URL to GitHup Repository: https://github.com/aliustunyer/Web-API-Design-with-Spring-Boot.git

URL to Youtube Video : https://youtu.be/HRTbN4jpZD8

```
package com.promineotech.jeep.controller.support;


public class CreateOrderTestSupport extends
BaseTest {

protected String createOrderBody() {

//@formatter: off

return "{ \"customer\":\"MORISON_LINA\"," +

"\"model\":\"WRANGLER\"," +

"\"trim\":\"Sport Altitude\"," +

"\"doors\":4," +

"\"color\":\"EXT_NACHO\"," +

"\"engine\":\"2_0_TURBO\"," +

"\"tire\":\"35_TOYO\"," +

"\"options\":[" +

"\"DOOR_QUAD_4\"," +

"\"EXT_AEV_LIFT\"," +

"\"EXT_WARN_WINCH\"," +

"\"EXT_WARN_BUMPER_FRONT\"," +

"\"EXT_WARN_BUMPER_REAR\"," +

"\"EXT_ARB_COMPRESSOR\"" +

"]" +

"}";
```

```
//formatter : on


package com.promineotech.jeep.controller;

import static org.assertj.core.api.Assertions.assertThat;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpEntity;
import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.ArgumentMatchers.isNotNull;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.context.SpringBootTest.WebEnvironment;
import org.springframework.http.HttpMethod;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.test.context.ActiveProfiles;
import org.springframework.test.context.jdbc.Sql;
import org.springframework.test.context.jdbc.SqlConfig;
import org.springframework.test.jdbc.JdbcTestUtils;
import com.promineotech.jeep.controller.support.CreateOrderTestSupport;
import com.promineotech.jeep.entity.JeepModel;
import com.promineotech.jeep.entity.Order;

@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
@ActiveProfiles("test")
@Sql(scripts = {
    "classpath:flyway/migrations/V1.0__Jeep_Schema.sql",
    "classpath:flyway/migrations/V1.1__Jeep_Data.sql"},
    config = @SqlConfig(encoding = "utf-8"))

class CreateOrderTest extends CreateOrderTestSupport {

  @Autowired
  private JdbcTemplate jdbcTemplate;

  @Test
  void testCreateOrderReturnsSuccess201() {

    //Given : an order as JSON
    String body =createOrderBody();
    String uri = getBaseUriForOrders();

    int numRowsOrders = JdbcTestUtils.countRowsInTable(jdbcTemplate, "orders");
    int numRowsOptions = JdbcTestUtils.countRowsInTable(jdbcTemplate, "order_options");

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
//send a POST request to a specific URI with a custom string body and headers.
//Then,check if the response status code is equal to 201 Created.
//If it is, the test passes; if it isn't, the test fails.

//To send the request,use a RestTemplate object that is created elsewhere.
//The exchange() method is used to send the request and receive the response.
//The request includes the URI, HTTP method (POST), request body, and custom headers.

HttpEntity<String> bodyEntity = new HttpEntity<>(body, headers);

    // When : the order is sent
//The response is checked using the assertThat() method.
//If the status code is 201 Created, the test passes; otherwise, it fails.
ResponseEntity<Order> response = getRestTemplate().exchange(uri, HttpMethod.POST,
        bodyEntity, Order.class);

    // Then : a 201 status is returned
        assertThat (response.getStatusCode()).isEqualTo(HttpStatus.CREATED);

    // And  : the returned order is correct
        assertThat(response.getBody()).isNotNull();

        Order order = response.getBody();

assertThat(order.getCustomer().getCustomerId()).isEqualTo("MORISON_LINA");
assertThat(order.getModel().getModelId()).isEqualTo(JeepModel.WRANGLER);
assertThat(order.getModel().getTrimLevel()).isEqualTo("Sport Altitude");
assertThat(order.getModel().getNumDoors()).isEqualTo(4);
assertThat(order.getColor().getColorId()).isEqualTo("EXT_NACHO");
assertThat(order.getEngine().getEngineId()).isEqualTo("2_0_TURBO");
assertThat(order.getTire().getTireId()).isEqualTo("35_TOYO");
assertThat(order.getOptions()).hasSize(6);

assertThat(JdbcTestUtils.countRowsInTable(jdbcTemplate, "orders"))
         .isEqualTo(numRowsOrders + 1);
assertThat(JdbcTestUtils.countRowsInTable(jdbcTemplate, "order_options"))
         .isEqualTo(numRowsOptions + 6);
  }




}
```

```
package com.promineotech.jeep.entity;
```

```
import java.util.List;
import javax.validation.constraints.Max;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Positive;
import org.hibernate.validator.constraints.Length;
import lombok.Data;


@Data
public class OrderRequest {
@NotNull
@Length(max = 30)
@Pattern (regexp ="[\\w\\s]*")
private String customer;
@NotNull
private JeepModel model;
@NotNull
@Length(max = 30)
@Pattern (regexp ="[\\w\\s]*")
private String trim;
@Positive
@Min(2)
@Max(4)
private int doors;
@NotNull
@Length(max = 30)
@Pattern (regexp ="[\\w\\s]*")
private String color;
@NotNull
@Length(max = 30)
@Pattern (regexp ="[\\w\\s]*")
private String engine;
@NotNull
@Length(max = 30)
@Pattern (regexp ="[\\w\\s]*")
private String tire;


private List<@NotNull @Length(max = 30) @Pattern
(regexp ="[\\w\\s]*")String> options;
}
```

```
package com.promineotech.jeep.service;


import com.promineotech.jeep.entity.Order;
import com.promineotech.jeep.entity.OrderRequest;
```

```
public interface JeepOrderService {

  Order createOrder(OrderRequest orderRequest);



  }




package com.promineotech.jeep.service;

import java.math.BigDecimal;
import java.util.List;
import java.util.NoSuchElementException;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import com.promineotech.jeep.dao.JeepOrderDao;
import com.promineotech.jeep.entity.Color;
import com.promineotech.jeep.entity.Customer;
import com.promineotech.jeep.entity.Engine;
import com.promineotech.jeep.entity.Jeep;
import com.promineotech.jeep.entity.Option;
import com.promineotech.jeep.entity.Order;
import com.promineotech.jeep.entity.OrderRequest;
import com.promineotech.jeep.entity.Tire;

@Service
public class DefaultJeepOrderService implements JeepOrderService {
```

```
@Autowired
private JeepOrderDao jeepOrderDao;

@Transactional
@Override
public Order createOrder(OrderRequest orderRequest) {
Customer customer = getCustomer(orderRequest);
Jeep jeep = getModel(orderRequest);
Color color = getColor(orderRequest);
Engine engine = getEngine(orderRequest);
Tire tire = getTire(orderRequest);
List<Option> options = getOption(orderRequest);

BigDecimal price = jeep.getBasePrice().add(color.getPrice())
.add(engine.getPrice()).add(tire.getPrice());

for(Option option : options) {
price = price.add(option.getPrice());
}
return jeepOrderDao.saveOrder(customer, jeep, color, engine, tire, price, options);
}

/**
 *
 * @param orderRequest
 * @return
 */
private List<Option> getOption(OrderRequest orderRequest) {
return jeepOrderDao.fetchOptions(orderRequest.getOptions());
}
```

```
/**
 *
 * @param orderRequest
 * @return
 */
private Tire getTire(OrderRequest orderRequest) {
return jeepOrderDao.fetchTire(orderRequest.getTire()).orElseThrow(
() -> new NoSuchElementException("Tire with ID=" + orderRequest.getTire() + " was not found"));
}

/**
 *
 * @param orderRequest
 * @return
 */
private Engine getEngine(OrderRequest orderRequest) {
return jeepOrderDao.fetchEngine(orderRequest.getEngine()).orElseThrow(
() -> new NoSuchElementException("Engine with ID=" + orderRequest.getEngine() + " was not
found"));
}

/**
 *
 * @param orderRequest
 * @return
 */
private Color getColor(OrderRequest orderRequest) {
return jeepOrderDao.fetchColor(orderRequest.getColor()).orElseThrow(
() -> new NoSuchElementException("Color with ID=" + orderRequest.getColor() + " was not found"));
}

/**
 *
 * @param orderRequest
 * @return
 */
private Jeep getModel(OrderRequest orderRequest) {
return jeepOrderDao.fetchModel(orderRequest.getModel(), orderRequest.getTrim(),
orderRequest.getDoors())
.orElseThrow(() -> new NoSuchElementException("Model with ID=" + orderRequest.getModel() + ",
trim="
+ orderRequest.getTrim() + orderRequest.getDoors() + " was not found"));
}

/**
 *
 * @param orderRequest
 * @return
 */
private Customer getCustomer(OrderRequest orderRequest) {
return jeepOrderDao.fetchCustomer(orderRequest.getCustomer()).orElseThrow(
() -> new NoSuchElementException("Customer with ID=" + orderRequest.getCustomer() + " was not
found"));
}
}

package com.promineotech.jeep.controller;
import com.promineotech.jeep.entity.Order;
import com.promineotech.jeep.entity.OrderRequest;
import java.util.List;
import javax.validation.Valid;
import javax.validation.constraints.Pattern;
import org.hibernate.validator.constraints.Length;
import org.springframework.http.HttpStatus;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
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.RequestParam;
import org.springframework.web.bind.annotation.ResponseStatus;
import com.promineotech.jeep.Constants;
import com.promineotech.jeep.entity.Jeep;
import com.promineotech.jeep.entity.JeepModel;
import io.swagger.v3.oas.annotations.OpenAPIDefinition;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;
import io.swagger.v3.oas.annotations.info.Info;
import io.swagger.v3.oas.annotations.media.Content;
import io.swagger.v3.oas.annotations.media.Schema;
import io.swagger.v3.oas.annotations.responses.ApiResponse;
import io.swagger.v3.oas.annotations.servers.Server;
@Validated
@RequestMapping("/orders")
@OpenAPIDefinition(info = @Info(title = "Jeep Order Service"), servers = {
@Server(url = "http://localhost:8080", description = "Local server.")})

public interface JeepOrderController {



  //@formatter:off
  @Operation(
      summary = "Create an order for a Jeep",
      description = "Returns the created Jeep",
      responses = {
          @ApiResponse(
              responseCode="201",
              description = "The created Jeep is returned",
              content = @Content(
                mediaType = "application/json",
                schema = @Schema(implementation = Order.class))),
          @ApiResponse(
              responseCode = "400",
              description = "The request parameters are invalid",
              content = @Content(mediaType = "application/json")),
          @ApiResponse(
              responseCode = "404",
              description = "A Jeep component was not found with the input criteria",
              content = @Content(mediaType = "application/json")),
          @ApiResponse(
              responseCode = "500",
              description = "An unplanned error occured",
              content = @Content(mediaType = "application/json")),
      },
       parameters = {
          @Parameter(name = "orderRequest",
              required = true,
              description = "The order as JSON"),
      }
    )
  @PostMapping
  @ResponseStatus(code = HttpStatus.CREATED)
  Order createOrder(@Valid @RequestBody OrderRequest orderRequest);
}


package com.promineotech.jeep.controller;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RestController;
import com.promineotech.jeep.entity.Order;
import com.promineotech.jeep.entity.OrderRequest;
import com.promineotech.jeep.service.JeepOrderService;
import lombok.extern.slf4j.Slf4j;

@RestController
@Slf4j
public class DefaultJeepOrderController implements JeepOrderController {

  @Autowired
  private JeepOrderService jeepOrderService;

  @Override
  public Order createOrder(OrderRequest orderRequest) {
    log.debug("Order={}", orderRequest);
    return jeepOrderService.createOrder(orderRequest);
  }

}

package com.promineotech.jeep.dao;


import java.math.BigDecimal;
import java.util.List;
import java.util.Optional;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Pattern;
import org.hibernate.validator.constraints.Length;
import com.promineotech.jeep.entity.Color;
import com.promineotech.jeep.entity.Customer;
import com.promineotech.jeep.entity.Engine;
import com.promineotech.jeep.entity.Jeep;
import com.promineotech.jeep.entity.JeepModel;
import com.promineotech.jeep.entity.Option;
import com.promineotech.jeep.entity.Order;
import com.promineotech.jeep.entity.OrderRequest;
import com.promineotech.jeep.entity.Tire;

public interface JeepOrderDao {

  Optional<Customer> fetchCustomer(String customerId);

  Optional<Jeep> fetchModel(JeepModel model, String trim, int doors);

  Optional<Color> fetchColor(String colorId);

  Optional<Engine> fetchEngine(String engineId);

  Optional<Tire> fetchTire(String tireId);

  List<Option> fetchOptions(List<String> optionIds);

  Order saveOrder(Customer customer, Jeep jeep, Color color, Engine engine, Tire tire,
      BigDecimal price, List<Option> options);

}


package com.promineotech.jeep.dao;

import java.util.Map;
import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import org.springframework.jdbc.core.RowMapper;
import java.util.Optional;
import java.math.BigDecimal;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.HashMap;
import java.util.List;
import java.util.LinkedList;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.stereotype.Component;
import com.promineotech.jeep.entity.Color;
import com.promineotech.jeep.entity.Customer;
import com.promineotech.jeep.entity.Engine;
import com.promineotech.jeep.entity.FuelType;
import com.promineotech.jeep.entity.Jeep;
import com.promineotech.jeep.entity.JeepModel;
import com.promineotech.jeep.entity.Option;
import com.promineotech.jeep.entity.OptionType;
import com.promineotech.jeep.entity.Order;
import com.promineotech.jeep.entity.OrderRequest;
import com.promineotech.jeep.entity.Tire;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.ResultSetExtractor;
import org.springframework.jdbc.support.GeneratedKeyHolder;
import org.springframework.jdbc.support.KeyHolder;

@Component
public class DefaultJeepOrderDao implements JeepOrderDao {
  @Autowired
  private NamedParameterJdbcTemplate jdbcTemplate;
```

```
@Override
public Order saveOrder(Customer customer, Jeep jeep, Color color, Engine engine, Tire tire,
    BigDecimal price, List <Option> options) {
  SqlParams params = generateInsertSql(customer, jeep, color, engine, tire, price);

  KeyHolder keyHolder = new GeneratedKeyHolder();
  jdbcTemplate.update(params.sql, params.source, keyHolder);

  Long orderPK = keyHolder.getKey().longValue();
  saveOptions(options, orderPK);
  //formatter : off
  return Order.builder()
      .orderPK(orderPK)
      .customer(customer)
      .model(jeep)
      .color(color)
      .engine(engine)
      .tire(tire)
      .options(options)
      .price(price)
      .build();
  //formatter : on
}

private void saveOptions(List<Option> options, Long orderPK) {

  for (Option option : options){
  SqlParams params =generateInsertSql(option, orderPK);
  jdbcTemplate.update(params.sql, params.source);
  }
  }

  private SqlParams generateInsertSql(Option option, Long orderPK) {

SqlParams params = new SqlParams();

    // @formatter:off
    params.sql = ""
        + "INSERT INTO order_options ("
        + "option_fk, order_fk"
        + ") VALUES ("
        + ":option_fk, :order_fk"
        + ")";
    // @formatter:on

    params.source.addValue("option_fk", option.getOptionPK());
    params.source.addValue("order_fk", orderPK);

    return params;
  }

  private SqlParams generateInsertSql(Customer customer, Jeep jeep, Color color, Engine engine,
      Tire tire, BigDecimal price) {
// @formatter:off
    String sql = ""
        + "INSERT INTO orders ("
        + "customer_fk, color_fk, engine_fk, tire_fk, model_fk, price"
        + ") VALUES ("
        + ":customer_fk, :color_fk, :engine_fk, :tire_fk, :model_fk, :price"
        + ")";
    // @formatter:on
  SqlParams params = new SqlParams();

  params.sql = sql;
  params.source.addValue("customer_fk", customer.getCustomerPK());
  params.source.addValue("color_fk", color.getColorPK());
  params.source.addValue("engine_fk", engine.getEnginePK());
  params.source.addValue("tire_fk", tire.getTirePK());
  params.source.addValue("model_fk", jeep.getModelPK());
  params.source.addValue("price", price);

  return params;
}


@Override
public List<Option> fetchOptions(List<String> optionIds) {
  if (optionIds.isEmpty()) {
    return new LinkedList<>();
  }

  Map<String, Object> params = new HashMap<>();

  // @formatter:off
  String sql = ""
      + "SELECT * "
      + "FROM options "
      + "WHERE option_id IN(";
  // @formatter:on

  for (int index = 0; index < optionIds.size(); index++) {
    String key = "option_" + index;
    sql += ":" + key + ", ";
    params.put(key, optionIds.get(index));
  }

  sql = sql.substring(0, sql.length() - 2);
  sql += ")";

  return jdbcTemplate.query(sql, params, new RowMapper<Option>() {
    @Override
    public Option mapRow(ResultSet rs, int rowNum) throws SQLException {
      // @formatter:off
      return Option.builder()
          .category(OptionType.valueOf(rs.getString("category")))
          .manufacturer(rs.getString("manufacturer"))
          .name(rs.getString("name"))
          .optionId(rs.getString("option_id"))
          .optionPK(rs.getLong("option_pk"))
          .price(rs.getBigDecimal("price"))
          .build();
      // @formatter:on
    }
  });
}


@Override
public Optional<Customer> fetchCustomer(String customerId) {
  String sql =  "SELECT * FROM customers WHERE customer_id = :customer_id";
  Map <String, Object> params = new HashMap<>();
  params.put("customer_id", customerId);
  return Optional.ofNullable(
   jdbcTemplate.query(sql, params,new CustomerResultSetExtractor()));
}
@Override
public Optional<Jeep> fetchModel(JeepModel model, String trim, int doors) {
  // @formatter:off
  String sql = ""
      + "SELECT * "
      + "FROM models "
      + "WHERE model_id = :model_id "
      + "AND trim_level = :trim_level "
      + "AND num_doors = :num_doors";
  // @formatter:on

  Map<String, Object> params = new HashMap<>();
  params.put("model_id", model.toString());
  params.put("trim_level", trim);
  params.put("num_doors", doors);

  return Optional.ofNullable(
      jdbcTemplate.query(sql, params, new ModelResultSetExtractor()));
}
@Override
public Optional<Color> fetchColor(String colorId) {
  // @formatter:off
  String sql = ""
      + "SELECT * "
      + "FROM colors "
      + "WHERE color_id = :color_id";
  // @formatter:on
```

```
    Map<String, Object> params = new HashMap<>();
    params.put("color_id", colorId);

    return Optional.ofNullable(
        jdbcTemplate.query(sql, params, new ColorResultSetExtractor()));
  }

@Override
public Optional<Engine> fetchEngine(String engineId) {
  // @formatter:off
  String sql = ""
      + "SELECT * "
      + "FROM engines "
      + "WHERE engine_id = :engine_id";
  // @formatter:on

  Map<String, Object> params = new HashMap<>();
  params.put("engine_id", engineId);

  return Optional.ofNullable(
      jdbcTemplate.query(sql, params, new EngineResultSetExtractor()));
}
@Override
public Optional<Tire> fetchTire(String tireId) {
  // @formatter:off
  String sql = ""
      + "SELECT * "
      + "FROM tires "
      + "WHERE tire_id = :tire_id";
  // @formatter:on
  Map<String, Object> params = new HashMap<>();
  params.put("tire_id", tireId);

  return Optional.ofNullable(
    jdbcTemplate.query(sql, params, new TireResultSetExtractor()));
}

  class CustomerResultSetExtractor implements ResultSetExtractor<Customer> {
   @Override
   public Customer extractData(ResultSet rs) throws SQLException {
    rs.next();

    // @formatter:off
    return Customer.builder()
        .customerId(rs.getString("customer_id"))
        .customerPK(rs.getLong("customer_pk"))
        .firstName(rs.getString("first_name"))
        .lastName(rs.getString("last_name"))
        .phone(rs.getString("phone"))
        .build();
    // @formatter:on

   }

}
  class ModelResultSetExtractor implements ResultSetExtractor<Jeep> {
   @Override
   public Jeep extractData(ResultSet rs) throws SQLException {
    rs.next();
    // @formatter:off
    return Jeep.builder()
        .basePrice(rs.getBigDecimal("base_price"))
        .modelId(JeepModel.valueOf(rs.getString("model_id")))
        .modelPK(rs.getLong("model_pk"))
        .numDoors(rs.getInt("num_doors"))
        .trimLevel(rs.getString("trim_level"))
        .wheelSize(rs.getInt("wheel_size"))
        .build();
    // @formatter:on
   }
 }
  class ColorResultSetExtractor implements ResultSetExtractor<Color> {
   @Override
   public Color extractData(ResultSet rs) throws SQLException {
    rs.next();

    // @formatter:off
    return Color.builder()
        .color(rs.getString("color"))
        .colorId(rs.getString("color_id"))
        .colorPK(rs.getLong("color_pk"))
        .isExterior(rs.getBoolean("is_exterior"))
        .price(rs.getBigDecimal("price"))
        .build();
    // @formatter:on
   }
  }
  class EngineResultSetExtractor implements ResultSetExtractor<Engine> {
   @Override
   public Engine extractData(ResultSet rs) throws SQLException {
    rs.next();

    // @formatter:off
    return Engine.builder()
        .description(rs.getString("description"))
        .engineId(rs.getString("engine_id"))
        .enginePK(rs.getLong("engine_pk"))
        .fuelType(FuelType.valueOf(rs.getString("fuel_type")))
        .hasStartStop(rs.getBoolean("has_start_stop"))
        .mpgCity(rs.getFloat("mpg_city"))
        .mpgHwy(rs.getFloat("mpg_hwy"))
        .name(rs.getString("name"))
        .price(rs.getBigDecimal("price"))
        .sizeInLiters(rs.getFloat("size_in_liters"))
        .build();
    // @formatter:on
  }
}
class TireResultSetExtractor implements ResultSetExtractor<Tire> {
  @Override
  public Tire extractData(ResultSet rs) throws SQLException {
    rs.next();

    // @formatter:off
    return Tire.builder()
        .manufacturer(rs.getString("manufacturer"))
        .price(rs.getBigDecimal("price"))
        .tireId(rs.getString("tire_id"))
        .tirePK(rs.getLong("tire_pk"))
        .tireSize(rs.getString("tire_size"))
        .warrantyMiles(rs.getInt("warranty_miles"))
```

```
         .build();
      // @formatter:on
    }
  }
class SqlParams {
   String sql;
   MapSqlParameterSource source = new MapSqlParameterSource();
}
  }
```