Step 1: Create Spring Boot Project

Use **Spring Initializr**: - Project: Maven - Language: Java - Spring Boot: latest stable - Group: com.example - Artifact: carsapi - Packaging: Jar - Java: 17 (or 11) - Dependencies: - Spring Web

---

Step 2: Project Structure

```
com.example.carsapi
 ├── controller
 │     └── CarController.java
 ├── service
 │     └── CarService.java
 ├── repository
 │     └── CarRepository.java
 ├── model
 │     └── Car.java
 └── CarsapiApplication.java
```

Step 3: Configure Port (8989)

application.properties

```
server.port=8989
```

Step 4: Car Model

```java
package com.example.carsapi.model;

public class Car {
    private int id;
    private String make;
    private String model;
    private double price;
    private int year;

    public Car() {}

    public Car(int id, String make, String model, double price, int year) {
        this.id = id;
        this.make = make;
        this.model = model;
        this.price = price;
        this.year = year;
    }

    // getters and setters
}
```

Step 5: Repository Layer (In-Memory Store)

```
package com.example.carsapi.repository;

import com.example.carsapi.model.Car;
import org.springframework.stereotype.Repository;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

@Repository
public class CarRepository {

    private List<Car> cars = new ArrayList<>();

    public List<Car> findAll() {
        return cars;
    }

    public Car findById(int id) {
        return cars.stream()
                .filter(c -> c.getId() == id)
                .findFirst()
                .orElse(null);
    }

    public void save(Car car) {
        cars.add(car);
    }

    public void updateByMake(String make, Car updatedCar) {
        for (Car car : cars) {
            if (car.getMake().equalsIgnoreCase(make)) {
                car.setModel(updatedCar.getModel());
                car.setPrice(updatedCar.getPrice());
                car.setYear(updatedCar.getYear());
            }
        }
    }

    public void deleteByPriceRange(double min, double max) {
        Iterator<Car> iterator = cars.iterator();
        while (iterator.hasNext()) {
            Car car = iterator.next();
            if (car.getPrice() >= min && car.getPrice() <= max) {
                iterator.remove();
            }
        }
    }

    public List<Car> findByMakeAndYear(String make, int year) {
        List<Car> result = new ArrayList<>();
        for (Car car : cars) {
            if (car.getMake().equalsIgnoreCase(make) && car.getYear() == year) {
                result.add(car);
            }
        }
        return result;
    }
}
```

Step 6: Service Layer

```
package com.example.carsapi.service;

import com.example.carsapi.model.Car;
import com.example.carsapi.repository.CarRepository;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class CarService {

    private final CarRepository repository;

    public CarService(CarRepository repository) {
        this.repository = repository;
    }

    public List<Car> getAllCars() {
        return repository.findAll();
    }

    public Car getCarById(int id) {
        return repository.findById(id);
    }

    public void addCar(Car car) {
        repository.save(car);
    }

    public void updateCarByMake(String make, Car car) {
        repository.updateByMake(make, car);
    }

    public void deleteCarsByPrice(double min, double max) {
        repository.deleteByPriceRange(min, max);
    }

    public List<Car> getCarsByMakeAndYear(String make, int year) {
        return repository.findByMakeAndYear(make, year);
    }
}
```

Step 7: Controller Layer (REST API)

```
package com.example.carsapi.controller;

import com.example.carsapi.model.Car;
import com.example.carsapi.service.CarService;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/cars-api")
public class CarController {

    private final CarService service;
    public CarController(CarService service) {
        this.service = service;
    }

    @GetMapping
    public List<Car> getAllCars() {
        return service.getAllCars();
    }

    @GetMapping("/{id}")
    public Car getCarById(@PathVariable int id) {
        return service.getCarById(id);
    }

    @DeleteMapping("/{minPrice}/{maxPrice}")
    public void deleteByPrice(@PathVariable double minPrice,
                              @PathVariable double maxPrice) {
        service.deleteCarsByPrice(minPrice, maxPrice);
    }

    @GetMapping("/{make}/{year}")
    public List<Car> getByMakeAndYear(@PathVariable String make,
                                      @PathVariable int year) {
        return service.getCarsByMakeAndYear(make, year);
    }

    @PutMapping("/{make}")
    public void updateCar(@PathVariable String make,
                          @RequestBody Car car) {
        service.updateCarByMake(make, car);
    }

    @PostMapping
    public void addCar(@RequestBody Car car) {
        service.addCar(car);
    }
}
```

Step 8: Run the Application

```
@SpringBootApplication
public class CarsapiApplication {
    public static void main(String[] args) {
        SpringApplication.run(CarsapiApplication.class, args);
    }
}
```

Step 9: Test Using Postman

- **POST** `http://localhost:8989/cars-api`
- **GET** `http://localhost:8989/cars-api`
- **GET** `http://localhost:8989/cars-api/1`
- **GET** `http://localhost:8989/cars-api/Toyota/2020`
- **DELETE** `http://localhost:8989/cars-api/500000/1000000`
- **PUT** `http://localhost:8989/cars-api/Toyota`

✔️This follows:

- REST principles
- Layered architecture
- In-memory persistence
- Embedded Tomcat on port 8989

If you want: **exception handling, validation, or Swagger**, tell me.