

1. Maven version terminal screenshot

```
p.yaswanthreddy — -zsh — 80x24
[(base) p.yaswanthreddy@PYASWANTHS-MacBook-Pro ~ % mvn -v
Apache Maven 3.9.11 (3e54c93a704957b63ee3494413a2b544fd3d825b)
Maven home: /opt/apache-maven-3.9.11
Java version: 24.0.1, vendor: Oracle Corporation, runtime: /Users/p.yaswanthreddy/Library/Java/JavaVirtualMachines/openjdk-24.0.1/Contents/Home
Default locale: en_IN, platform encoding: UTF-8
OS name: "mac os x", version: "15.5", arch: "aarch64", family: "mac"
(base) p.yaswanthreddy@PYASWANTHS-MacBook-Pro ~ %
```

2. Maven Central Repository

- A remote, online repository managed by the Maven community.
- It Contains a vast collection of open-source libraries and dependencies.
- Maven checks here when a dependency is not found locally.

Local Repository

- A directory on our local machine (usually ~/.m2/repository).
- Stores dependencies that are downloaded from central or remote repositories.
- Maven checks the local repository first before going online.
- Improves build speed and reduces internet dependency.

3. Maven commands

- a. To build the maven project - **mvn clean install**
- b. To run the maven tests - **mvn test**

4. Maven settings.xml

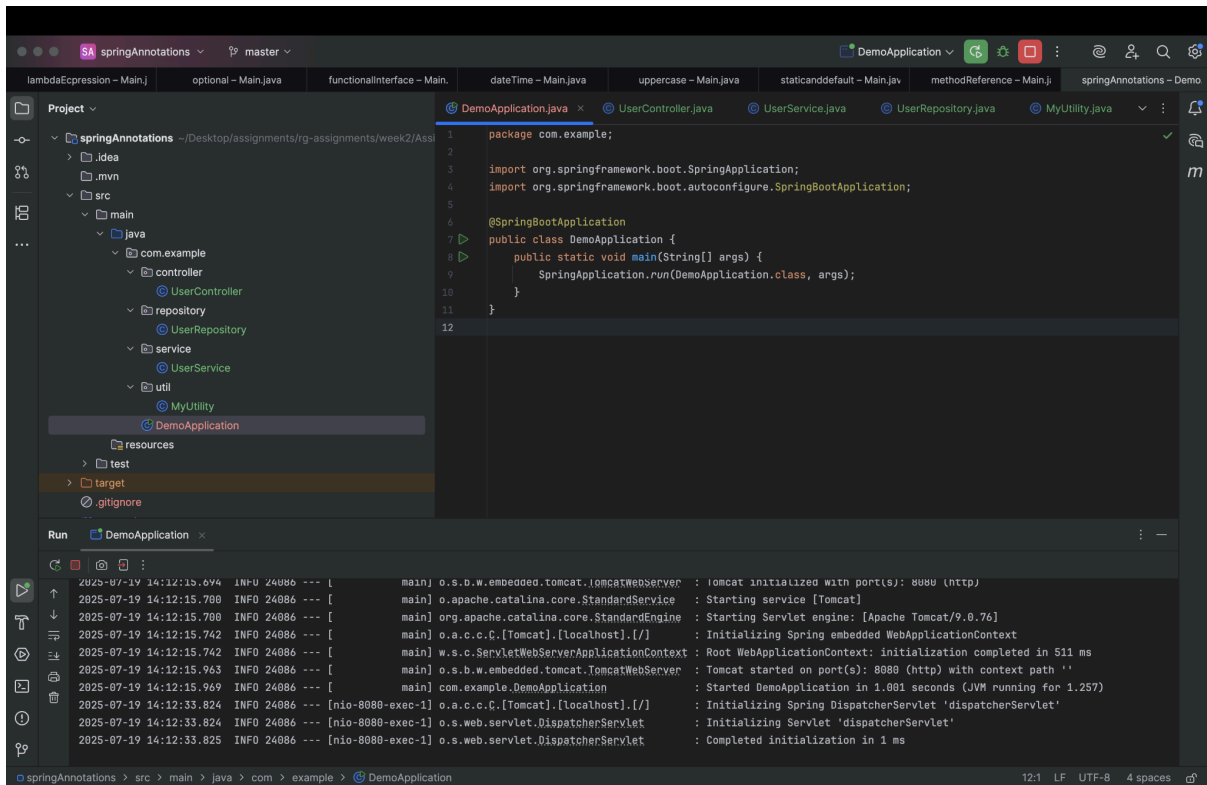
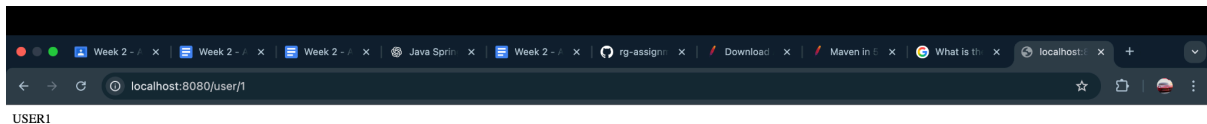
```
conf — -zsh — 80x24
commons-chain          mysql
commons-codec          net
commons-collections    org
commons-digester       oro
commons-io             resolver-status.properties
commons-lang           xml-apis
[(base) p.yaswanthreddy@PYASWANTHS-MacBook-Pro repository % cd~
zsh: command not found: cd~
[(base) p.yaswanthreddy@PYASWANTHS-MacBook-Pro repository % cd ~
[(base) p.yaswanthreddy@PYASWANTHS-MacBook-Pro ~ % cd ..
[(base) p.yaswanthreddy@PYASWANTHS-MacBook-Pro /Users % cd ..
[(base) p.yaswanthreddy@PYASWANTHS-MacBook-Pro / % ls
Applications    etc          private     Users
bin             home        sbin        usr
cores          Library     System      var
dev            opt         tmp         Volumes
[(base) p.yaswanthreddy@PYASWANTHS-MacBook-Pro / % cd opt/apache-maven-3.9.11
[(base) p.yaswanthreddy@PYASWANTHS-MacBook-Pro apache-maven-3.9.11 % ls
bin            conf          LICENSE     README.txt
boot          lib           NOTICE
[(base) p.yaswanthreddy@PYASWANTHS-MacBook-Pro apache-maven-3.9.11 % cd conf
[(base) p.yaswanthreddy@PYASWANTHS-MacBook-Pro conf % ls
logging      settings.xml  toolchains.xml
(base) p.yaswanthreddy@PYASWANTHS-MacBook-Pro conf %
```

Maven local repository

```
repository — -zsh — 80x24
Last login: Sat Jul 19 13:10:51 on ttys001
[(base) p.yaswanthreddy@PYASWANTHS-MacBook-Pro ~ % cd ~/.m2
[(base) p.yaswanthreddy@PYASWANTHS-MacBook-Pro .m2 % ls
repository
[(base) p.yaswanthreddy@PYASWANTHS-MacBook-Pro .m2 % cd repository && ls
aopalliance                                commons-logging
archetype-catalog-central.xml              dom4j
archetype-catalog-central.xml.sha1         io
avalon-framework                          javax
classworlds                               junit
com                                         log4j
commons-beanutils                          logkit
commons-chain                              mysql
commons-codec                              net
commons-collections                        org
commons-digester                           oro
commons-io                                 resolver-status.properties
commons-lang                               xml-apis
(base) p.yaswanthreddy@PYASWANTHS-MacBook-Pro repository %
```

5. Different ways to Define Dependencies in DI:
 - a. Setter Injection
 - b. Constructor Injection
 - c. Field Injection
 - d. Interface Injection
6. `@Autowired` can be applied to fields or constructors to allow the Spring Framework to automatically inject dependencies. Similarly, `@Inject` serves the same purpose. The main difference is that `@Inject` is a standard annotation defined by JSR-330, whereas `@Autowired` is specific to the Spring Framework.
7. In Spring Framework, the annotations `@Component`, `@Repository`, `@Service`, and `@Controller` are stereotype annotations which are used to define Spring-managed beans and their specific roles in an application.
 1. `@Component`
 - Generic stereotype for any Spring-managed component.
 - It is the parent of all stereotype annotations.
 2. `@Repository`
 - Specialization of `@Component`.
 - Used for DAO (Data Access Object) classes.
 - Enables exception translation into Spring's `DataAccessException`
 3. `@Service`
 - Specialization of `@Component`.
 - Used for business logic/service layer classes.
 4. `@Controller`
 - Specialization of `@Component`.
 - Used to define a web controller in Spring MVC.

Output



Github link:

<https://github.com/PathireddyYaswanthReddy/rg-assignments/blob/master/week2/Assignment%20Spring/springAnnotations/src/main/java/com/example/DemoApplication.java>

8. The issue is with the below line

@Value("app.name")

The above statement is just injecting the literal string "app.name" instead of the value of the property from application.properties.

To fix this we need to wrap the property key in `${}` so Spring knows it's a property reference.

Correct Code

```
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

@Component
public class AppNamePrinter {

    @Value("${app.name}") // Correct way to reference a property
    private String appName;
```

```

public void printAppName() {
    System.out.println("Application Name: " + appName);
}
}

```

9. The `@SpringBootApplication` is a convenience annotation in Spring Boot that combines the following three annotations:

- `@Configuration`
- `@EnableAutoConfiguration`
- `@ComponentScan`

It tells Spring Boot to start auto-configuring your app, scan for `@Component`, `@Service`, `@Repository`, and `@Controller` classes and treat the class as a configuration class.

10. Maven command to start spring boot application - `mvn spring-boot:run`

11. Spring Task Output Screenshots

```

// Main.java
public class Main {
    public static void main(String[] args) {
        System.out.println("Employees List");
        dao.getAll().forEach( Employee e ->
            System.out.println(e.getId() + " , " + e.getName() + " , " + e.getDepartment()
        );
        System.out.println("\nUpdating Employee with ID 1:");
        dao.update(new Employee( id: 1, name: "Yaswanth", department: "Finance"));
        System.out.println("Employee Updated.");
        System.out.println("\nDeleting Employee with ID 2:");
        dao.delete( id: 2);
        System.out.println("Employee Deleted.");
        System.out.println("\nFinal Employees List:");
    }
}

```

```

// Run Output
/Users/p.yaswanthreddy/Library/Java/JavaVirtualMachines/azul-1.8.0_452/Contents/Home/bin/java ...
Employee Creation
Employees Created successfully
Employees List
1, Yaswanth, IT
2, Arun, HR

Updating Employee with ID 1:
Employee Updated.

Deleting Employee with ID 2:
Employee Deleted.

Final Employees List:
1, Yaswanth, Finance

Process finished with exit code 0

```

```

[mysql> select * from Employee;
+-----+-----+-----+
| id | name      | department |
+-----+-----+-----+
| 1 | Yaswanth | Finance    |
+-----+-----+-----+
1 row in set (0.003 sec)

mysql>

```

Github link:

<https://github.com/PathireddyYaswanthReddy/rg-assignments/blob/master/week2/Assignment%20Spring/Employee-Spring/src/main/java/com/example/Main.java>

12. Spring Boot and Spring Data JPA output screenshots

The first screenshot shows a web browser at `localhost:8080/employees` displaying a JSON array of two employees: `[{"id": 1, "name": "Yaswanth", "department": "Finance"}, {"id": 2, "name": "Ravi", "department": "HR"}]`.

The second screenshot shows a web browser at `localhost:8080/employees/1` displaying a JSON object for the first employee: `{"id": 1, "name": "Yaswanth", "department": "Finance"}`.

The third screenshot shows a MySQL terminal output for the query `select * from Employee;`:

id	name	department
1	Yaswanth	Finance
2	Ravi	HR

2 rows in set (0.001 sec)

Github link:

<https://github.com/PathireddyYaswanthReddy/rq-assignments/blob/master/week2/Assignment%20Spring/Employee-SpringBoot-JPA/src/main/java/com/example/Main.java>

13. Spring Batch output screenshots

The screenshot shows an IDE with the following components:

- Project View:** Shows the project structure for `Customer-SpringBatch`, including `src/main/java/com/example/config` with `CustomerProcessor` and `SpringBatchConfig`.
- Code Editor:** Displays the `Main.java` file with the following code:

```
1 @SpringBootApplication
2 public class Main {
3
4     public static void main(String[] args) {
5         SpringApplication.run(Main.class, args);
6     }
7 }
```
- Terminal:** Shows the output of the application, including Hibernate SQL queries and the completion of the `ImportCustomers` job. The output indicates that the job completed successfully in 276ms.

```
(base) p.yaswanthreddy@PYASWANTHS-MacBook-Pro ~ % curl -X POST http://localhost:9191/jobs/importCustomers
(base) p.yaswanthreddy@PYASWANTHS-MacBook-Pro ~ %
```

```
mysql> show tables;
+-----+
| Tables_in_testdb |
+-----+
| Employee          |
+-----+
1 row in set (0.004 sec)

mysql> show tables;
+-----+
| Tables_in_testdb |
+-----+
| BATCH_JOB_EXECUTION |
| BATCH_JOB_EXECUTION_CONTEXT |
| BATCH_JOB_EXECUTION_PARAMS |
| BATCH_JOB_EXECUTION_SEQ |
| BATCH_JOB_INSTANCE |
| BATCH_JOB_SEQ |
| BATCH_STEP_EXECUTION |
| BATCH_STEP_EXECUTION_CONTEXT |
| BATCH_STEP_EXECUTION_SEQ |
| customers_info |
| Employee |
+-----+
11 rows in set (0.006 sec)

mysql> select * from customers_info limit 5;
+-----+
| customer_id | contact | country | dob | email | first_name | gender | last_name |
+-----+
| 3 | 997-395-9270 | France | 12-05-1994 | sgaw2@mit.edu | Sallie | Female | Gaw |
| 6 | 615-444-4318 | Brazil | 24-05-1990 | wdouberday5@mozilla.org | West | Male | Douberday |
| 10 | 654-596-4163 | Netherlands | 04-08-1999 | borridge9@exblog.jp | Babs | Female | Orridge |
| 13 | 627-566-6711 | China | 16-11-2012 | thoodlassc@ucsd.edu | Thelma | Female | Hoodlass |
| 18 | 641-445-0261 | Brazil | 22-02-2015 | nbaggallyh@discuz.net | Noelani | Female | Baggally |
+-----+
5 rows in set (0.004 sec)

mysql> select count(*) from customers_info;
+-----+
| count(*) |
+-----+
| 1000 |
+-----+
1 row in set (0.001 sec)

mysql>
```

Github link:

<https://github.com/PathireddyYaswanthReddy/rg-assignments/blob/master/week2/Assignment%20Spring/Customer-SpringBatch/src/main/java/com/example/Main.java>