

Component	Purpose
Employee.java	Model/POJO class for Employee entity
EmployeeService.java	Business logic (CRUD operations) using Spring's <code>@Service</code>
MainApp.java	Configuration + Entry point; uses <code>@Configuration</code> and <code>@ComponentScan</code>
pom.xml	Maven configuration file — manages dependencies for Spring Boot

Employee.java (Model Class)

=====

```
package com.example;

import org.springframework.stereotype.Component;

@Component
public class Employee {
    private int id;
    private String name;

    @Override
    public String toString() {
        return "Employee [id=" + id + ", name=" + name + "]";
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

Explanation:

- `@Component` tells Spring that this is a *Spring Bean*.
- Spring automatically creates an object of `Employee` class and manages its lifecycle.
- This is called **Bean creation via Component Scanning**.
- Fields `id` and `name` represent employee details.

```
EmployeeService.java (Service Layer)
=====
package com.example;

import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
@Service
public class EmployeeService {
    @Autowired
    private Employee employee;
    private Map<Integer, Employee> employeeDatabase = new HashMap<>();
    private Scanner scanner = new Scanner(System.in);

    public Employee getEmployee() {
        return employee;
    }

    public void setEmployee(Employee employee) {
        this.employee = employee;
    }

    public Map<Integer, Employee> getEmployeeDatabase() {
        return employeeDatabase;
    }

    public void setEmployeeDatabase(Map<Integer, Employee> employeeDatabase) {
        this.employeeDatabase = employeeDatabase;
    }

    public void addEmployee(Employee employee) {
        employeeDatabase.put(employee.getId(), employee);
    }

    public void updateEmployee(Employee updatedEmployee) {
        employeeDatabase.put(updatedEmployee.getId(), updatedEmployee);
    }

    public void deleteEmployee(int employeeId) {
        employeeDatabase.remove(employeeId);
    }

    public Employee getEmployee(int employeeId) {
        return employeeDatabase.get(employeeId);
    }

    public void displayAllEmployees() {
        System.out.println("Employee List:");
        for (Employee emp : employeeDatabase.values()) {
            System.out.println(emp);
        }
    }
}
```

```
}

public void runMenu() {
    boolean exit = false;
    while (!exit) {
        System.out.println("\nMenu:");
        System.out.println("1. Add Employee");
        System.out.println("2. Update Employee");
        System.out.println("3. Delete Employee");
        System.out.println("4. Get Employee by ID");
        System.out.println("5. Display All Employees");
        System.out.println("6. Exit");
        System.out.print("Enter your choice: ");

        int choice = scanner.nextInt();

        switch (choice) {
            case 1:
                addEmployeeFromInput();
                break;
            case 2:
                updateEmployeeFromInput();
                break;
            case 3:
                deleteEmployeeFromInput();
                break;
            case 4:
                getEmployeeById();
                break;
            case 5:
                displayAllEmployees();
                break;
            case 6:
                exit = true;
                break;
            default:
                System.out.println("Invalid choice. Please try again.");
                break;
        }
    }
}

private void addEmployeeFromInput() {
    System.out.print("Enter employee ID: ");
    int id = scanner.nextInt();
    System.out.print("Enter employee name: ");
    String name = scanner.next();
    Employee newEmployee = new Employee();
    newEmployee.setId(id);
    newEmployee.setName(name);
    addEmployee(newEmployee);
    System.out.println("Employee added successfully.");
}
```

```

private void updateEmployeeFromInput() {
    System.out.print("Enter employee ID to update: ");
    int id = scanner.nextInt();
    Employee existingEmployee = getEmployee(id);
    if (existingEmployee != null) {
        System.out.print("Enter updated employee name: ");
        String name = scanner.next();
        existingEmployee.setName(name);
        updateEmployee(existingEmployee);
        System.out.println("Employee updated successfully.");
    } else {
        System.out.println("Employee not found with ID " + id);
    }
}

private void deleteEmployeeFromInput() {
    System.out.print("Enter employee ID to delete: ");
    int id = scanner.nextInt();
    Employee existingEmployee = getEmployee(id);
    if (existingEmployee != null) {
        deleteEmployee(id);
        System.out.println("Employee deleted successfully.");
    } else {
        System.out.println("Employee not found with ID " + id);
    }
}

private void getEmployeeById() {
    System.out.print("Enter employee ID to retrieve: ");
    int id = scanner.nextInt();
    Employee existingEmployee = getEmployee(id);
    if (existingEmployee != null) {
        System.out.println("Employee details:");
        System.out.println(existingEmployee);
    } else {
        System.out.println("Employee not found with ID " + id);
    }
}
}

```

Key Concepts:

Annotation	Meaning
@Service	Marks this class as a <i>service bean</i> (business logic layer).
@Autowired	Automatically injects the Employee bean created by Spring.
Map<Integer, Employee>	Acts as an in-memory database.

Method	Purpose
addEmployee (Employee e)	Adds employee to HashMap
updateEmployee (Employee e)	Updates existing employee
deleteEmployee (int id)	Deletes employee
getEmployee (int id)	Fetches employee details
displayAllEmployees ()	Prints all employees
runMenu ()	Provides a console-based menu for user interaction

MainApp.java (Configuration + Runner)

```
=====
package com.example;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.support.ClassPathXmlApplicationContext;

@Configuration
@ComponentScan("com.example")
public class MainApp {
    public static void main(String[] args) {
        // Load the Spring context
        AnnotationConfigApplicationContext context = new
        AnnotationConfigApplicationContext(MainApp.class);

        // Retrieve the bean from the Spring context
        EmployeeService employeeService = (EmployeeService) context.getBean("employeeService");

        employeeService.runMenu();
    }
}
```

Annotation	Purpose
@Configuration	Marks this class as a Spring configuration class (instead of XML).
@ComponentScan ("com.example")	Tells Spring to scan the package and detect all @Component, @Service, etc. classes automatically.
AnnotationConfigApplicationContext	Loads context based on annotations (no XML needed).

Spring automatically wires everything:

- Creates a `EmployeeService` bean.
- Injects a `Employee` bean inside it.

```
pom.xml
=====
<?xml version="1.0" encoding="UTF-8"?>
<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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.1.4</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.example</groupId>
  <artifactId>SpringAnnotationDemo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>
  <name>SpringAnnotationDemo</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>17</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-tomcat</artifactId>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>
```