1. Overview

The Employee Management System is a simple Spring-based application that facilitates the management of employee records. It utilizes the Spring framework's features such as Dependency Injection, JDBC Template, and Bean Scopes.

2. Code Structure

2.1. Package Structure

- **com.bean:** Contains the Employee class representing the model for an employee.
- **com.dao:** Includes the EmployeeDao class responsible for database operations using Spring's JDBC Template.
- **com.main:** Houses the main class (App) for user interaction and application execution.
- **com.service:** Defines the EmployeeService class, handling business logic for employee operations.

2.2. Class Details

2.2.1. Employee (com.bean.Employee)

- **Attributes:**
  - `id`: Employee ID
  - `name`: Employee Name
  - `salary`: Employee Salary
- **Methods:**
  - Constructors for creating Employee instances.
  - Getter and setter methods for attributes.
  - `toString()`: Overrides the default `toString` method to provide a formatted string representation of the Employee.

2.2.2. EmployeeDao (com.dao.EmployeeDao)

- **Attributes:**
  - `jdbcTemplate`: Autowired instance of Spring's JdbcTemplate for database operations.
- **Methods:**
  - `storeEmployee(Employee employee)`: Stores an employee record in the database.
  - `updateEmployee(Employee employee)`: Updates the salary of an existing employee.
  - `deleteEmployee(int id)`: Deletes an employee record by ID.
  - `findEmployee()`: Retrieves all employee records in map format.
  - `findEmployeeByRowMapper()`: Retrieves all employee records in list format using a custom RowMapper.

2.2.3. App (com.main.App)

- **Methods:**
  - `main(String[] args)`: Entry point of the application.
  - Accepts user input to perform CRUD operations on employee records.

2.2.4. EmployeeService (com.service.EmployeeService)

- **Attributes:**
  - **`employeeDao`**: Autowired instance of the EmployeeDao for database operations.
- **Methods:**
  - **`storeEmployee(Employee employee)`**: Stores an employee record.
  - **`updateEmployee(Employee employee)`**: Updates an employee record.
  - **`deleteEmployee(int id)`**: Deletes an employee record by ID.
  - **`findEmployeeInMapFormat()`**: Retrieves all employee records in map format.
  - **`findEmployeeInListFormat()`**: Retrieves all employee records in list format.

3. Bean Configuration

The application likely includes a Spring XML configuration file (**`beans.xml`**) defining the beans and their dependencies. However, the specific content of this file is not provided in the shared code.

4. Usage

- Run the main class **`App`** to interact with the application.
- Follow the on-screen prompts to add, update, delete, or retrieve employee records.

5. Dependencies

- Spring Framework: Utilized for Dependency Injection, JDBC Template, and other features.

6. Notes

- The application uses a prototype scope for the **`Employee`** bean, indicating that a new instance is created for each request.