WEEK – 3 HANDS ON EXERCISES

SPRING CORE MAVEN

1. BASIC SPRING APPLICATION

//MyApp.java

```
package com.library;

import com.library.service.BookService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MyApp {
   private static BookService bookService;

   public static void main(String[] args) {
      ApplicationContext context = new
ClassPathXmlApplicationContext("applcontent.xml");
      bookService = context.getBean("bookService", BookService.class);
      bookService.displayBook();
   }
}
```

//BookService.java

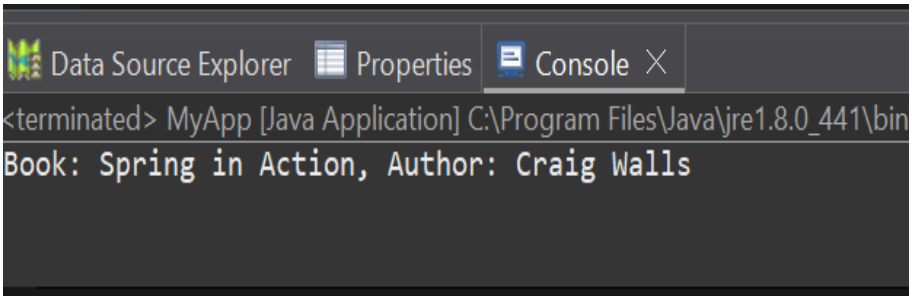
```
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {
```

```
    private BookRepository bookRepository;

    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void displayBook() {
        System.out.println(bookRepository.getBookDetails());
    }
}
```

//BookRepository.java

```
package com.library.repository;

public class BookRepository {
    public String getBookDetails() {
        return "Book: Spring in Action, Author: Craig Walls";
    }
}
```

OUTPUT:



2. IMPLEMENTING DEPENDENCY INJECTION

//MyApp.java

```
package com.library;


import com.library.service.BookService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;


public class MyApp {
    private static BookService bookService;

    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applcontent.xml");
        bookService = context.getBean("bookService", BookService.class);
        bookService.showBookDetails();
    }
}
```

//BookService.java

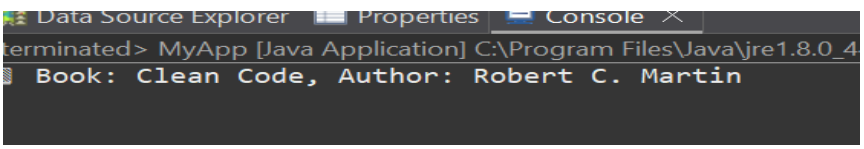
```
package com.library.service;


import com.library.repository.BookRepository;
```

```
public class BookService {
    private BookRepository repository;

    // Setter for Dependency Injection
    public void setRepository(BookRepository repository) {
        this.repository = repository;
    }

    public void showBookDetails() {
        System.out.println(repository.fetchBookInfo());
    }
}
```

//BookRepository.java

```
package com.library.repository;

public class BookRepository {
    public String fetchBookInfo() {
        return "📘 Book: Clean Code, Author: Robert C. Martin";
    }
}
```

OUTPUT:



3. CREATING AND CONFIGURING A MAVEN PROJECT

//MyApp.java

```
package com.library;


import com.library.service.BookService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;


public class MyApp {
   private static BookService bookService;


   public static void main(String[] args) {
      ApplicationContext context = new
ClassPathXmlApplicationContext("applcontent.xml");
      bookService = context.getBean("bookService", BookService.class);
      bookService.showBookDetails();
   }
}
```

//BookService.java

```
package com.library.service;


import com.library.repository.BookRepository;
```

```
public class BookService {
    private BookRepository repository;

    // Setter for Dependency Injection
    public void setRepository(BookRepository repository) {
        this.repository = repository;
    }

    public void showBookDetails() {
        System.out.println(repository.fetchBookInfo());
    }
}
```

//BookRepository.java

```
package com.library.repository;

public class BookRepository {
    public String fetchBookInfo() {
        return "📘 Book: Clean Code, Author: Robert C. Martin";
    }
}
```

//pom.xml

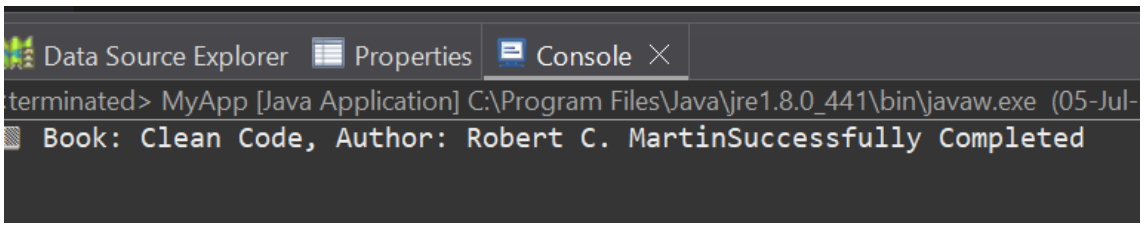
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
```

```
        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.library</groupId>
    <artifactId>library-management</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>LibraryManagement</name>
    <description>A Spring-based Library Management
system</description>

    <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>

    <dependencies>
        <!-- ✅ Spring Context -->
        <dependency>
            <groupId>org.springframework</groupId>
```

```
        <artifactId>spring-context</artifactId>
        <version>5.3.30</version>
    </dependency>

    <!-- ✅ Spring AOP -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aop</artifactId>
        <version>5.3.30</version>
    </dependency>

    <!-- ✅ Spring WebMVC -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.3.30</version>
    </dependency>
</dependencies>

<build>
    <plugins>
        <!-- ✅ Maven Compiler Plugin -->
        <plugin>
```

```
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.1</version>
            <configuration>
              <source>1.8</source>
              <target>1.8</target>
            </configuration>
          </plugin>
        </plugins>
      </build>

    </project>
```

OUTPUT:



SPRING DATA

1. SPRING DATA JPA EXAMPLE

//Country.java

```
package com.cognizant.orm_learn.model;
 import jakarta.persistence.*;
```

```
@Entity
@Table(name = "country")
public class Country {

   @Id
   @Column(name = "co_code")
   private String code;

   @Column(name = "co_name")
   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;
   }

   // toString
   @Override
   public String toString() {
      return "Country [code=" + code + ", name=" + name + "]";
```

```
    }
}
```

//CountryRepsitory.java

```
  package com.cognizant.orm_learn.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.cognizant.orm_learn.model.Country;

@Repository
public interface CountryRepository extends JpaRepository<Country, String> {
}
```

//CountryService.java

```
package com.cognizant.orm_learn.service;

import java.util.List;

import jakarta.transaction.Transactional;



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

import com.cognizant.orm_learn.model.Country;
import com.cognizant.orm_learn.repository.CountryRepository;

@Service
public class CountryService {
```

```
    @Autowired
    private CountryRepository countryRepository;

    @Transactional
    public List<Country> getAllCountries() {
        return countryRepository.findAll();
    }
}
```

//OrmLearnApplication.java

```
package com.cognizant.orm_learn;

import com.cognizant.orm_learn.model.Country;
import com.cognizant.orm_learn.service.CountryService;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;

import java.util.List;

@SpringBootApplication
public class OrmLearnApplication {

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

    public static void main(String[] args) {
```

```
        ApplicationContext context = SpringApplication.run(OrmLearnApplication.class,
args);
        LOGGER.info("Inside main");
        countryService = context.getBean(CountryService.class);
        testGetAllCountries();
    }

    private static void testGetAllCountries() {
        LOGGER.info("Start");
        List<Country> countries = countryService.getAllCountries();
        LOGGER.debug("countries={}", countries);
        LOGGER.info("End");
    }
}
```

OUTPUT:

2. DIFFERENCE BETWEEN JPA, HIBERNATE AND SPRING DATA JPA

//addEmployeeHibernate.java

```
public Integer addEmployeeHibernate(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;
}
```

//ddEmployeeSpringDataJPA.java

```
@Autowired
private EmployeeRepository employeeRepository;

@Transactional
public void addEmployee(Employee employee) {
    employeeRepository.save(employee);
}
```

OUTPUT:

```
Hibernate: insert into employee (name, age, department) values (?, ?, ?)
✔ Employee added using Hibernate. ID: 101
```