**What is Spring Boot?**

**Spring Framework?**

1. Spring is a very popular Java EE framework for

building web applications and enterprise applications

2. Spring Framework made Java EE much easier for

building enterprise applications

The main goal of spring boot is to quickly create production ready spring-based application without requiring developers to write the same boilerplate configuration again and again.

Spring Boot Provides

Spring Boot

1. Spring Boot starters

2. Spring Boot Autoconfiguration

3. Externalized Configuration

4. Spring Boot actuator

5. Easy-to-use embedded servlet

Container support

Spring Framework

**In Spring based Framework**

(Developer does lots of configuration)

1. Spring-based applications have a lot of configuration.

2. When we use Spring MVC, we need to configure

- component scan,

- dispatcher Servlet,

- view resolver,

- web jar (for delivering content) among other things

3. When we use Hibernate&JPA, we would need to configure a

- data source

- entity manager factory/session factory

- transaction manager among a host of other things.

4. When you use a cache

- cache configuration

5. When you use Message Queue

- message queue configuration

6. When you use a NoSQL database

- NoSQL database configuration

**1. Spring Boot Starters**

Spring Boot provides a few starters that allow us to add jars in the classpath.

Spring Boot built-in starters make development easier and more rapid.

Spring Boot Starters are the dependency descriptors.

Spring-boot-starter (dependency)

**It is used for core starters, including auto-configuration support, logging, and YAML.**

**2. Spring Boot Auto-configuration**

Spring Boot auto-configuration attempts to automatically configure your

spring application based on the jar dependencies that you have added.

**Why do we need spring Boot Auto Configuration?**

-- Spring-based applications have a lot of configurations.

-- When we use Spring MVC, we need to configure

- component scan,

- dispatcher Servlet,

- view resolver,

- web jar (for delivering content)among other things

-- When we use Hibernate/JPA, we would need to configure a

- data source

- entity manager factory/session factory

- transaction manager among a host of other things.

**In Spring Boot:**

Spring Boot provides an easy way to configure, we just add dependencies, spring

the boot will automatically configure

**Where is Spring Boot Auto-configuration implemented?**

**-** All auto-configuration logic is implemented in spring-boot-autoconfigure.jar

[- @Conditional](mailto:-@Conditional)OnClass: The configuration is enabled only when these classes are available in the classpath.

- @ConditionalOnMissingBean: This bean is configured only if there is no other bean configuration with the same

**3. Externalized Configuration**

Spring Boot lets you externalize your configuration so that you can work with the

same application code in different environments.

you can use properties files, YAML files,and environment variables to externalize configuration.

**4. Spring Boot Actuator**

spring-boot-starter-actuator(dependency)

It is used for Spring Boot's Actuator which provides production-ready features to help you monitor and manage your application.

**For Example:**

- View the application bean configuration details

- View the application URL mappings

- View environment details

- View Configuration parameter values

- view the registered health check metrics

**5. Easy-to-use embedded servlet Container support**

The rising popularity of cloud-native applications and micro-services generates increased demand for embedded servlet containers.

Spring Boot allows developers to easily build applications or services using the 3 most mature containers available: Tomcat, Undertow, and Jetty.

**Different Ways to Create Spring Boot Project**

1. Using spring initializr

2. Using Spring Starter Project in STS (Eclipse)

3. Spring Boot CLI

The Spring Boot CLI is a command line tool that you can if you want to quickly develop a spring application.

**Spring Boot Application Execution Process:**

**1. Spring boot application execution will start from the main() method**

**2. The main() method internally call springApllication.run() method**

**3. SpringApplication.run() method performs bootstraping for our spring boot application**

**4. Starts StopWatch(Class) to identify time taken to bootstrap the spring boot application**

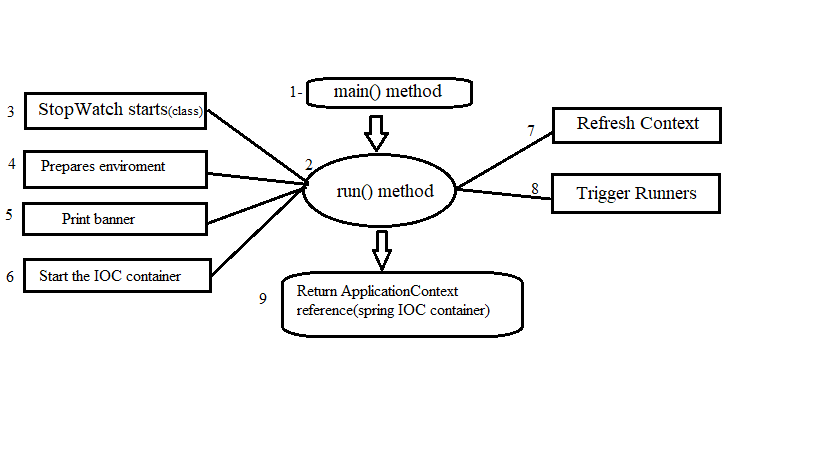
**5. Prepares environment to run our spring boot application (dev, prod, qa, uat)**

**6. Print banner (spring Boot Logo prints on console)**

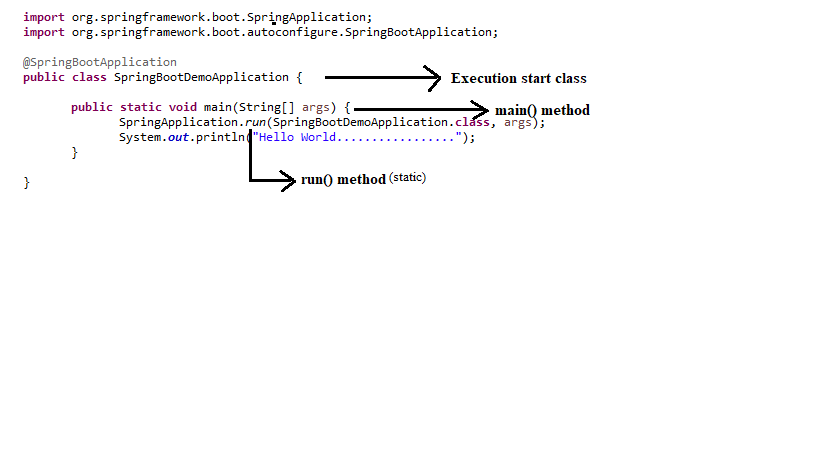
**7. Start the IOC container (ApplicationContext) based on the classpath(deafult, web servlet/Reactive)**

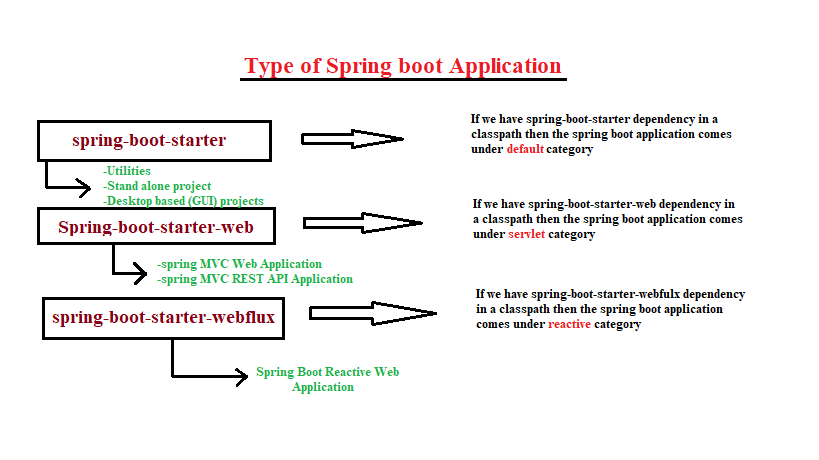
**8. Refresh context**

**9. Tigger Runners (Application or CommandLineRunner)**

**10. Return Application Context reference(Spring IOC)**

**run()method:**



**Types of Spring Boot Application:**

**Spring Boot Starter Parent Dependency**

spring-boot-starter-parent: This dependency provides default configurations for our application

This dependency provides default version 1.8 Java And then they have lots of Maven plug-ins also they have maintained dependency versions.

**Some Third-party dependency**

**1. spring-boot-starter-thymeleaf**

It is used to build MVC web applications using Thymeleaf views.

**2. spring-boot-starter-web-services**

It is used for Spring Web Services.

**3. spring-boot-starter-web**

It is used for building web applications, including RESTful applications using spring MVC.

It uses Tomcat as the default embedded container.

**4. spring-boot-starter-test**

It is used to test Spring Boot applications with libraries, including JUnit, Hamcrest, and mockito.

**5. spring-boot-starter-jdbc**

It is used for JDBC with the Tomcat JDBC connection pool.

**6. spring-boot-starter-validation**

It is used for Java Bean Validation with Hibernate Validator.

**7. spring-boot-starter-jersey**

It is used to build RESTful web applications using JAX-RS and Jersey.

An alternative to spring-boot-starter-web.

**8. spring-boot-starter-security**

It is used for Spring Security.

**9.spring-boot-starter-data-jpa**

It is used for Spring Data JPA with Hibernate.

**10. spring-boot-starter-freemarker**

It is used for building MVC web applications using FreeMarker views.

**11. spring-boot-starter-cache**

It is used for Spring Framework's caching support.

**12. spring-boot-starter-data-rest**

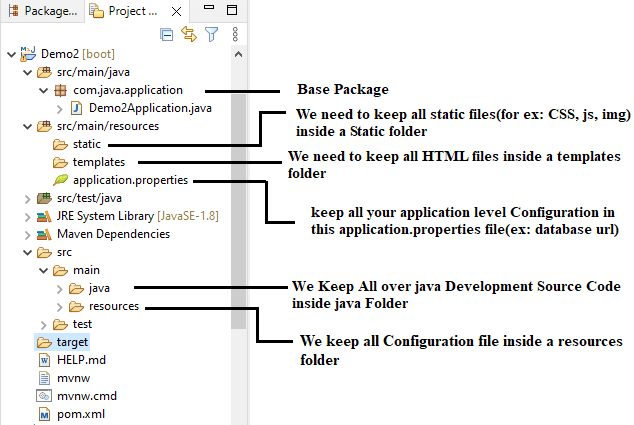
It is used for exposing Spring Data repositories over

REST using Spring Data REST.

**Create a Spring Starter Project:**

(click) New --- Spring Starter Project --- we need to enter Project details and (click) Next --- Spring Boot Version (as default) --- Available (in search box we to enter the dependency name) --- and then (click) Finish.

**This is the Universal standard maven project structure:**

**@SpringBootApplication:**

@SpringBootAppkication Annotation combination of these three annotations.

@SpringBootApplication = @SpringBootConfiguration + @EnableAutoConfiguration + @ComponentScan

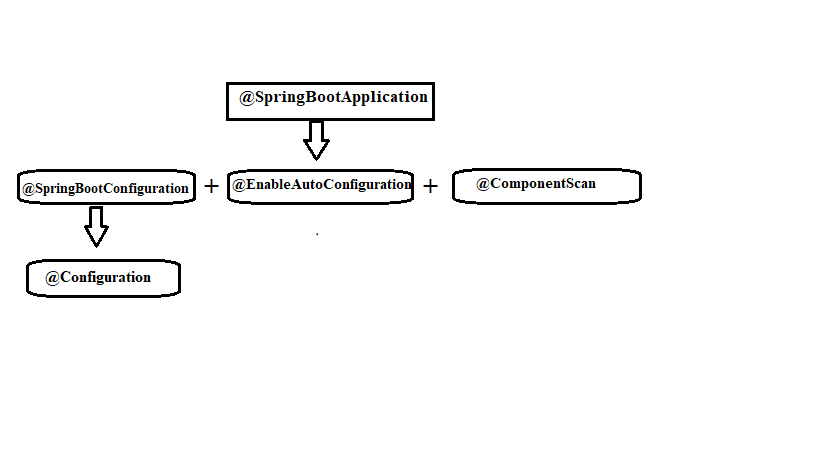
**1. @SpringBootConfiguration**

This annotation internally annotated with the **@Configuration** annotation

**@Configuration** annotation which indicates that the class has @Bean definition methods. So, Spring container can process the class and generate Spring Beans to be used in the application.

**2. @EnableAutoConfiguration** annotation auto-configures the beans that are present in the classpath. This simplifies the developers work by guessing the required beans from the classpath and configure it to run the application.

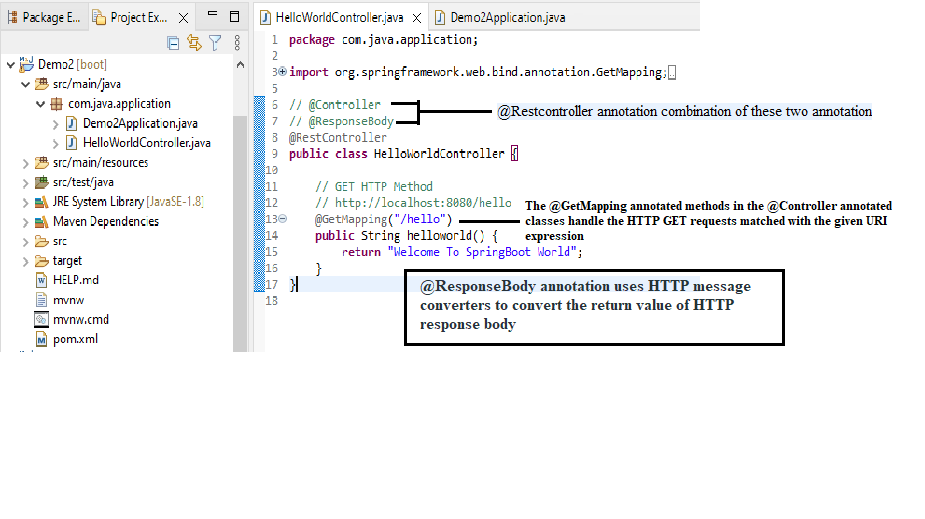
**3. @ComponentScan:** It is used when we want to scan a package for beans. It is used with the annotation @Configuration. We can also specify the base packages to scan for Spring Components

****

**Types of @Configuration in Spring Boot Applications:**

1. XML Configuration
2. Java-Based Configuration
3. Annotations-Based Configuration

**Create Simple Hello World REST API:**

****

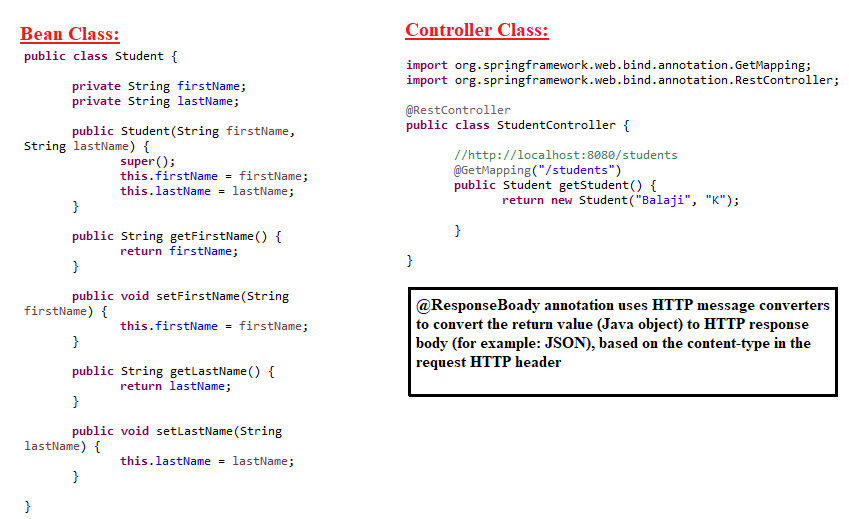
**And then** run the spring boot application. Check the Rest API in the browser (or) postman.

**This URL: http://localhost:8080/hello**

* **@Controller:**This is used to specify the controller.
* **@RequestMapping:** This is used to map to the Spring MVC controller method.
* **@ResponseBody:** Used to bind the HTTP response body with a domain object in the turn type

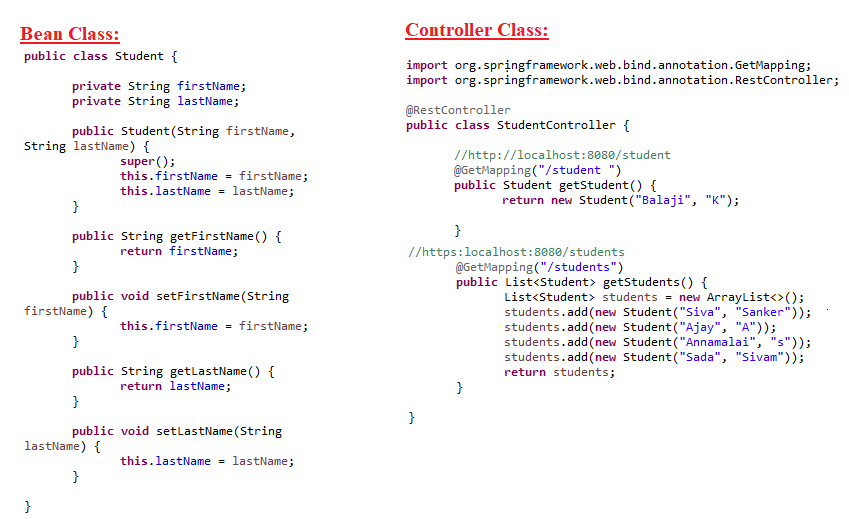
.

**Create REST API which Returns Java Bean as JSON:**



After run the springs boot application, and then enter the URL (http://localhost:8080/students) in the browser or postman.

**Create REST API returns List as JSON Array:**



After run the spring boot application and then use the URL (https:localhost:8080/students) and see the result in the browser or postman.

**Spring Boot REST API with Path Variable - @PathVariable:**



**@pathVariable annotation we bind the request URL template path variable to the method variable.**

**Spring Boot REST API with Request Param - @RequestParam:**

****

1. The **@RequestParam** is used to extract query parameters while **@PathVariable** is used to extract data right from the URI.

2. Even though both are used to extract data from URL, **@RequestParam** is used to retrieve query parameters, anything after? In the URL, while **@PathVariable** is used to retrieve values from URI itself.

**Embedded Servers - Tomcat, Jetty and Undertow:**

1. Default embedded server with spring boot – **Tomcat (**default server**)**

2. Embedded Servers Spring boot supports:

* + Tomcat
  + Jetty
  + Undertow

3. Spring-boot-starter-web (dependency): This dependency intently provides **Tomcat** embedded server as default.

**Running Spring Boot Apps from the Command Line:**

Two options for running spring boot app:

1. Using java –jar command.
2. Using Spring boot maven plugin – mvn spring-boot:run.

**Spring Boot Dev Tools (dependency):**

Without this dependency **Problem:**

In spring boot application development:

* If you make changes in spring boot app
* Then you have to manually restart your application

**Solution:**

Add **spring-boot-devtools** dependency in the class path.

**Spring Boot DevTools provides the following features:**

* **Property Defaults**
* **Automatic Restart**
* **LiveReload**
* **Remote Debug Tunneling**
* **Remote Update and Restart**

**What is a Lombok?**

Project Lombok (from now on, Lombok) is **an annotation-based Java library that allows you to reduce boilerplate code**. Lombok offers various annotations aimed at replacing Java code that is well known for being boilerplate, repetitive, or tedious to write.

**@Data** annotation is important

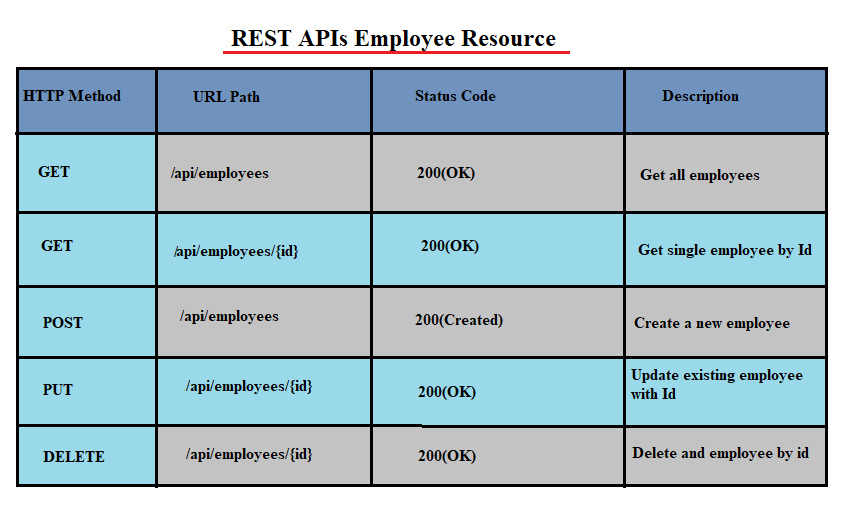
**Create Spring Boot CRUD REST APIs:**

**API Requirement from Client:**

**Create REST APIs for Employee Management System**

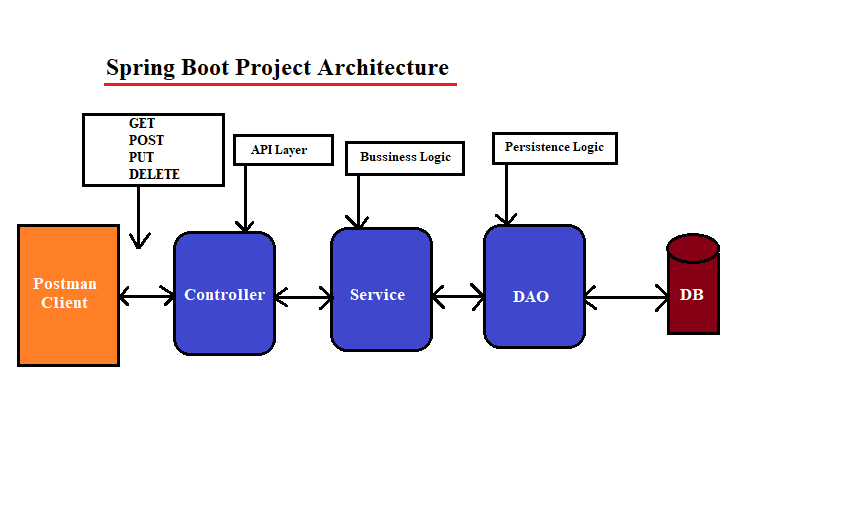
**Rest Clients should be able to:**

* Get a list of employees
* Get a single employee by id
* Create a new employee
* Update an existing employee
* Delete an employee

****

**Tools and Technologies used:**

* Spring Data JPA (Hibernate)
* Lombok Library
* JDK 16 (your wish)
* Embedded Tomcat 8.5+
* MySQL Database
* Maven installed version
* Eclipse STS
* Postman (Client)



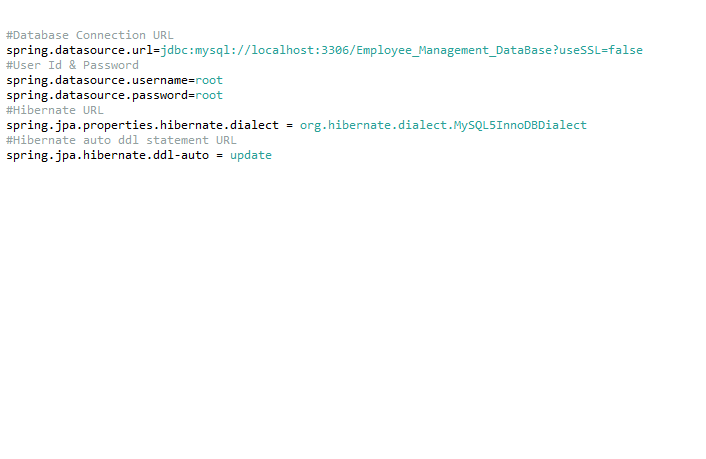
1. The **controller class** is **responsible for processing incoming REST API requests**, preparing a model, and returning the view to be rendered as a response. The controller classes in spring are annotated either by the **@Controller or the @RestController annotation**.

2. **@Service** annotation is used in your **service layer** and annotates classes that perform service tasks, often you don't use it but, in any case, you use this annotation to represent a best practice.

3. **DAO layer (or) Repository layer** is used to interact with a database.

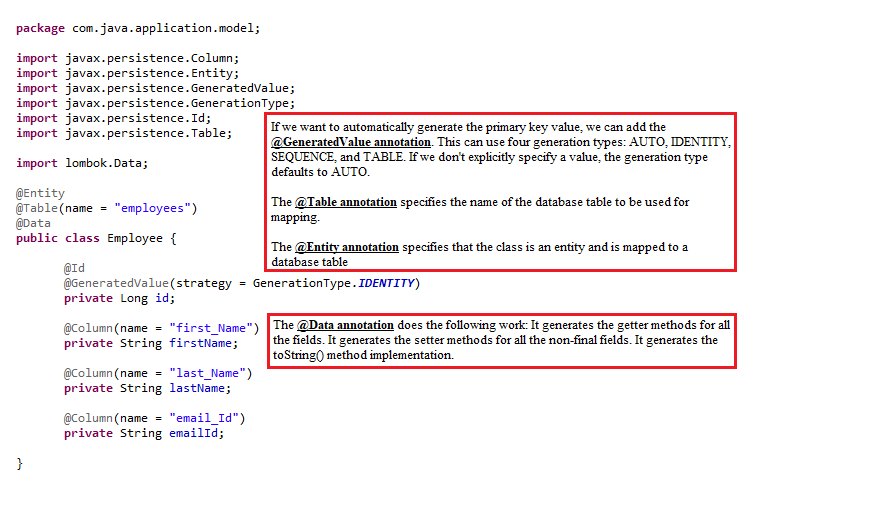
**1. Configure MySQL Database:**

This is **application.properties**:

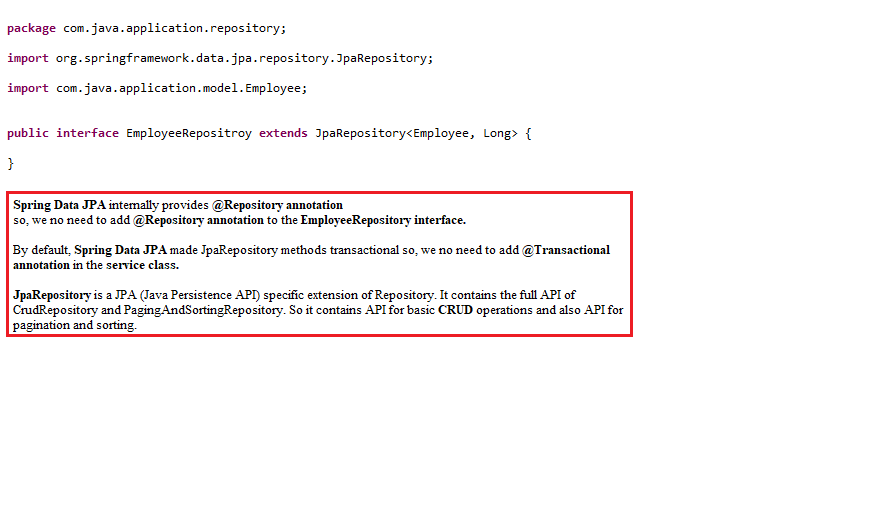


**2. Create JPA Entity and JPA Repository:**

This is **JPA Entity**:



This is **JPA Repository:**

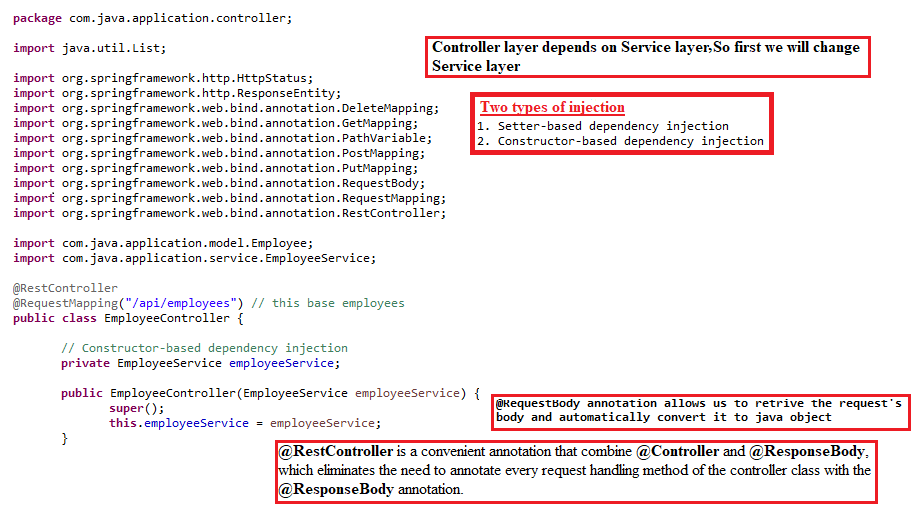
****

**3. Create Custom Exception:**

This is the **custom exception**:



**4. Employee Controller:**

****

****

****

**5. Service layer:**

Text

Description automatically generated

HTTP Verb

POST 🡪 Create

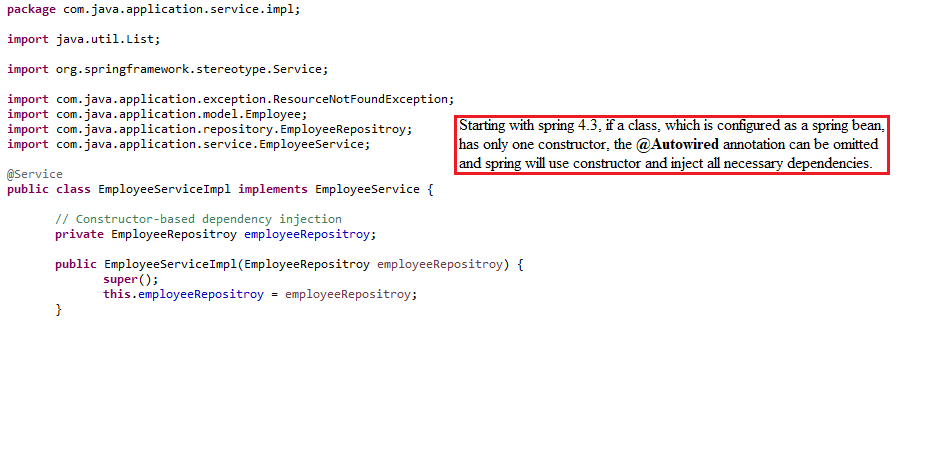
GET 🡪 Read

PUT 🡪 Update / Replace

PATCH 🡪 Update / Modify

DELETE 🡪 Delete

**6. Employee Service Implementation:**

****

****

****

**Spring boot Interview Question and Answers: Important spring boot concept for interview**

1. What is a Spring boot? And what problem spring boot solves?

2. Explain a few important spring boot key features?

3. What is Spring Boot Autoconfiguration?

4. How Spring boot internally works and Explain run() method in spring boot?

5. What are the different ways to create a spring boot application?

6. Explain @SpringBootApplication, @Configuration and @ComponentScan annotations

7. What is Spring boot starters and name a few important spring boot starter dependencies?

8. What is Spring Boot Starter parent?

9. How many types of projects we can create using spring boot?

10. Explain the types of Embedded Server in spring boot? How to change the default server port?

11. How to run the spring boot application from the command line?

12. Describe the flow of REST API HTTP requests through the spring boot application?

13. Explain Spring boot MVC application architecture (Three Tier(Three Layer) Architecture)

14. How to create and deploy spring boot WAR to External security Auto Configuration?

15. What is spring boot DevTools used for?

16. How to add Security to spring boot and Explain Security Auto Configuration?

17. What is Spring Data REST used for?

18. Have you used profiles in your Spring Boot project? If yes, Explain briefly?