Stage 4:

Topics learnt in stage 4

* Spring REST
* Introduction to AWS – EC2, RDS, Dynommo DB
* Spring Microservices
* AWS – CI/CD, Apache Kafka and other services

Spring ReSTful Webservices

ReST stands for Representational State Transfer

What is a webservice

Webservices make 2 applications written in different languages to communicate with each other

Usually webservices shares the data in one common format like JSON/XML that end applications will convert to respective language it is written in.

Types of Webservices

There are two types of webservices

1. ReST: It is a new approach uses XML/JSON/CSV/Text and so on for structuring the data, it can intercommunicate with mobile devices, enterprise applications, web applications, desktop applications and so on
2. SOAP: It is an old approach uses only XML, it can intercommunicate with only enterprise applications.

How to create ReSTful webservices

We need to use Spring Framework to create ReSTful webservices, we must have knowledge on

* Spring Data JPA or DAO layer
* Spring Boot
* MVC pattern
* Spring annotations & Dependency Injection
* Eclipse IDE

Topics Remaining:

Exception Handling

Spring Security

Revision on Spring Boot Concepts

Spring Boot & Spring Data JPA with H2 database

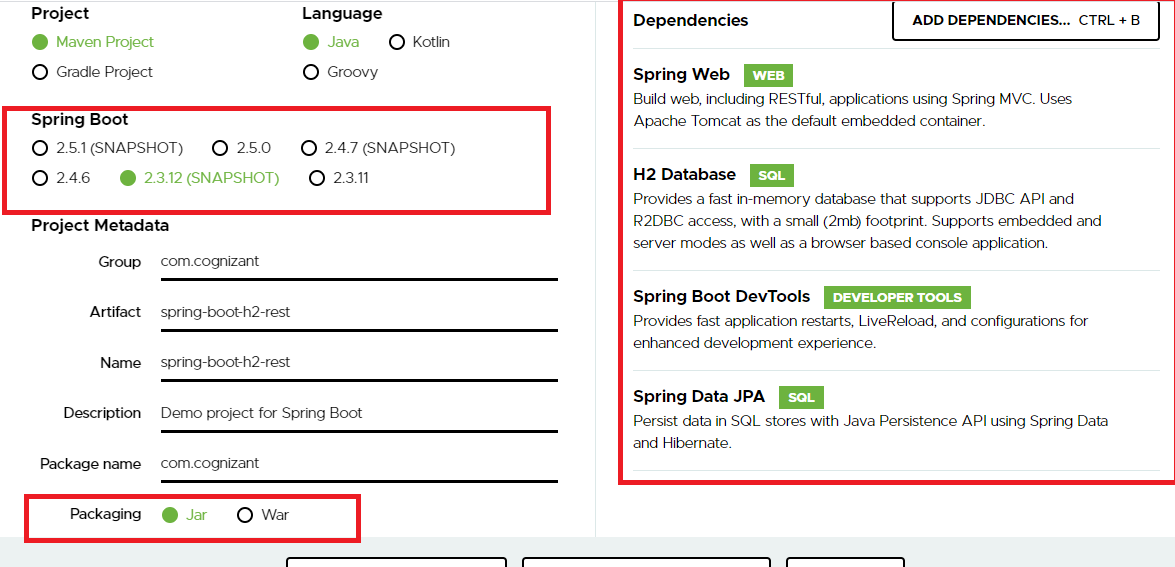
H2 database is an in-memory database which is removed once the application restarts, it is mainly used at the time of development & testing.

Since Spring Boot provides you the feature which separates application code & configurations at the time of development, testing & deployment the changes made in the application.properties would not affect the application logic.

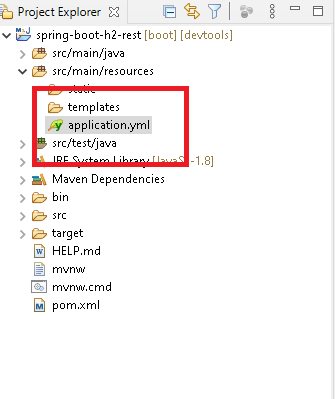
Dependencies required for spring boot & spring data jpa with RESTful webservices

1. Web
2. Devtools
3. H2 database
4. JPA

Spring Initialzr



Change the .properties to .yml so that you can avoid writing repeated properties



Since we are connecting to the database we need to have entities, service, service interface & repositories and also we need to configure the application.yml

application.yml

server:

port: 8081

servlet:

context-path: /api

spring:

datasource:

driver-class-name: org.h2.Driver

url: jdbc:h2:mem:mydb

username: admin

password: admin

Note: Since we are using in-memory database username & password can be anything including database name also

Employee.java

**package** com.cognizant;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

@Entity

**public** **class** Employee {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** **int** employeeId;

**private** String name;

**private** **double** salary;

**public** **int** getEmployeeId() {

**return** employeeId;

}

**public** **void** setEmployeeId(**int** employeeId) {

**this**.employeeId = employeeId;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **double** getSalary() {

**return** salary;

}

**public** **void** setSalary(**double** salary) {

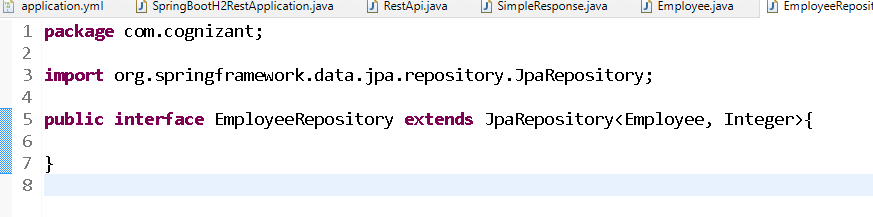
**this**.salary = salary;

}

}

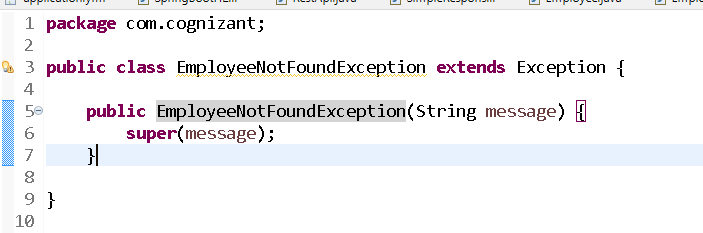
Creating the repository to perform the CURD operations

EmployeeRepository.java



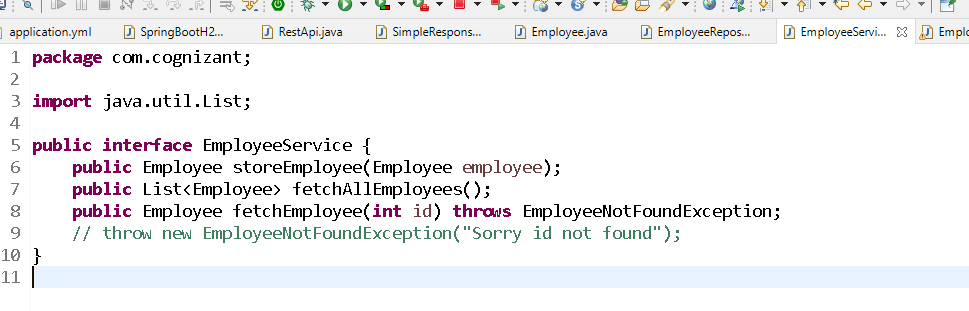
Create one customized exception so that you can throw the exception when necessary

EmployeeNotFoundException.java



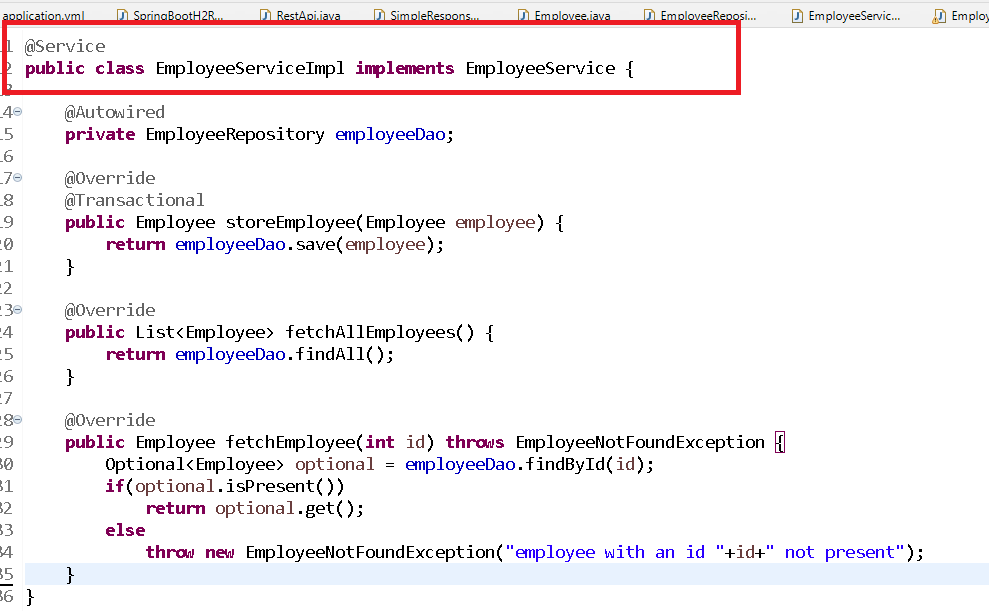
Implementing the service with the help of interface

EmployeeService.java



Implementing the interface

EmployeeServiceImpl.java



Now the @RestController classes call the service methods

**package** com.cognizant;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.MediaType;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("employees")

**public** **class** RestApi {

@Autowired

**private** EmployeeService service;

@PostMapping(consumes = MediaType.***APPLICATION\_JSON\_VALUE***,

produces = MediaType.***APPLICATION\_JSON\_VALUE***)

**public** ResponseEntity<Object> storeApi(@RequestBody Employee emp) {

**return** ResponseEntity.*status*(HttpStatus.***CREATED***).body(service.storeEmployee(emp));

}

@GetMapping(produces = MediaType.***APPLICATION\_JSON\_VALUE***)

**public** ResponseEntity<Object> getAllApi() {

**return** ResponseEntity.*status*(HttpStatus.***OK***).body(service.fetchAllEmployees());

}

@GetMapping(path = "{eid}")

**public** ResponseEntity<Object> getEmployee(@PathVariable("eid") **int** id) {

ResponseEntity<Object> response = **null**;

**try** {

Employee employee = service.fetchEmployee(id);

response = ResponseEntity.*status*(200).body(employee);

} **catch** (EmployeeNotFoundException e) {

String err = e.getMessage();

SimpleResponse sr = **new** SimpleResponse();

sr.setMessage(err);

response = ResponseEntity.*status*(HttpStatus.***NOT\_FOUND***).body(sr);

}

**return** response;

}

}

Output:

