Spring Boot Backend Development Steps for Online Car Service Station

# Step 1: Setup Spring Boot Project

1. Generate a Spring Boot Project:  
 - Use Spring Initializr (https://start.spring.io/) to generate your project. Select:  
 - Project: Maven Project  
 - Language: Java  
 - Spring Boot version: 2.7.x or later  
 - Dependencies: Spring Web, Spring Data JPA, Spring Security, H2 (or your preferred database), Spring Boot DevTools (optional for development), and Lombok (for reducing boilerplate code).  
 - Download the generated .zip file and extract it.  
2. Directory Structure:  
 Your project should have the following basic structure:  
 - src/  
 └── main/  
 ├── java/  
 │ └── com/  
 │ └── carservice/  
 │ ├── controller/  
 │ ├── model/  
 │ ├── repository/  
 │ ├── service/  
 │ └── CarServiceApplication.java  
 ├── resources/  
 │ ├── application.properties  
 │ └── static/  
 └── test/  
 └── java/  
 └── com/  
 └── carservice/

# Step 2: Define the Entity Classes

## User Entity

```java  
// User Entity class

package com.carservice.model;  
  
import javax.persistence.\*;  
import lombok.\*;  
  
@Entity  
@Table(name = "users")  
@Data  
public class User {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long userId;  
  
 // Add your fields here  
}

```

## Vehicle Entity

```java  
// Vehicle Entity class

package com.carservice.model;  
  
import javax.persistence.\*;  
import lombok.\*;  
  
@Entity  
@Table(name = "vehicles")  
@Data  
public class Vehicle {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long vehicleId;  
  
 // Add your fields here  
}

```

## ServiceCenter Entity

```java  
// ServiceCenter Entity class

package com.carservice.model;  
  
import javax.persistence.\*;  
import lombok.\*;  
  
@Entity  
@Table(name = "servicecenters")  
@Data  
public class ServiceCenter {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long servicecenterId;  
  
 // Add your fields here  
}

```

## Service Entity

```java  
// Service Entity class

package com.carservice.model;  
  
import javax.persistence.\*;  
import lombok.\*;  
  
@Entity  
@Table(name = "services")  
@Data  
public class Service {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long serviceId;  
  
 // Add your fields here  
}

```

## Booking Entity

```java  
// Booking Entity class

package com.carservice.model;  
  
import javax.persistence.\*;  
import lombok.\*;  
  
@Entity  
@Table(name = "bookings")  
@Data  
public class Booking {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long bookingId;  
  
 // Add your fields here  
}

```

## Payment Entity

```java  
// Payment Entity class

package com.carservice.model;  
  
import javax.persistence.\*;  
import lombok.\*;  
  
@Entity  
@Table(name = "payments")  
@Data  
public class Payment {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long paymentId;  
  
 // Add your fields here  
}

```

# Step 3: Create Repositories

Each entity needs a repository interface to interact with the database. For example, UserRepository for the User entity:

```java  
package com.carservice.repository;  
  
import com.carservice.model.User;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface UserRepository extends JpaRepository<User, Long> {  
 // Custom query methods if needed  
}  
```

# Step 4: Service Layer

In the service layer, you will create business logic. For example, in UserService.java:

```java  
package com.carservice.service;  
  
import com.carservice.model.User;  
import com.carservice.repository.UserRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class UserService {  
 @Autowired  
 private UserRepository userRepository;  
  
 public List<User> getAllUsers() {  
 return userRepository.findAll();  
 }  
  
 public User getUserById(Long id) {  
 return userRepository.findById(id).orElse(null);  
 }  
  
 public User createUser(User user) {  
 return userRepository.save(user);  
 }  
  
 public void deleteUser(Long id) {  
 userRepository.deleteById(id);  
 }  
}  
```

# Step 5: Controller Layer

Create RESTful APIs in the controller layer. For example, UserController.java:

```java  
package com.carservice.controller;  
  
import com.carservice.model.User;  
import com.carservice.service.UserService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/users")  
public class UserController {  
 @Autowired  
 private UserService userService;  
  
 @GetMapping  
 public List<User> getAllUsers() {  
 return userService.getAllUsers();  
 }  
  
 @GetMapping("/{id}")  
 public User getUserById(@PathVariable Long id) {  
 return userService.getUserById(id);  
 }  
  
 @PostMapping  
 public User createUser(@RequestBody User user) {  
 return userService.createUser(user);  
 }  
  
 @DeleteMapping("/{id}")  
 public void deleteUser(@PathVariable Long id) {  
 userService.deleteUser(id);  
 }  
}  
```

# Step 6: Application Properties

In the `src/main/resources/application.properties`, set up your database configurations:

```properties  
spring.datasource.url=jdbc:h2:mem:testdb  
spring.datasource.driverClassName=org.h2.Driver  
spring.datasource.username=sa  
spring.datasource.password=password  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true  
```