 8
* Eclipse IDE for Enterprise Java Developers 2019-03 R
* Maven 3.6.2

**Create a Eclipse Project using Spring Initializr**

* Go to <https://start.spring.io/>
* Change Group as “com.fis”
* Change Artifact Id as “orm-learn”
* In Options > Description enter "Demo project for Spring Data JPA and Hibernate"
* Click on menu and select "Spring Boot DevTools", "Spring Data JPA" and "MySQL Driver"
* Click Generate and download the project as zip
* Extract the zip in root folder to Eclipse Workspace
* Import the project in Eclipse "File > Import > Maven > Existing Maven Projects > Click Browse and select extracted folder > Finish"
* Create a new schema "ormlearn" in MySQL database. Execute the following commands to open MySQL client and create schema.

> mysql -u root -p

mysql> create schema ormlearn;

* In orm-learn Eclipse project, open src/main/resources/application.properties and include the below database and log configuration.

# Spring Framework and application log

logging.level.org.springframework=info

logging.level.com.fis=debug

# Hibernate logs for displaying executed SQL, input and output

logging.level.org.hibernate.SQL=trace

logging.level.org.hibernate.type.descriptor.sql=trace

# Log pattern

logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n

# Database configuration

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=root

# Hibernate configuration

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

* Build the project using ‘mvn clean package’ command in command line
* Include logs for verifying if main() method is called.

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

  LOGGER.info("Inside main");

}

* Execute the OrmLearnApplication and check in log if main method is called.

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. OrmLearnApplication.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.

**Country table creation**

* Create a new table country with columns for code and name. For sample, let us insert one country with values 'IN' and 'India' in this table.

create table country(co\_code varchar(2) primary key, co\_name varchar(50));

* Insert couple of records into the table

insert into country values ('IN', 'India');

insert into country values ('US', 'United States of America');

**Persistence Class - com.fis.orm-learn.model.Country**

* Open Eclipse with orm-learn project
* Create new package com.fis.orm-learn.model
* Create Country.java, then generate getters, setters and toString() methods.
* Include @Entity and @Table at class level
* Include @Column annotations in each getter method specifying the column name.

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name="country")

public class Country {

  @Id

    @Column(name="code")

    private String code;

    @Column(name="name")

    private String name;

// getters and setters

  // toString()

}

*Notes:*

* @Entity is an indicator to Spring Data JPA that it is an entity class for the application
* @Table helps in defining the mapping database table
* @Id helps is defining the primary key
* @Column helps in defining the mapping table column

**Repository Class - com.fis.orm-learn.CountryRepository**

* Create new package com.fis.orm-learn.repository
* Create new interface named CountryRepository that extends JpaRepository<Country, String>
* Define @Repository annotation at class level

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.fis.ormlearn.model.Country;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

**Service Class - com.fis.orm-learn.service.CountryService**

* Create new package com.fis.orm-learn.service
* Create new class CountryService
* Include @Service annotation at class level
* Autowire CountryRepository in CountryService
* Include new method getAllCountries() method that returns a list of countries.
* Include @Transactional annotation for this method
* In getAllCountries() method invoke countryRepository.findAll() method and return the result

**Testing in OrmLearnApplication.java**

* Include a static reference to CountryService in OrmLearnApplication class

private static CountryService countryService;

* Define a test method to get all countries from service.

    private static void testGetAllCountries() {

        LOGGER.info("Start");

        List<Country> countries = countryService.getAllCountries();

        LOGGER.debug("countries={}", countries);

        LOGGER.info("End");

    }

* Modify SpringApplication.run() invocation to set the application context and the CountryService reference from the application context.

        ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

        countryService = context.getBean(CountryService.class);

        testGetAllCountries();

* Execute main method to check if data from ormlearn database is retrieved.

**Hibernate XML Config implementation walk through**   
  
Provide explanation on the sample Hibernate implementation available in the link below:  
https://www.tutorialspoint.com/hibernate/hibernate\_examples.htm  
  
Explanation Topics

* Explain how object to relational database mapping done in hibernate xml configuration file
* Explain about following aspects of implementing the end to end operations in Hibernate:
  + SessionFactory
  + Session
  + Transaction
  + beginTransaction()
  + commit()
  + rollback()
  + session.save()
  + session.createQuery().list()
  + session.get()
  + session.delete()

**Hibernate Annotation Config implementation walk through**   
  
Provide explanation on the sample Hibernate implementation available in the link below:  
https://www.tutorialspoint.com/hibernate/hibernate\_annotations.htm  
  
Explanation Topics

* Explain how object to relational database mapping done in persistence class file Employee
* Explain about following aspects of implementing the end to end operations in Hibernate:
  + @Entity
  + @Table
  + @Id
  + @GeneratedValue
  + @Column
  + Hibernate Configuration (hibernate.cfg.xml)
    - Dialect
    - Driver
    - Connection URL
    - Username
    - Password

**Difference between JPA, Hibernate and Spring Data JPA**   
  
Java Persistence API (JPA)

* JSR 338 Specification for persisting, reading and managing data from Java objects
* Does not contain concrete implementation of the specification
* Hibernate is one of the implementation of JPA

Hibernate

* ORM Tool that implements JPA

Spring Data JPA

* Does not have JPA implementation, but reduces boiler plate code
* This is another level of abstraction over JPA implementation provider like Hibernate
* Manages transactions

**Refer code snippets below on how the code compares between Hibernate and Spring Data JPA  
Hibernate**

   /\* Method to CREATE an employee in the database \*/

   public Integer addEmployee(Employee employee){

      Session session = factory.openSession();

      Transaction tx = null;

      Integer employeeID = null;

      try {

         tx = session.beginTransaction();

         employeeID = (Integer) session.save(employee);

         tx.commit();

      } catch (HibernateException e) {

         if (tx != null) tx.rollback();

         e.printStackTrace();

      } finally {

         session.close();

      }

      return employeeID;

   }

**Spring Data JPA**  
EmployeeRespository.java

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

EmployeeService.java

@Autowire

  private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

  employeeRepository.save(employee);

  }

​​​​​​​   
  
**Reference Links:**   
<https://dzone.com/articles/what-is-the-difference-between-hibernate-and-sprin-1>   
<https://www.javaworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html>

**Implement services for managing Country**   
  
An application requires for features to be implemented with regards to country. These features needs to be supported by implementing them as service using Spring Data JPA.

* Find a country based on country code
* Add new country
* Update country
* Delete country
* Find list of countries matching a partial country name

Before starting the implementation of the above features, there are few configuration and data population that needs to be incorporated. Please refer each topic below and implement the same.   
  
**Explanation for Hibernate table creation configuration**

* Moreover the ddl-auto defines how hibernate behaves if a specific table or column is not present in the database.
  + create - drops existing tables data and structure, then creates new tables
  + validate - check if the table and columns exist or not, throws an exception if a matching table or column is not found
  + update - if a table does not exists, it creates a new table; if a column does not exists, it creates a new column
  + create-drop - creates the table, once all operations are completed, the table is dropped

# Hibernate ddl auto (create, create-drop, update, validate)

spring.jpa.hibernate.ddl-auto=validate

Populate country table

* Delete all the records in Country table and then use the below script to create the actual list of all countries in our world.

insert into country (co\_code, co\_name) values ("AF", "Afghanistan");

insert into country (co\_code, co\_name) values ("AL", "Albania");

insert into country (co\_code, co\_name) values ("DZ", "Algeria");

insert into country (co\_code, co\_name) values ("AS", "American Samoa");

insert into country (co\_code, co\_name) values ("AD", "Andorra");

insert into country (co\_code, co\_name) values ("AO", "Angola");

insert into country (co\_code, co\_name) values ("AI", "Anguilla");

insert into country (co\_code, co\_name) values ("AQ", "Antarctica");

insert into country (co\_code, co\_name) values ("AG", "Antigua and Barbuda");

insert into country (co\_code, co\_name) values ("AR", "Argentina");

insert into country (co\_code, co\_name) values ("AM", "Armenia");

insert into country (co\_code, co\_name) values ("AW", "Aruba");

insert into country (co\_code, co\_name) values ("AU", "Australia");

insert into country (co\_code, co\_name) values ("AT", "Austria");

insert into country (co\_code, co\_name) values ("AZ", "Azerbaijan");

insert into country (co\_code, co\_name) values ("BS", "Bahamas");

insert into country (co\_code, co\_name) values ("BH", "Bahrain");

insert into country (co\_code, co\_name) values ("BD", "Bangladesh");

insert into country (co\_code, co\_name) values ("BB", "Barbados");

insert into country (co\_code, co\_name) values ("BY", "Belarus");

insert into country (co\_code, co\_name) values ("BE", "Belgium");

insert into country (co\_code, co\_name) values ("BZ", "Belize");

insert into country (co\_code, co\_name) values ("BJ", "Benin");

insert into country (co\_code, co\_name) values ("BM", "Bermuda");

insert into country (co\_code, co\_name) values ("BT", "Bhutan");

insert into country (co\_code, co\_name) values ("BO", "Bolivia, Plurinational State of");

insert into country (co\_code, co\_name) values ("BQ", "Bonaire, Sint Eustatius and Saba");

insert into country (co\_code, co\_name) values ("BA", "Bosnia and Herzegovina");

insert into country (co\_code, co\_name) values ("BW", "Botswana");

insert into country (co\_code, co\_name) values ("BV", "Bouvet Island");

insert into country (co\_code, co\_name) values ("BR", "Brazil");

insert into country (co\_code, co\_name) values ("IO", "British Indian Ocean Territory");

insert into country (co\_code, co\_name) values ("BN", "Brunei Darussalam");

insert into country (co\_code, co\_name) values ("BG", "Bulgaria");

insert into country (co\_code, co\_name) values ("BF", "Burkina Faso");

insert into country (co\_code, co\_name) values ("BI", "Burundi");

insert into country (co\_code, co\_name) values ("KH", "Cambodia");

insert into country (co\_code, co\_name) values ("CM", "Cameroon");

insert into country (co\_code, co\_name) values ("CA", "Canada");

insert into country (co\_code, co\_name) values ("CV", "Cape Verde");

insert into country (co\_code, co\_name) values ("KY", "Cayman Islands");

insert into country (co\_code, co\_name) values ("CF", "Central African Republic");

insert into country (co\_code, co\_name) values ("TD", "Chad");

insert into country (co\_code, co\_name) values ("CL", "Chile");

insert into country (co\_code, co\_name) values ("CN", "China");

insert into country (co\_code, co\_name) values ("CX", "Christmas Island");

insert into country (co\_code, co\_name) values ("CC", "Cocos (Keeling) Islands");

insert into country (co\_code, co\_name) values ("CO", "Colombia");

insert into country (co\_code, co\_name) values ("KM", "Comoros");

insert into country (co\_code, co\_name) values ("CG", "Congo");

insert into country (co\_code, co\_name) values ("CD", "Congo, the Democratic Republic of the");

insert into country (co\_code, co\_name) values ("CK", "Cook Islands");

insert into country (co\_code, co\_name) values ("CR", "Costa Rica");

insert into country (co\_code, co\_name) values ("HR", "Croatia");

insert into country (co\_code, co\_name) values ("CU", "Cuba");

insert into country (co\_code, co\_name) values ("CW", "Curaçao");

insert into country (co\_code, co\_name) values ("CY", "Cyprus");

insert into country (co\_code, co\_name) values ("CZ", "Czech Republic");

insert into country (co\_code, co\_name) values ("CI", "Côte d'Ivoire");

insert into country (co\_code, co\_name) values ("DK", "Denmark");

insert into country (co\_code, co\_name) values ("DJ", "Djibouti");

insert into country (co\_code, co\_name) values ("DM", "Dominica");

insert into country (co\_code, co\_name) values ("DO", "Dominican Republic");

insert into country (co\_code, co\_name) values ("EC", "Ecuador");

insert into country (co\_code, co\_name) values ("EG", "Egypt");

insert into country (co\_code, co\_name) values ("SV", "El Salvador");

insert into country (co\_code, co\_name) values ("GQ", "Equatorial Guinea");

insert into country (co\_code, co\_name) values ("ER", "Eritrea");

insert into country (co\_code, co\_name) values ("EE", "Estonia");

insert into country (co\_code, co\_name) values ("ET", "Ethiopia");

insert into country (co\_code, co\_name) values ("FK", "Falkland Islands (Malvinas)");

insert into country (co\_code, co\_name) values ("FO", "Faroe Islands");

insert into country (co\_code, co\_name) values ("FJ", "Fiji");

insert into country (co\_code, co\_name) values ("FI", "Finland");

insert into country (co\_code, co\_name) values ("FR", "France");

insert into country (co\_code, co\_name) values ("GF", "French Guiana");

insert into country (co\_code, co\_name) values ("PF", "French Polynesia");

insert into country (co\_code, co\_name) values ("TF", "French Southern Territories");

insert into country (co\_code, co\_name) values ("GA", "Gabon");

insert into country (co\_code, co\_name) values ("GM", "Gambia");

insert into country (co\_code, co\_name) values ("GE", "Georgia");

insert into country (co\_code, co\_name) values ("DE", "Germany");

insert into country (co\_code, co\_name) values ("GH", "Ghana");

insert into country (co\_code, co\_name) values ("GI", "Gibraltar");

insert into country (co\_code, co\_name) values ("GR", "Greece");

insert into country (co\_code, co\_name) values ("GL", "Greenland");

insert into country (co\_code, co\_name) values ("GD", "Grenada");

insert into country (co\_code, co\_name) values ("GP", "Guadeloupe");

insert into country (co\_code, co\_name) values ("GU", "Guam");

insert into country (co\_code, co\_name) values ("GT", "Guatemala");

insert into country (co\_code, co\_name) values ("GG", "Guernsey");

insert into country (co\_code, co\_name) values ("GN", "Guinea");

insert into country (co\_code, co\_name) values ("GW", "Guinea-Bissau");

insert into country (co\_code, co\_name) values ("GY", "Guyana");

insert into country (co\_code, co\_name) values ("HT", "Haiti");

insert into country (co\_code, co\_name) values ("HM", "Heard Island and McDonald Islands");

insert into country (co\_code, co\_name) values ("VA", "Holy See (Vatican City State)");

insert into country (co\_code, co\_name) values ("HN", "Honduras");

insert into country (co\_code, co\_name) values ("HK", "Hong Kong");

insert into country (co\_code, co\_name) values ("HU", "Hungary");

insert into country (co\_code, co\_name) values ("IS", "Iceland");

insert into country (co\_code, co\_name) values ("IN", "India");

insert into country (co\_code, co\_name) values ("ID", "Indonesia");

insert into country (co\_code, co\_name) values ("IR", "Iran, Islamic Republic of");

insert into country (co\_code, co\_name) values ("IQ", "Iraq");

insert into country (co\_code, co\_name) values ("IE", "Ireland");

insert into country (co\_code, co\_name) values ("IM", "Isle of Man");

insert into country (co\_code, co\_name) values ("IL", "Israel");

insert into country (co\_code, co\_name) values ("IT", "Italy");

insert into country (co\_code, co\_name) values ("JM", "Jamaica");

insert into country (co\_code, co\_name) values ("JP", "Japan");

insert into country (co\_code, co\_name) values ("JE", "Jersey");

insert into country (co\_code, co\_name) values ("JO", "Jordan");

insert into country (co\_code, co\_name) values ("KZ", "Kazakhstan");

insert into country (co\_code, co\_name) values ("KE", "Kenya");

insert into country (co\_code, co\_name) values ("KI", "Kiribati");

insert into country (co\_code, co\_name) values ("KP", "Democratic People's Republic of Korea");

insert into country (co\_code, co\_name) values ("KR", "Republic of Korea");

insert into country (co\_code, co\_name) values ("KW", "Kuwait");

insert into country (co\_code, co\_name) values ("KG", "Kyrgyzstan");

insert into country (co\_code, co\_name) values ("LA", "Lao People's Democratic Republic");

insert into country (co\_code, co\_name) values ("LV", "Latvia");

insert into country (co\_code, co\_name) values ("LB", "Lebanon");

insert into country (co\_code, co\_name) values ("LS", "Lesotho");

insert into country (co\_code, co\_name) values ("LR", "Liberia");

insert into country (co\_code, co\_name) values ("LY", "Libya");

insert into country (co\_code, co\_name) values ("LI", "Liechtenstein");

insert into country (co\_code, co\_name) values ("LT", "Lithuania");

insert into country (co\_code, co\_name) values ("LU", "Luxembourg");

insert into country (co\_code, co\_name) values ("MO", "Macao");

insert into country (co\_code, co\_name) values ("MK", "Macedonia, the Former Yugoslav Republic of");

insert into country (co\_code, co\_name) values ("MG", "Madagascar");

insert into country (co\_code, co\_name) values ("MW", "Malawi");

insert into country (co\_code, co\_name) values ("MY", "Malaysia");

insert into country (co\_code, co\_name) values ("MV", "Maldives");

insert into country (co\_code, co\_name) values ("ML", "Mali");

insert into country (co\_code, co\_name) values ("MT", "Malta");

insert into country (co\_code, co\_name) values ("MH", "Marshall Islands");

insert into country (co\_code, co\_name) values ("MQ", "Martinique");

insert into country (co\_code, co\_name) values ("MR", "Mauritania");

insert into country (co\_code, co\_name) values ("MU", "Mauritius");

insert into country (co\_code, co\_name) values ("YT", "Mayotte");

insert into country (co\_code, co\_name) values ("MX", "Mexico");

insert into country (co\_code, co\_name) values ("FM", "Micronesia, Federated States of");

insert into country (co\_code, co\_name) values ("MD", "Moldova, Republic of");

insert into country (co\_code, co\_name) values ("MC", "Monaco");

insert into country (co\_code, co\_name) values ("MN", "Mongolia");

insert into country (co\_code, co\_name) values ("ME", "Montenegro");

insert into country (co\_code, co\_name) values ("MS", "Montserrat");

insert into country (co\_code, co\_name) values ("MA", "Morocco");

insert into country (co\_code, co\_name) values ("MZ", "Mozambique");

insert into country (co\_code, co\_name) values ("MM", "Myanmar");

insert into country (co\_code, co\_name) values ("NA", "Namibia");

insert into country (co\_code, co\_name) values ("NR", "Nauru");

insert into country (co\_code, co\_name) values ("NP", "Nepal");

insert into country (co\_code, co\_name) values ("NL", "Netherlands");

insert into country (co\_code, co\_name) values ("NC", "New Caledonia");

insert into country (co\_code, co\_name) values ("NZ", "New Zealand");

insert into country (co\_code, co\_name) values ("NI", "Nicaragua");

insert into country (co\_code, co\_name) values ("NE", "Niger");

insert into country (co\_code, co\_name) values ("NG", "Nigeria");

insert into country (co\_code, co\_name) values ("NU", "Niue");

insert into country (co\_code, co\_name) values ("NF", "Norfolk Island");

insert into country (co\_code, co\_name) values ("MP", "Northern Mariana Islands");

insert into country (co\_code, co\_name) values ("NO", "Norway");

insert into country (co\_code, co\_name) values ("OM", "Oman");

insert into country (co\_code, co\_name) values ("PK", "Pakistan");

insert into country (co\_code, co\_name) values ("PW", "Palau");

insert into country (co\_code, co\_name) values ("PS", "Palestine, State of");

insert into country (co\_code, co\_name) values ("PA", "Panama");

insert into country (co\_code, co\_name) values ("PG", "Papua New Guinea");

insert into country (co\_code, co\_name) values ("PY", "Paraguay");

insert into country (co\_code, co\_name) values ("PE", "Peru");

insert into country (co\_code, co\_name) values ("PH", "Philippines");

insert into country (co\_code, co\_name) values ("PN", "Pitcairn");

insert into country (co\_code, co\_name) values ("PL", "Poland");

insert into country (co\_code, co\_name) values ("PT", "Portugal");

insert into country (co\_code, co\_name) values ("PR", "Puerto Rico");

insert into country (co\_code, co\_name) values ("QA", "Qatar");

insert into country (co\_code, co\_name) values ("RO", "Romania");

insert into country (co\_code, co\_name) values ("RU", "Russian Federation");

insert into country (co\_code, co\_name) values ("RW", "Rwanda");

insert into country (co\_code, co\_name) values ("RE", "Réunion");

insert into country (co\_code, co\_name) values ("BL", "Saint Barthélemy");

insert into country (co\_code, co\_name) values ("SH", "Saint Helena, Ascension and Tristan da Cunha");

insert into country (co\_code, co\_name) values ("KN", "Saint Kitts and Nevis");

insert into country (co\_code, co\_name) values ("LC", "Saint Lucia");

insert into country (co\_code, co\_name) values ("MF", "Saint Martin (French part)");

insert into country (co\_code, co\_name) values ("PM", "Saint Pierre and Miquelon");

insert into country (co\_code, co\_name) values ("VC", "Saint Vincent and the Grenadines");

insert into country (co\_code, co\_name) values ("WS", "Samoa");

insert into country (co\_code, co\_name) values ("SM", "San Marino");

insert into country (co\_code, co\_name) values ("ST", "Sao Tome and Principe");

insert into country (co\_code, co\_name) values ("SA", "Saudi Arabia");

insert into country (co\_code, co\_name) values ("SN", "Senegal");

insert into country (co\_code, co\_name) values ("RS", "Serbia");

insert into country (co\_code, co\_name) values ("SC", "Seychelles");

insert into country (co\_code, co\_name) values ("SL", "Sierra Leone");

insert into country (co\_code, co\_name) values ("SG", "Singapore");

insert into country (co\_code, co\_name) values ("SX", "Sint Maarten (Dutch part)");

insert into country (co\_code, co\_name) values ("SK", "Slovakia");

insert into country (co\_code, co\_name) values ("SI", "Slovenia");

insert into country (co\_code, co\_name) values ("SB", "Solomon Islands");

insert into country (co\_code, co\_name) values ("SO", "Somalia");

insert into country (co\_code, co\_name) values ("ZA", "South Africa");

insert into country (co\_code, co\_name) values ("GS", "South Georgia and the South Sandwich Islands");

insert into country (co\_code, co\_name) values ("SS", "South Sudan");

insert into country (co\_code, co\_name) values ("ES", "Spain");

insert into country (co\_code, co\_name) values ("LK", "Sri Lanka");

insert into country (co\_code, co\_name) values ("SD", "Sudan");

insert into country (co\_code, co\_name) values ("SR", "Suriname");

insert into country (co\_code, co\_name) values ("SJ", "Svalbard and Jan Mayen");

insert into country (co\_code, co\_name) values ("SZ", "Swaziland");

insert into country (co\_code, co\_name) values ("SE", "Sweden");

insert into country (co\_code, co\_name) values ("CH", "Switzerland");

insert into country (co\_code, co\_name) values ("SY", "Syrian Arab Republic");

insert into country (co\_code, co\_name) values ("TW", "Taiwan, Province of China");

insert into country (co\_code, co\_name) values ("TJ", "Tajikistan");

insert into country (co\_code, co\_name) values ("TZ", "Tanzania, United Republic of");

insert into country (co\_code, co\_name) values ("TH", "Thailand");

insert into country (co\_code, co\_name) values ("TL", "Timor-Leste");

insert into country (co\_code, co\_name) values ("TG", "Togo");

insert into country (co\_code, co\_name) values ("TK", "Tokelau");

insert into country (co\_code, co\_name) values ("TO", "Tonga");

insert into country (co\_code, co\_name) values ("TT", "Trinidad and Tobago");

insert into country (co\_code, co\_name) values ("TN", "Tunisia");

insert into country (co\_code, co\_name) values ("TR", "Turkey");

insert into country (co\_code, co\_name) values ("TM", "Turkmenistan");

insert into country (co\_code, co\_name) values ("TC", "Turks and Caicos Islands");

insert into country (co\_code, co\_name) values ("TV", "Tuvalu");

insert into country (co\_code, co\_name) values ("UG", "Uganda");

insert into country (co\_code, co\_name) values ("UA", "Ukraine");

insert into country (co\_code, co\_name) values ("AE", "United Arab Emirates");

insert into country (co\_code, co\_name) values ("GB", "United Kingdom");

insert into country (co\_code, co\_name) values ("US", "United States");

insert into country (co\_code, co\_name) values ("UM", "United States Minor Outlying Islands");

insert into country (co\_code, co\_name) values ("UY", "Uruguay");

insert into country (co\_code, co\_name) values ("UZ", "Uzbekistan");

insert into country (co\_code, co\_name) values ("VU", "Vanuatu");

insert into country (co\_code, co\_name) values ("VE", "Venezuela, Bolivarian Republic of");

insert into country (co\_code, co\_name) values ("VN", "Viet Nam");

insert into country (co\_code, co\_name) values ("VG", "Virgin Islands, British");

insert into country (co\_code, co\_name) values ("VI", "Virgin Islands, U.S.");

insert into country (co\_code, co\_name) values ("WF", "Wallis and Futuna");

insert into country (co\_code, co\_name) values ("EH", "Western Sahara");

insert into country (co\_code, co\_name) values ("YE", "Yemen");

insert into country (co\_code, co\_name) values ("ZM", "Zambia");

insert into country (co\_code, co\_name) values ("ZW", "Zimbabwe");

insert into country (co\_code, co\_name) values ("AX", "Åland Islands");

Refer subsequent hands on exercises to implement the features related to country.

**Find a country based on country code** 

* Create new exception class CountryNotFoundException in com.fis.spring-learn.service.exception
* Create new method findCountryByCode() in CountryService with @Transactional annotation
* In findCountryByCode() method, perform the following steps:
  + Method signature

@Transactional

public Country findCountryByCode(String countryCode) throws CountryNotFoundException

* Get the country based on findById() built in method

Optional<Country> result = countryRepository.findById(countryCode);

* From the result, check if a country is found. If not found, throw CountryNotFoundException

if (!result.isPresent())

* Use get() method to return the country fetched.

Country country = result.get();

* Include new test method in OrmLearnApplication to find a country based on country code and compare the country name to check if it is valid.

    private static void getAllCountriesTest() {

        LOGGER.info("Start");

        Country country = countryService.findCountryByCode("IN");

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

        LOGGER.info("End");

    }

* Invoke the above method in main() method and test it.

**NOTE:** Explain the importance of @Transactional annotation. Spring takes care of creating the Hibernate session and manages the transactionality when executing the service method.

**Add a new country** 

* Create new method in CountryService.

@Transactional

public void addCountry(Country country)

* Invoke save() method of repository to get the country added.

countryRepository.save(country)

* Include new testAddCountry() method in OrmLearnApplication. Perform steps below:
  + Create new instance of country with a new code and name
  + Call countryService.addCountry() passing the country created in the previous step.
  + Invoke countryService.findCountryByCode() passing the same code used when adding a new country
  + Check in the database if the country is added

**Update a country based on code** 

* Create a new method updateCountry() in CountryService with parameters code and name. Annotate this method with @Transactional. Implement following steps in this method.
  + Get the reference of the country using findById() method in repository
  + In the country reference obtained, update the name of country using setter method
  + Call countryRepository.save() method to update the name
* Include new test method in OrmLearnApplication, which invokes updateCountry() method in CountryService passing a country's code and different name for the country.
* Check in database table if name is modified.

**Delete a country based on code** 

* Create new method deleteCountry() in CountryService. Annotate this method with @Transactional.
* In deleteCountry() method call deleteById() method of repository.
* Include new test method in OrmLearnApplication with following steps
  + Call the delete method based on the country code during the add country hands on
* Check in database if the country is deleted

#### [Angular Integration and Mockito](javascript:void(0);)

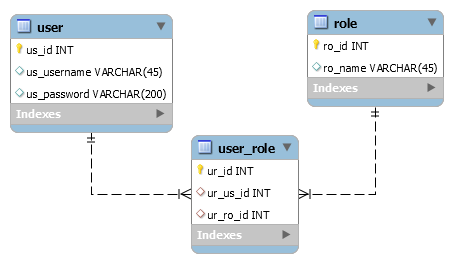
**Objectives**

* Demonstrate implementation of spring security integrated with repository to get user details
  + UserDetailsService, UserDetail, loadUserByUsername(), UsernameNotFoundException
    - Reference - https://docs.spring.io/spring-security/site/docs/5.2.1.BUILD-SNAPSHOT/reference/htmlsingle/#tech-userdetailsservice
* Demonstrate implementation of isolated testing of service using Mockito
  + Importance of isolated testing, MockitoJUnitRunner, Mockito.mock(), when(), thenReturn()
    - Spring Boot with Mockito - https://www.springboottutorial.com/spring-boot-unit-testing-and-mocking-with-mockito-and-junit
* Demonstrate code coverage report generation on unit test cases executed
  + Eclipse, ECL Emma, understanding the color coding
    - EMMA Reference - https://www.eclemma.org/
* Demonstrate end to end integration with Angular, RESTful Web Services and Spring Data JPA
  + Implement an event triggered in angular that executes a restful web service, which in turn invoke Spring Data JPA Repository that gets or persists data in database

Reference Code : FSE-ORM-029Complexity : Level1

**Integrate Spring Security with database**   
  
In "Spring RESTful Web Services" module, we implemented Spring Security with in memory database. In this hands on we will integrate Spring Security to use Spring Data JPA repository that manages the user details.  
  
**Prepare the Schema and Repository classes**

* Create relevant tables for storing user and role in ormlearn schema. (Refer the ER diagram in the end)
* Create classes User and Role with necessary many-to-many mapping
* Define roleList in User with eager fetch
* Create UserRepository with inclusion of a Query Method 'findByUsername' to get an user based on username.
* In user table create two users one for user and one for admin
* Make the password as $2a$10$R/lZJuT9skteNmAku9Y7aeutxbOKstD5xE5bHOf74M2PHZipyt3yK, which is the bcrypt encoding of the text 'pwd'
* Test if findByUsername() method works fine and retrieves the user and roles

​​​​​​​

**AppUser.java**  
This new bean class has to be implement UserDetails class of spring security framework, so that spring framework can read user details from this bean. Refer steps below to implement.

* Create bean class AppUser to hold the user details. This needs to be implement UserDetails class of spring security framework, so that spring framework can read user details from this bean. Refer steps below to implement:
* Right click on com.fis.spring-learn.security > New > Class
* Enter "Name" as "AppUser"
* In "Interfaces" click "Add.." and select "org.springframework.security.core.userdetails.UserDetails"
* Click Finish
* The above steps will create AppUser class and populate the code with all the methods that needs to be implemented.
* Include below instance variables in this class:

    private User user; // entity reference

    private Collection<? extends GrantedAuthority> authorities; // to store role details

* Create a constructor with user as parameter. Obtain the roles from the user and convert them to authorities. Refer code below for authorities conversion that needs to be included in the constructor.

        this.authorities = user.getRoleList().stream()

                .map(role -> new SimpleGrantedAuthority(role.getName())).collect(Collectors.toList());

* The above code converts the list of roles into a list of SimpleGrantedAuthority.
* Return authorities in getAuthorities() method
* Return user.getPassword() in getPassword() method
* Return true in all the below methods:
  + isAccountNonExpired()
  + isAccountNonLocked()
  + isCredentialsNonExpired()
  + isEnabled()

**AppUserDetailsService.java**

* Create a new class AppUserDetailsService in com.fis.spring-learn.security package.
* This class has to implement org.springframework.security.core.userdetails.UserDetailsService, so that spring security framework uses this class to retrieve user details.
* Autowire UserRepository
* Implement loadUserByUsername() method

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException

* Import UsernameNotFoundException from org.springframework.security.core.userdetails;
* Invoke findByUsername() method from UserRepository and get the user.
* If user is null throw UsernameNotFoundException
* If user is not null then create an instance of AppUser passing user obtained from repository.
* Return the AppUser reference.
* Test the execution of loadUserByUsername() to see if user details are retrieved along with roles.

**SecurityConfig.java**

* Autowire AppUserDetailsService
* In configure() method, comment out the in memory definition and bind spring framework to use the appUserDetailsService for authentication.

auth.userDetailsService(appUserDetailsService).passwordEncoder(passwordEncoder());

* Test the authentication web service using curl or postman.
* Reference Code : FSE-ORM-030Complexity : Level1
* **A Unit Test should be self sufficient**   
    
  As the name "Unit Testing" suggests, it is supposed to test a specific method in isolation without worrying about the dependencies.  
    
  Let us try to understand this with an example. Consider a Service method that calls a method in Repository. The service method is dependent on repository and the repository is dependent on database. When we have to test a service method, we should test it without the dependency on repository.  
    
  A good unit test should be isolated. Avoid dependencies such as environment settings, register values, or databases.  
    
  Let us consider the loadUserByUsername() method we implemented recently.  
    
  In this code below, find out the lines that has dependency on repository and database.
* public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
* LOGGER.info("Start");
* LOGGER.debug("UserRepository:{}", userRepository);
* User user = userRepository.findByUsername(username);
* LOGGER.debug("User:{}", user);
* if (user == null) {
* throw new UsernameNotFoundException(username);
* }
* LOGGER.info("End");
* return new AppPrincipalUser(user);
* }
* SME to walkthrough each line of the above code and identify the lines that has dependency.  
    
  In the subsequent hands on we will identify how to avoid the dependency using mockito.

Reference Code : FSE-ORM-031Complexity : Level1

**Implement integrated testing of loadUserByUsername() method** 

* Open spring-learn application in Eclipse
* Create a new package com.fis.spring-learn.service in src/test/java
* Create a new class UserDetailsServiceTest
* Define the following annotations at class level. This ensures that the class has spring application context.

@RunWith(SpringRunner.class)

@SpringBootTest

* Autowire UserDetailsService and implement the below test method. This method validates if the password returned is as per what is available in the database.

    @Test

    public void testLoadByUserName() {

        LOGGER.info("Start");

        UserDetails user = userDetailsService.loadUserByUsername("usr");

        String expected = "$2a$10$R/lZJuT9skteNmAku9Y7aeutxbOKstD5xE5bHOf74M2PHZipyt3yK";

        assertEquals(expected, user.getPassword());

        LOGGER.info("End");

    }

* Execute this as JUnit Test Case by right clicking on this test class and selecting Run > JUnit Test
* Check the logs to see that the service calls repository and it retrieves the data from database
* Ask yourself the following questions:
  + Is this method executed in isolation?
  + Is this test unit testing?

This kind of testing is integration testing and not unit testing.  
  
In the next hands on we will see how to isolate the testing using Mockito.

Reference Code : FSE-ORM-032Complexity : Level1

**Implement isolated testing with Mockito**   
  
Refer the code of the method to be tested:

    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

        LOGGER.info("Start");

        LOGGER.debug("UserRepository:{}", userRepository);

        User user = userRepository.findByUsername(username);

        LOGGER.debug("User:{}", user);

        if (user == null) {

            throw new UsernameNotFoundException(username);

        }

        LOGGER.info("End");

        return new AppPrincipalUser(user);

    }

Our unit test cases should ensure that all the lines and all possible scenarios are tested.  
  
The above method can be tested in two scenarios:

* The method findByUsername() returns a valid
* The method findByUsername() returns null due to non-availability of user in the database

**Implement the steps below to test the first scenario:**

* Create a new test class UserDetailsServiceMockTest in src/test/java, com.fis.spring-learn.service package
* Annotate @RunWith with MockitoJUnitRunner.class (import org.mockito.junit.MockitoJUnitRunner)
* Annotate @SpringBootTest
* Include test method mockTestLoadUserByUsername()
* Implement following steps in the test method.
* Using Mockit.mock() method create a mock of the repository

UserRepository repository = Mockito.mock(UserRepository.class);

* Using below line of code define what happens when findByUsername() method in repository is called. This is how we are isolating the dependency.

when(repository.findByUsername("usr")).thenReturn(createUser());

* Due to the above defintion the database call will not happen, we need to simulate the return of User as part of this call. We achieve this by using a new createUser() method that takes care of creating a user instance with necessary password and role that is expected. Implement this method within the test class.
* Create a new constructor in AppUserDetailsService that accepts UserRepository as input and sets the userRepository instance variable.
* Create AppUserDetailsService instance using the mocked repository

AppUserDetailsService service = new AppUserDetailsService(repository); // repository refers to the mock repository created

* Invoke the method we want to test. When this method is executed, the repository's find method will not be invoke, but mockito will just return whatever is returned by createUser() method.

UserDetails user = service.loadUserByUsername("usr");

* Include the lines below to assert if the password matches.

String expected = "$2a$10$R/lZJuT9skteNmAku9Y7aeutxbOKstD5xE5bHOf74M2PHZipyt3yK";

assertEquals(expected, user.getPassword());

* Run the class as JUnit test and check if the test passes.

**Implement the mockito test for the second scenario when user is returned as null**

* Mock the findByUsername() to return null
* Pass the test case if UsernameNotFoundException is thrown. This can be implemented by asserting true in the catch block of UsernameNotFoundException handling. Include return after assert.
* Assert it as false after catch block

Reference Code : FSE-ORM-033Complexity : Level1

**Code Coverage of JUnit Testing**   
  
Following steps below to check the test coverage of JUnit testing:

* Right click on test class
* Select "Coverage As" > "JUnit Test"
* This executes the following actions:
  + Executes the unit test cases
  + Marks each line as red and green to denote if a line was executed or not. Open various files of your choice and check the relevance of red and green lines.
  + Generates a coverage report that can be seen in a new view "Coverage". Maximize the "Coverage" view, expand the packages to view the class wise code coverage percentage.

Reference Code : FSE-ORM-034Complexity : Level1

**Integrate employee web services with Spring Data JPA**   
  
Hope you all remember the following features implemented as part of angular hands on:

* Display list of employees
* Clicking edit link on an employee will open employee form with populated values
* Clicking save on edit form to persist the employee details
* Clicking delete link on an employee has to delete the employee

Currently the above implementations are done with in memory static data.  
  
Using repository and appropriate beans implement this with repository and service.   
  
Test in the angular application to see if the features work end to end.

Reference Code : FSE-ORM-035Complexity : Level1

**Implement end to end country search feature**   
  
Create a new component and route in angular application for searching country.  
  
This component should have a text box and a div in the bottom of the text box. This div will list the countries that matches with the keyword entered in text box. The countries within the div needs to be displayed line by line.  
  
Create necessary service in angular application that invokes the respective REST API created for returning the country list based on keyword.  
This REST API needs to be called on the key press event of the search text box.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1) | |  | | --- | | **Which class in Spring Security Framework is used to define role?** | | |  |  |  | | --- | --- | --- | |  | **1)** | **GrantedAuthority** | |  | 2) | Authority | |  | 3) | Role | |  | 4) | RoleAuthority | | |
| 2) | |  | | --- | | **Identify the class that helps to get the current spring security configuration** | | |  |  |  | | --- | --- | --- | |  | 1) | SecurityConfigHolder | |  | **2)** | **SecurityContextHolder** | |  | 3) | SecurityContext | |  | 4) | SecurityConfig | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3) | |  | | --- | | **What needs to be passed as input parameter of thenReturn() method?** | | |  |  |  | | --- | --- | --- | |  | **1)** | **The actual object that has to be returned by the mocked method** | |  | 2) | The object that needs to be mocked | |  | 3) | The class that needs to be mocked | |  | 4) | The method that needs to be mocked | | |
| 4) | |  | | --- | | **A unit test should be self sufficient** | | |  |  |  | | --- | --- | --- | |  | 1) | FALSE | |  | **2)** | **TRUE** | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5) | |  | | --- | | **A method in CountryService needs to be unit tested. In this scenario, identify the**  **right code that will mock the class.** | | |  |  |  | | --- | --- | --- | |  | **1)** | **CountryService service = Mockito.mock(CountryService.class);** | |  | 2) | CountryService service = Mockito.createMock(CountryService.class); | |  | 3) | Mockito.mock(CountryService.class); | |  | 4) | CountryService service = Mockito.generateMock(CountryService.class); | | |
| 6) | |  | | --- | | **Which method in UserDetailsService needs to be implemented, so that Spring**  **Security Framework can get the user details of a specific user?** | | |  |  |  | | --- | --- | --- | |  | **1)** | **loadUserByUsername()** | |  | 2) | loadUserByUserName() | |  | 3) | loadByUserName() | |  | 4) | loadByUsername() | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7) | |  | | --- | | **Select all aspects applicable for the statement "Unit Test should be self sufficient"** | | |  |  |  | | --- | --- | --- | |  | **1)** | **Avoid dependencies on file system** | |  | **2)** | **Avoid dependencies on database** | |  | 3) | Avoid dependencies within methods and classes | |  | **4)** | **Avoid dependencies on environment settings** | | |

#### [Jenkins](javascript:void(0);)

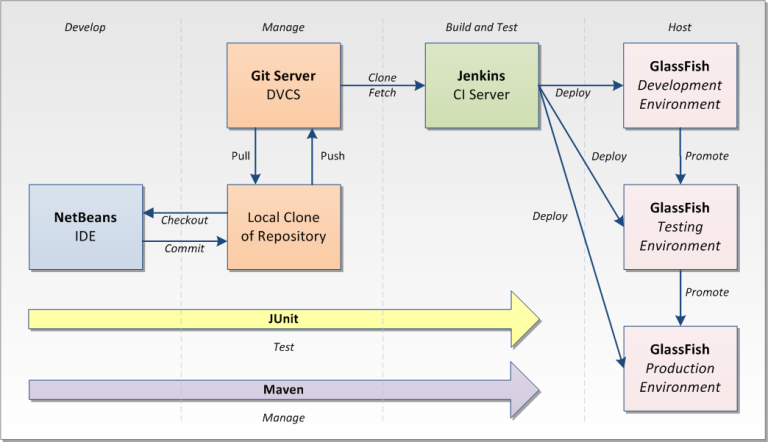
**Objectives**

* Demonstrate setting up Jenkins to get code from GitLab and build the maven project
  + About Jenkins, build automation, Jenkins installation and user setup, plugins for GitLab and Maven, configure GitLab with API authentication token, configure GitLab SSL Certificate, configure git.exe path and maven path, setup maven project, build project
    - Jenkins - https://jenkins.io/doc/
    - Jenkins Java - https://jenkins.io/solutions/java/

Reference Code : FSE-ORM-040Complexity : Level1

**About Jenkins**

* Open source automation server
* Automate build and deployement of a project
* Refer diagram below which has a sample continuous integration server setup:
  + In our scenario it is git bash client instead of NetBeans IDE
  + Git Server is the gitlab server https://code.fis.com
  + Ideally in real time environment Jenkins should be a separate server. For our learning purpose, we will use the local desktop as build server environment.
  + In our case it is "tomcat" server instead of "GlassFish"

SME to explain the diagram in detail.   
  
[ *Picture Courtesy*:  <https://programmaticponderings.com/2013/11/13/building-a-deployment-pipeline-using-git-maven-jenkins-and-glassfish-part-2-of-2/>]   


Reference Code : FSE-ORM-036Complexity : Level1

**Jenkins Installation**

Follow steps below to install Jenkins:

**Downloads**

* SME to download following files from SharePoint and share them with the learners:
* SME to download the following files from the SharePoint path GenCSharePath > GenC Shared Content > OBL - Java FSE > software:
  + jenkins.war
  + jenkins-plugins.zip

**Start Jenkins**

* In command prompt go to the folder where jenkins.war is downloaded and execute the below command to start Jenkins

java -jar jenkins.war

* On execution of the above command the log rolls. Look for the generated admin password in the log. copy the password and have it future reference
* Open http://localhost:8080 in browser
* This will open the screen to enter the admin password.
* Get the password from where you had stored or get the password from initialAdminPassword file in "C:\Users\<YOUR\_EMP\_ID>\.jenkins\secrets" folder.
* Enter the admin password
* Select the "Select plugins to install" option
* Due to proxy issues plugin installation will be done later, so select the option "None", so that plugins installation will not be initiated by Jenkins.
* Click "Install" button, which will not initiate plugin installation and will display a form for user details
* Enter employee id in username, fill up other details and click "Save and Continue"
* In the next screen verify the URL and click "Save and Finish" button
* It should display the message "Jenkins is ready"
* Select "Start using Jenkins", which will open the dashboard.
* Sign out once and check if login is working.
* Logout and close the browser window, we  will login after installation of plugin.

**Plugin installation (without internet access)**

* Stop the jenkins server by pressing Ctrl+C in the command prompt window where jenkins server is running
* Get the jenkins-plugins.zip file from the SME. The zip file is available in the same folder as the jenkins.war file
* In Windows File Explorer go to folder D:\Users\<EMP\_ID>\.jenkins and delete the plugins folder
* Copy the plugins folder from jenkins-plugins.zip file and paste the contents in the .jenkins folder
* Start jenkins server by running "java -jar jenkins.war" command in command prompt
* Login into jenkins in browser using your employee id and password
* Subsequent hands on will have steps to configure git and maven to build the project

Reference Code : FSE-ORM-038Complexity : Level1

**Configure Maven and Git path**

* Login into Jenkins
* Go to "Manage Jenkins" > "Global Tool Configuration"
* Git Path
  + Go to "Git" section
  + Click "Add Git"
  + Give "Name" as "Default"
  + Open windows explorer and find out where Git is installed. Check in "Program Files" folder of C drive.
  + "Path to Git executable" should look something like "C:\Program Files\Git\bin\git.exe".
* Maven Path
  + Go to "Maven" section
  + In "Name" provide "Maven" along with version number
  + In "MAVEN\_HOME" provide the root path of maven. (Example: D:\apache-maven-3.5.2)
* Click "Save" to save the configuration

Reference Code : FSE-ORM-039Complexity : Level1

**Build spring-learn project in Jenkins**   
  
**Configure spring-learn project build**

* Login into Jenkins
* Click "New Item"
* Enter item name as "spring-learn-build"
* Click "Maven Project"
* Click "OK"
* In "Source Code Management" section select Git
* In "Repository URL" provide your spring-learn project URL with ".git" suffixed
* Click "Add" > "Jenkins"
* Select "Kind" as "Username with password"
* Provide "Username" as employee id
* Provide "Password" as network password
* Provide "ID" as "gitlab-credentials"
* Provide "Description" as "gitlab-credentials"
* Click "Add"
* In Credentials drop down select "[EMP\_ID]/\*\*\*\*\* (git-credentials)"
* If git configuration give SSL certificate error, execute the below command in GitBash

git config --global http.sslVerify false

* Reopen the spring-learn-build configuration to check if the SSL certificate error is gone
* In Build section in "Goals and options" provide value as "clean package"
* Click "Save"

**Executing the build**

* Now in right hand side of the Jenkins dashboard the "spring-learn-build" project will be listed
* Click on "spring-learn-build"
* Click "Build Now" in the left hand side menu
* In the left hand side menu bottom, there will be a section "Build History"
* In "Build History" section there will be a blinking icon
* Click on the blinking icon to view the build progress log
* Check if the build is successful and the JAR file is generated.
* If there are any test cases that fail, make changes to have successful test execution and make the project build status as green.
* In windows explorer go to C:\Users\[EMP\_ID]\.jenkins\workspace\spring-learn-build to see the code download from git and build output available in target folder.