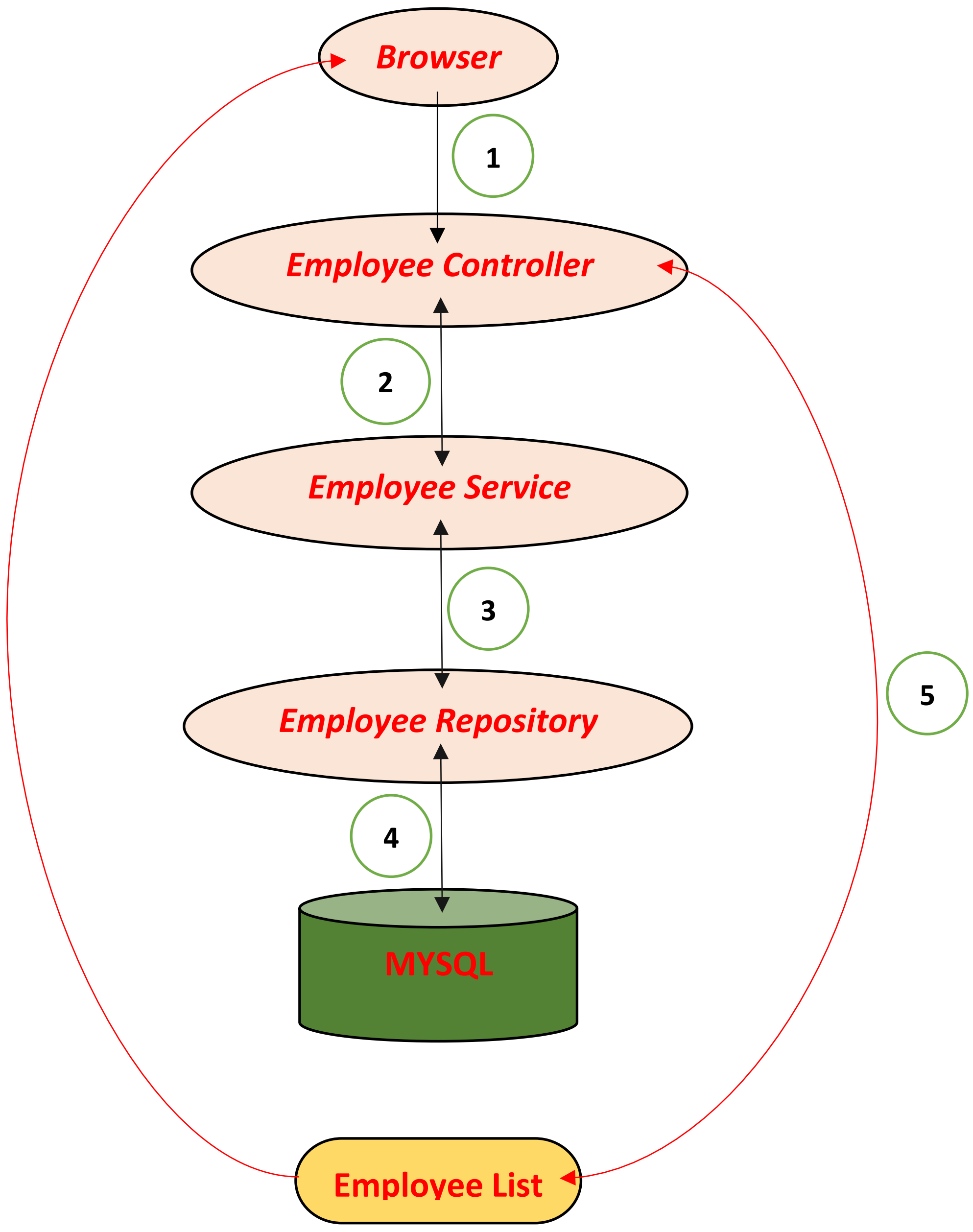
Spring Boot CRUD Application Project

Project Requirements: -

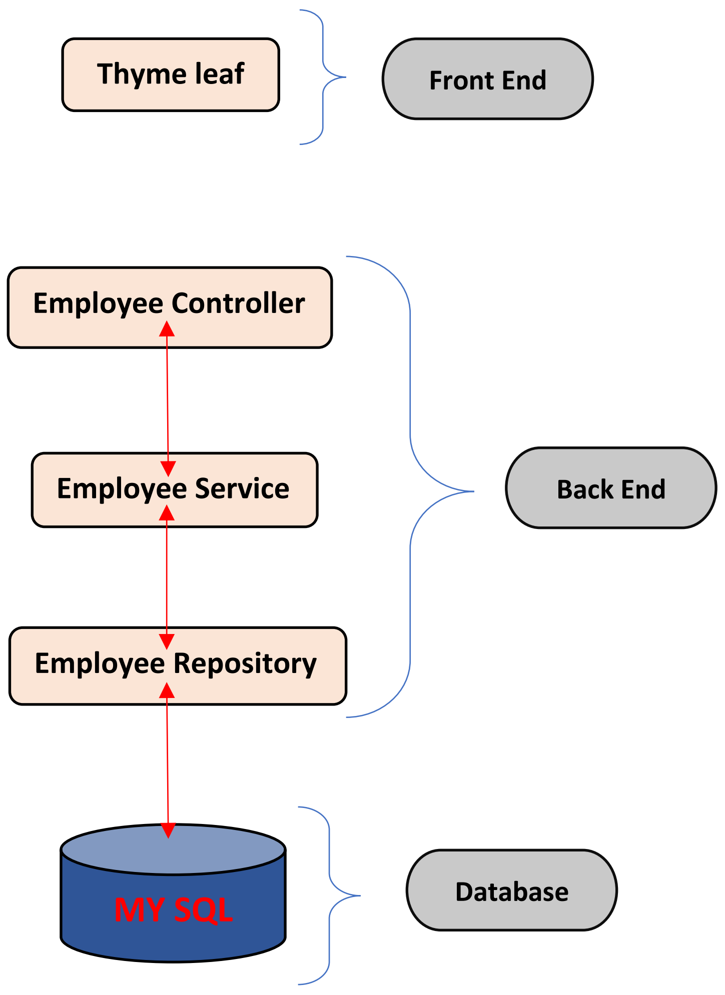
Create a web Application for Employee Management System using All the features of CRUD Application.

- Get all the Employees.
- Add a new Employee.
- Update an employee data.
- Delete an Employee.



1st Step: -

Application Architecture



❖ What is the CRUD operation?

- The CRUD operation can be defined as user interface conventions that allow view, search, and modify information through computer-based forms and reports.
- CRUD is data-oriented and the standardized use of HTTP action verbs. HTTP has a few important verbs.

❖ How CRUD Operations Works.

- CRUD operations are at the foundation of the most dynamic websites. Therefore, we should differentiate CRUD from the HTTP action verbs.
- Suppose, if we want to create a new record, we should use HTTP action verb POST. To update a record, we should use the PUT verb. Similarly, if we want to delete a record, we should use the DELETE verb. Through CRUD operations, users and administrators have the right to retrieve, create, edit, and delete records online.

Spring Boot Crud Repository

Spring Boot provides an interface called Crud Repository that contains methods for CRUD operations. It is defined in the package org.springboot.data.repository. It extends the Spring Data Repository interface. It provides generic Crud operation on a repository. If we want to use Crud Repository in an application, we have to create an interface and extend the Crud Repository.

Syntax

```
public interface CrudRepository<T,ID> extends Repository<T,ID>
```

where,

-T is the domain type that repository manages.

-ID is the type of the id of the entity that repository manages.

For example: -

```
public interface EmployeeRepository extends CrudRepository<Employee, Integer>
{
}
```

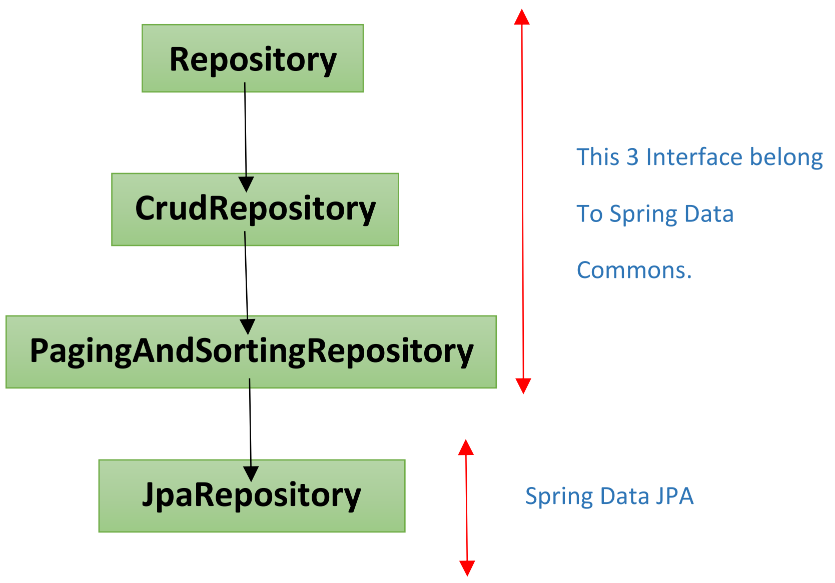
Spring Boot JpaRepository

JpaRepository provides JPA related methods such as flushing, persistence context, and deletes a record in a batch. It is defined in the package org.springframework.data.jpa.repository.JpaRepository extends both CrudRepository and PagingAndSortingRepository.

For example: -

```
public interface Employee extends JpaRepository {
}
```

Spring Data Repository Interface



Spring Boot CRUD Operation Example

Let's set up a Spring Boot application and perform CRUD operation.

Step 1: Open Spring Initializer http://start.spring.io.

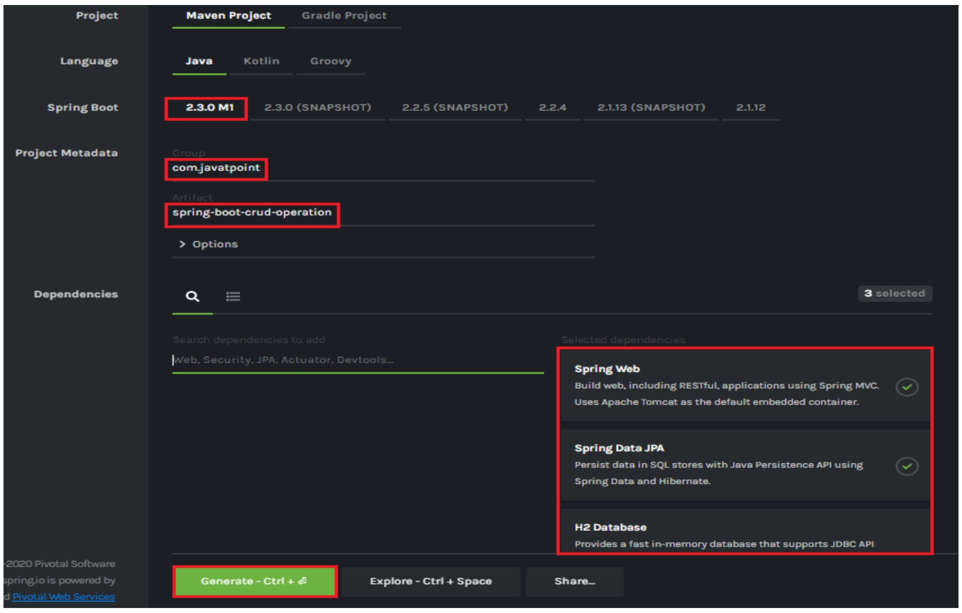
Step 2: Select the Spring Boot version 2.3.0.M1.

Step 3: Provide the Group name. We have provided com.springboot.crud.

Step 4: Provide the Artifact Id. We have provided spring-boot-crud-operation.

Step 5: Add the dependencies Spring Web, Spring Data JPA, and H2 Database.

Step 6: Click on the Generate button. When we click on the Generate button, it wraps the specifications in a Jar file and downloads it to the local system.



Group: com.javatpoint

Artifact: spring-boot-crud-operation

Spring Web
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

H2 Database
Provides a fast in-memory database that supports JDBC API

Step 7: Extract the Jar file and paste it into the STS workspace.

Step 8: Import the project folder into STS.

File -> Import -> Existing Maven Projects -> Browse -> Select the folder spring-boot-crud-operation -> Finish

It takes some time to import.

Step 9: Create a package with the name com. springboot.crudapp.model in the folder src/main/java.

Step 10: Create a model class in the package com. springboot.crudapp.model . We have created a model class with the name Employee.java . In the Employee class, we have done the following:

Step 11: Create a package with the name com. springboot.crudapp.controller in the folder src/main/java.

Step 12: Create a Controller class in the package com. springboot.crudapp.controller . We have created a controller class with the name EmployeeController.java . In the EmployeeController class, we have done the following: Mark the class as RestController by using the annotation @RestController. Autowire the EmployeeService class by using the annotation @Autowired.

Define the following methods:

getAllBooks(): It returns a List of all Employee Details.

getBooks(): It returns an employee detail that we have specified in the path variable. We have passed Emp ID as an argument by using the annotation @PathVariable. The annotation indicates that a method parameter should be bound to a URI template variable.

deleteBook(): It deletes a specific Employee that we have specified in the path variable.

saveBook(): It saves the Employee detail. The annotation @RequestBody indicates that a method parameter should be bound to the body of the web request.

update(): The method updates a record. We must specify the record in the body, which we want to update. To achieve the same, we have used the annotation @RequestBody.

Step 13: Create a package with the name com. springboot.crudapp.service in the folder src/main/java.

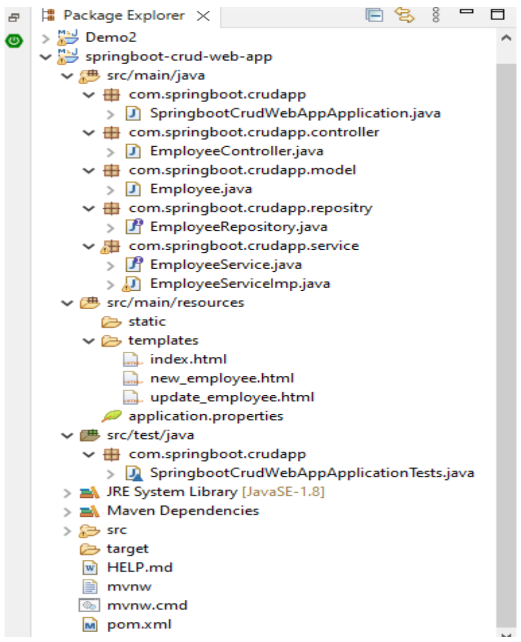
Step 14: Create a Service class. We have created a service class with the name BooksService in the package com. springboot.crudapp.service.

Step 15: Create a package with the name com.springboot.crudapp.repository in the folder src/main/java.

Step 16: Create a Repository interface. We have created a repository interface with the name Employee Repository in the package com.springboot.crudapp.repository. It extends the Crud Repository interface.

application.properties

```
# DATASOURCE (DataSourceAutoConfiguration & DataSourceProperties)

spring.datasource.url=jdbc:postgresql://localhost:5432/Employee?useSSL=false&serverTimezone=UTC&useLegacyDatetimeCode=false

spring.datasource.username=postgres

spring.datasource.password=postgres

# Hibernate

# The SQL dialect makes Hibernate generate better SQL for the chosen database

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQL82Dialect

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

spring.jpa.hibernate.ddl-auto = update

logging.level.org.hibernate.SQL=DEBUG

logging.level.org.hibernate.type=TRACE
```



```
Package Explorer
> Demo2
v springboot-crud-web-app
  v src/main/java
    v com.springboot.crudapp
      > SpringbootCrudWebAppApplication.java
    v com.springboot.crudapp.controller
      > EmployeeController.java
    v com.springboot.crudapp.model
      > Employee.java
    v com.springboot.crudapp.repositry
      > EmployeeRepository.java
    v com.springboot.crudapp.service
      > EmployeeService.java
      > EmployeeServiceImp.java
  v src/main/resources
      static
    v templates
        index.html
        new_employee.html
        update_employee.html
      application.properties
  v src/test/java
    v com.springboot.crudapp
      > SpringbootCrudWebAppApplicationTests.java
  > JRE System Library [JavaSE-1.8]
  > Maven Dependencies
  > src
    target
    HELP.md
    mvnw
    mvnw.cmd
    pom.xml
```

End-------------O-------------End