Welcome to the Final Project for the OTC St. Paul Class.

Project Name: Block Buster Movies

Project Members:

Charlee Thao

Adam Fite

Andrew Cham

Project Contributions:

Group:

Database – Schema SQL File, Data Entry SQL File

Pom File

Component Scanner

Global Error Handler

Main Application

Final Screen Shot Documentation

Charlee Thao:

Movies: Controller, Dao, Entity, Service

Concessions: Controller, Dao, Entity, Service

Adam Fite:

Employees: Controller, Dao, Entity, Service

User Roles: Controller, Dao, Entity, Service

User Validation: Controller, Dao, Entity, Service

Transaction: Controller, Dao, Entity, Service

Transactions Test: Controller, Test Body, Base Test

Andrew Cham :

Stores: Controller, Dao, Entity, Service

Customers: Controller, Dao, Entity, Service

Thanks to everyone who may also have provided help trouble shooting when we ran into trouble.

Now On With The Show!

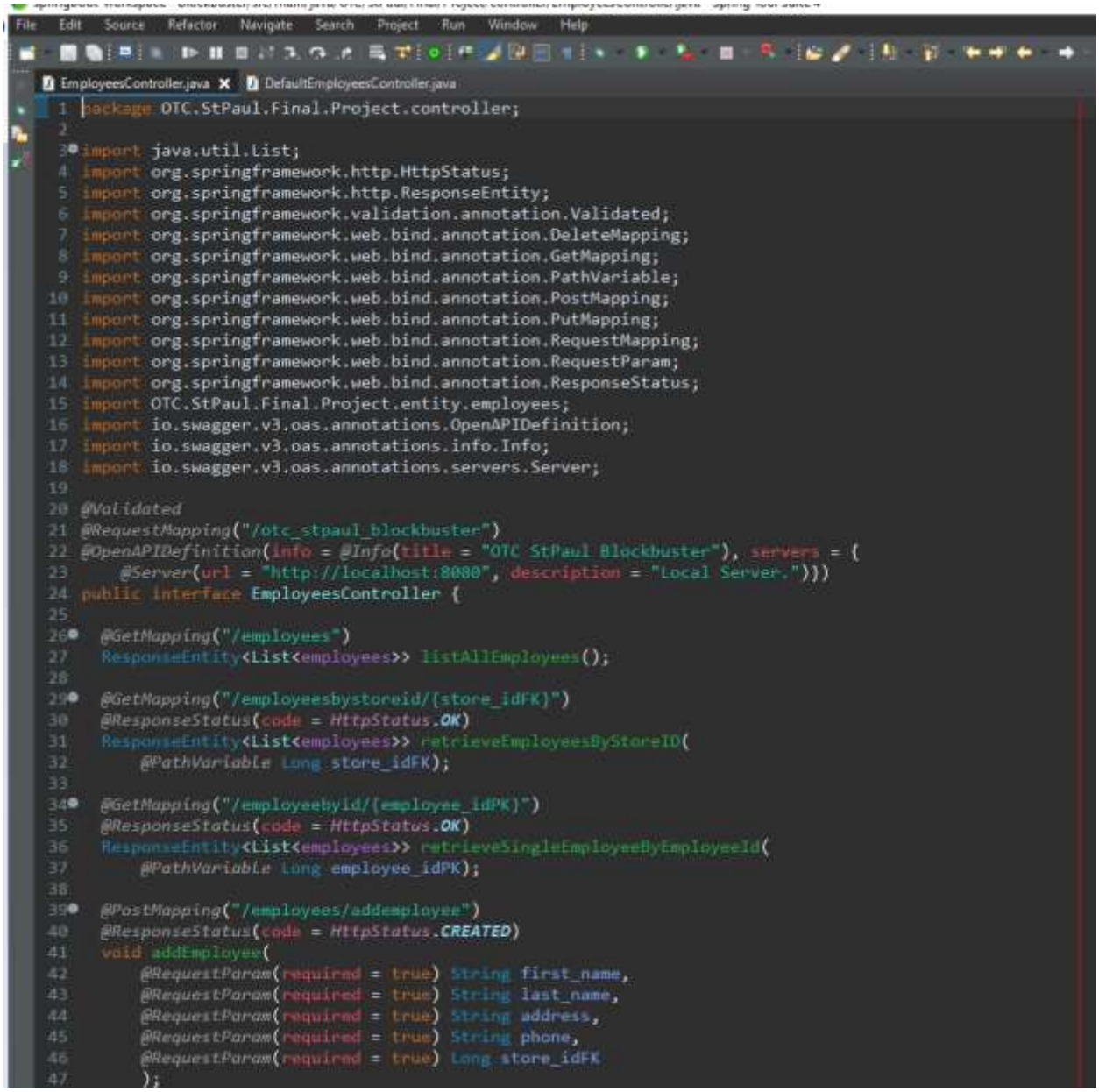
GitHub URL:

https://github.com/cthao0502/Promineo/tree/main/Homework/WeekSpringBoot6

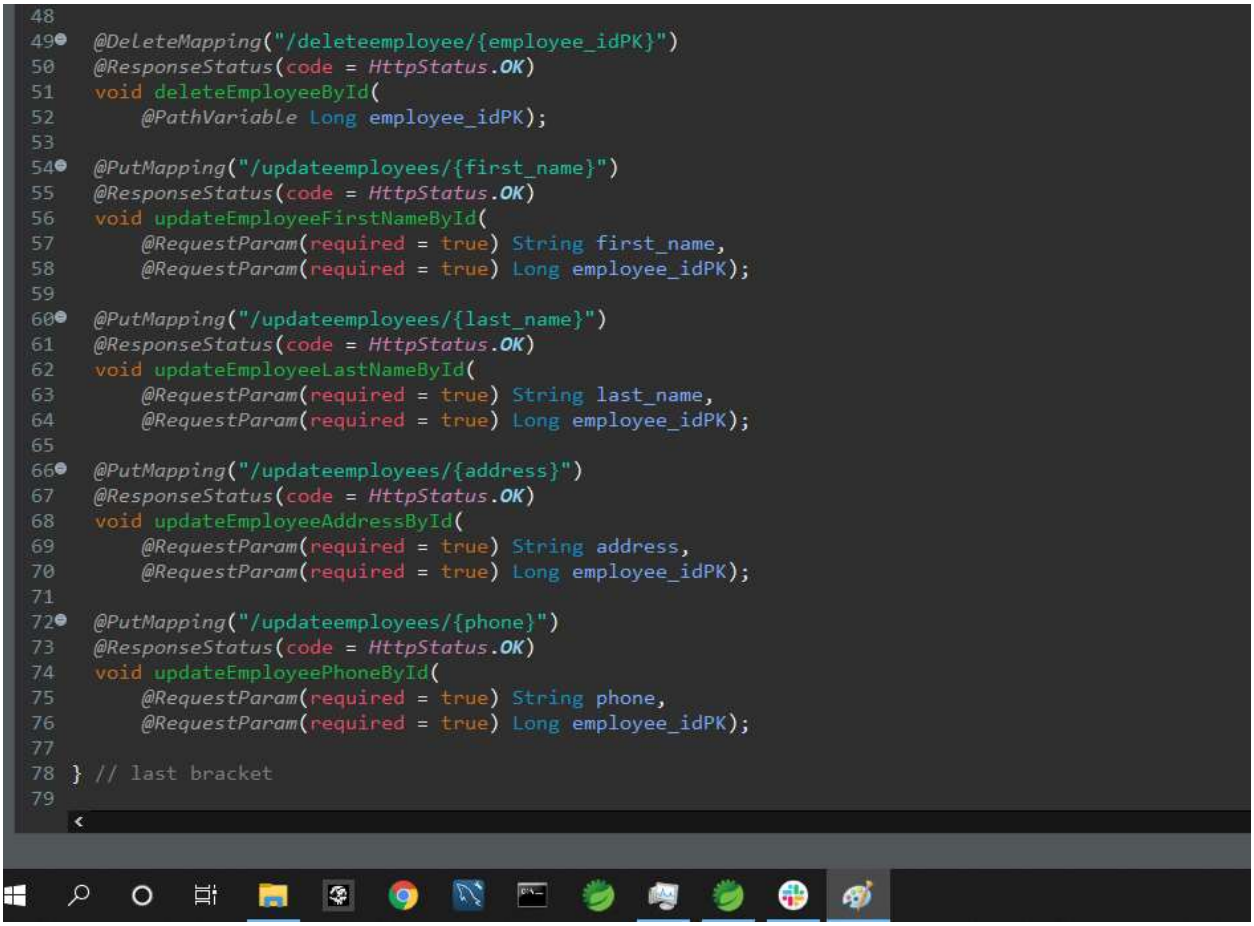
Raw Code:

Controller:

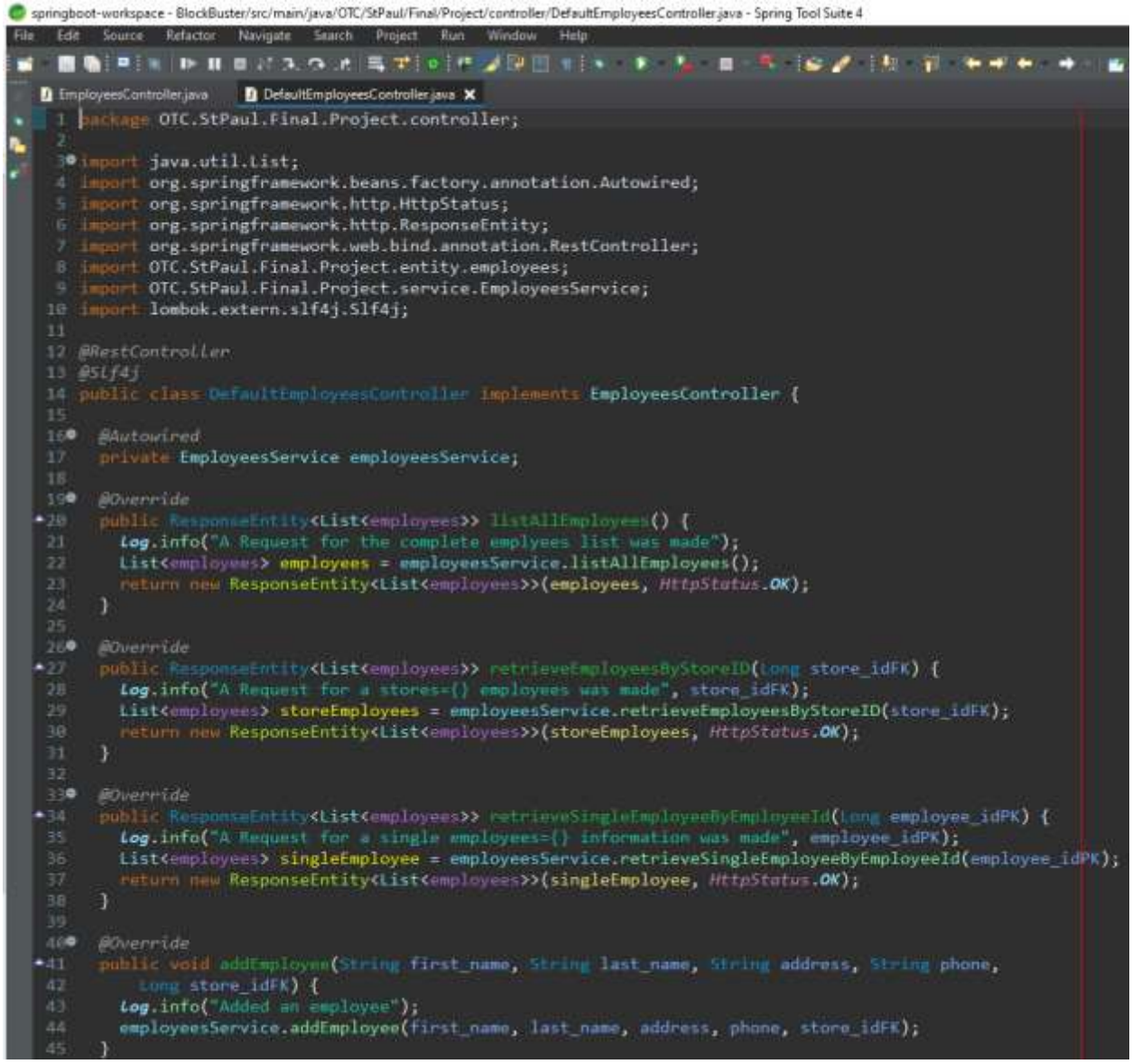
Employees-

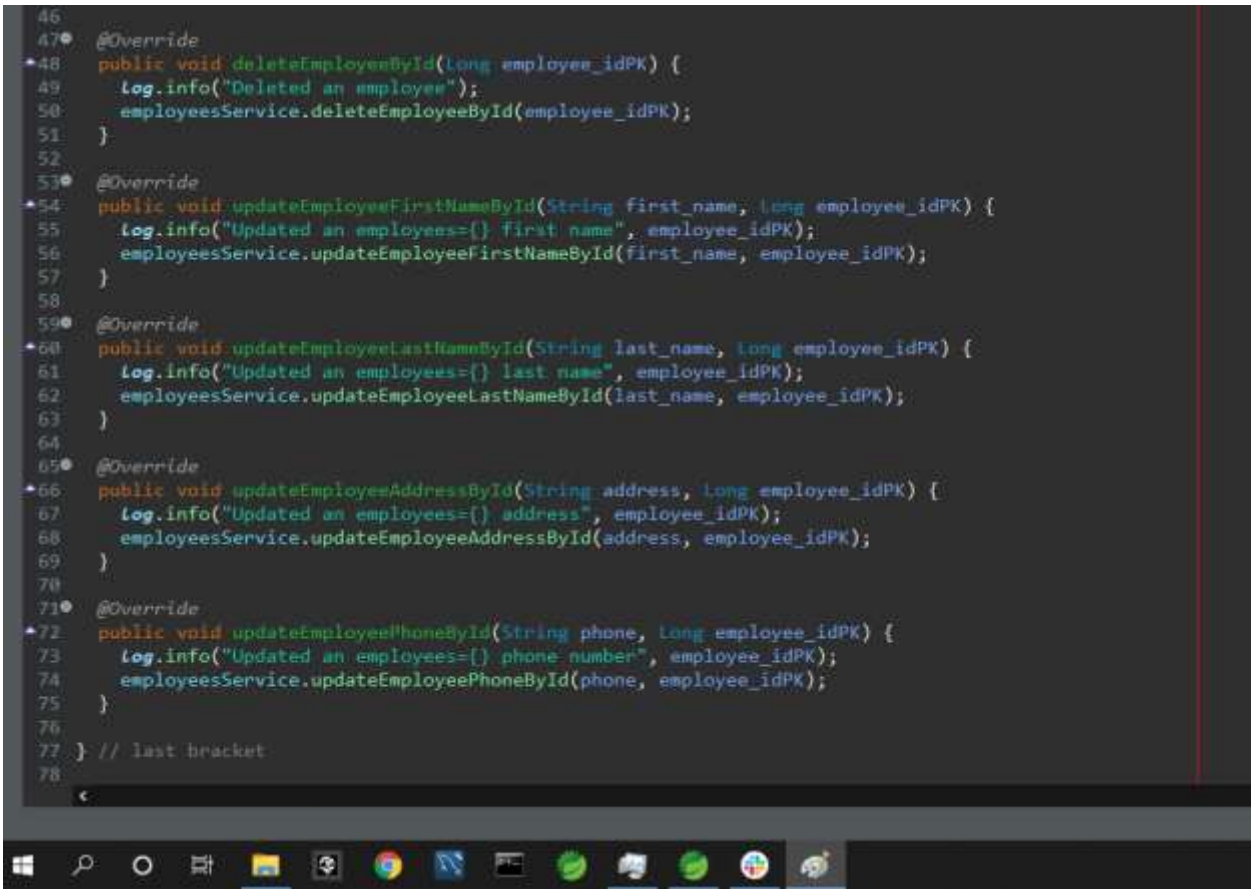


```
File  Edit  Source  Refactor  Navigate  Search  Project  Run  Window  Help

EmployeesController.java  DefaultEmployeesController.java

1  package OTC.StPaul.Final.Project.controller;
2
3  import java.util.List;
4  import org.springframework.http.HttpStatus;
5  import org.springframework.http.ResponseEntity;
6  import org.springframework.validation.annotation.Validated;
7  import org.springframework.web.bind.annotation.DeleteMapping;
8  import org.springframework.web.bind.annotation.GetMapping;
9  import org.springframework.web.bind.annotation.PathVariable;
10 import org.springframework.web.bind.annotation.PostMapping;
11 import org.springframework.web.bind.annotation.PutMapping;
12 import org.springframework.web.bind.annotation.RequestMapping;
13 import org.springframework.web.bind.annotation.RequestParam;
14 import org.springframework.web.bind.annotation.ResponseStatus;
15 import OTC.StPaul.Final.Project.entity.employees;
16 import io.swagger.v3.oas.annotations.OpenAPIDefinition;
17 import io.swagger.v3.oas.annotations.info.Info;
18 import io.swagger.v3.oas.annotations.servers.Server;
19
20 @Validated
21 @RequestMapping("/otc_stpaul_blockbuster")
22 @OpenAPIDefinition(info = @Info(title = "OTC StPaul Blockbuster"), servers = {
23     @Server(url = "http://localhost:8080", description = "Local Server.")})
24 public interface EmployeesController {
25
26   @GetMapping("/employees")
27   ResponseEntity<List<employees>> listAllEmployees();
28
29   @GetMapping("/employeesbystoreid/{store_idFK}")
30   @ResponseStatus(code = HttpStatus.OK)
31   ResponseEntity<List<employees>> retrieveEmployeesByStoreID(
32       @PathVariable Long store_idFK);
33
34   @GetMapping("/employeebyid/{employee_idPK}")
35   @ResponseStatus(code = HttpStatus.OK)
36   ResponseEntity<List<employees>> retrieveSingleEmployeeByEmployeeId(
37       @PathVariable Long employee_idPK);
38
39   @PostMapping("/employees/addemployee")
40   @ResponseStatus(code = HttpStatus.CREATED)
41   void addEmployee(
42       @RequestParam(required = true) String first_name,
43       @RequestParam(required = true) String last_name,
44       @RequestParam(required = true) String address,
45       @RequestParam(required = true) String phone,
46       @RequestParam(required = true) Long store_idFK
47       );
```



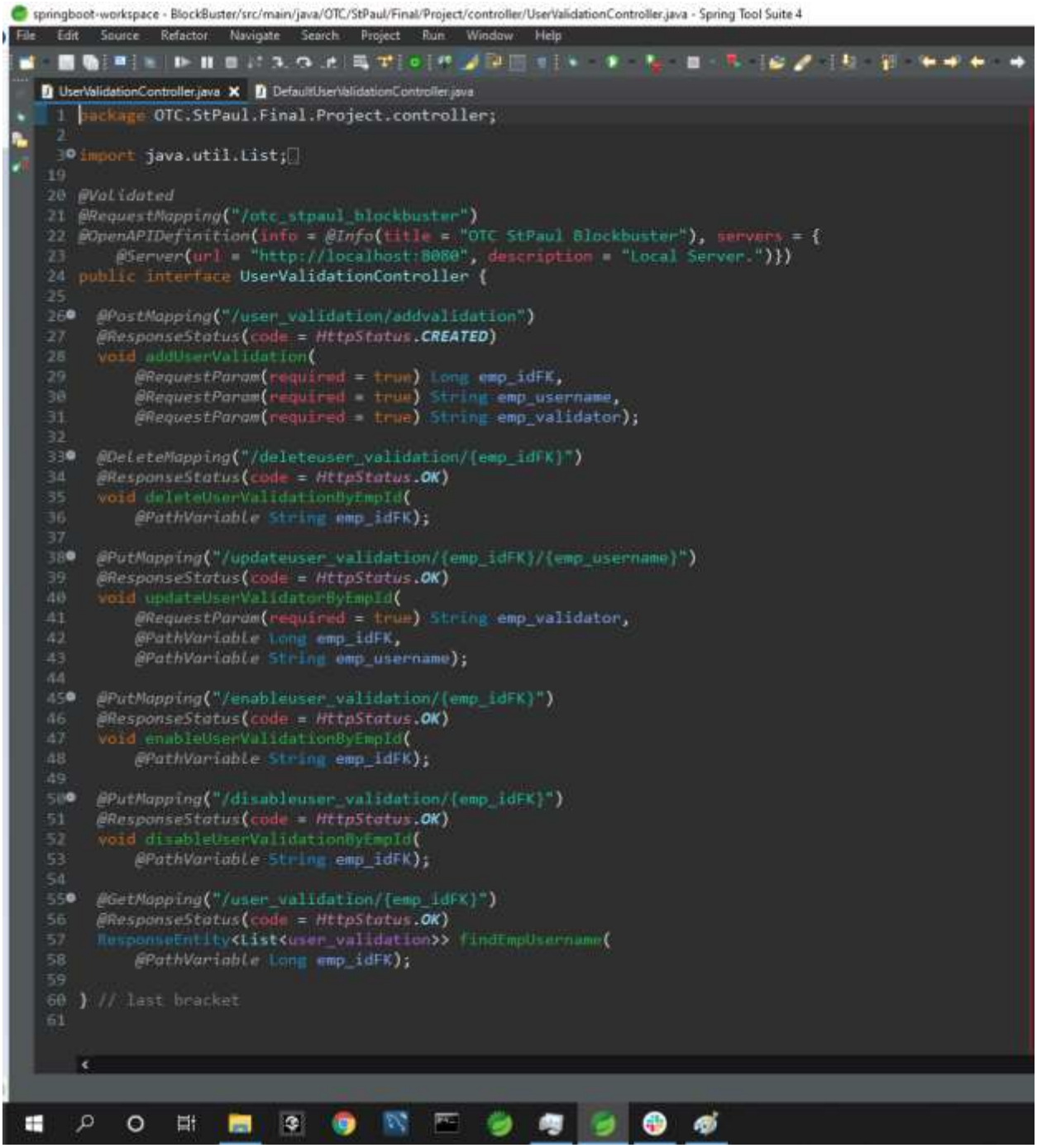
```
48
49  @DeleteMapping("/deleteemployee/{employee_idPK}")
50  @ResponseStatus(code = HttpStatus.OK)
51  void deleteEmployeeById(
52      @PathVariable Long employee_idPK);
53
54  @PutMapping("/updateemployees/{first_name}")
55  @ResponseStatus(code = HttpStatus.OK)
56  void updateEmployeeFirstNameById(
57      @RequestParam(required = true) String first_name,
58      @RequestParam(required = true) Long employee_idPK);
59
60  @PutMapping("/updateemployees/{last_name}")
61  @ResponseStatus(code = HttpStatus.OK)
62  void updateEmployeeLastNameById(
63      @RequestParam(required = true) String last_name,
64      @RequestParam(required = true) Long employee_idPK);
65
66  @PutMapping("/updateemployees/{address}")
67  @ResponseStatus(code = HttpStatus.OK)
68  void updateEmployeeAddressById(
69      @RequestParam(required = true) String address,
70      @RequestParam(required = true) Long employee_idPK);
71
72  @PutMapping("/updateemployees/{phone}")
73  @ResponseStatus(code = HttpStatus.OK)
74  void updateEmployeePhoneById(
75      @RequestParam(required = true) String phone,
76      @RequestParam(required = true) Long employee_idPK);
77
78 } // last bracket
79
```

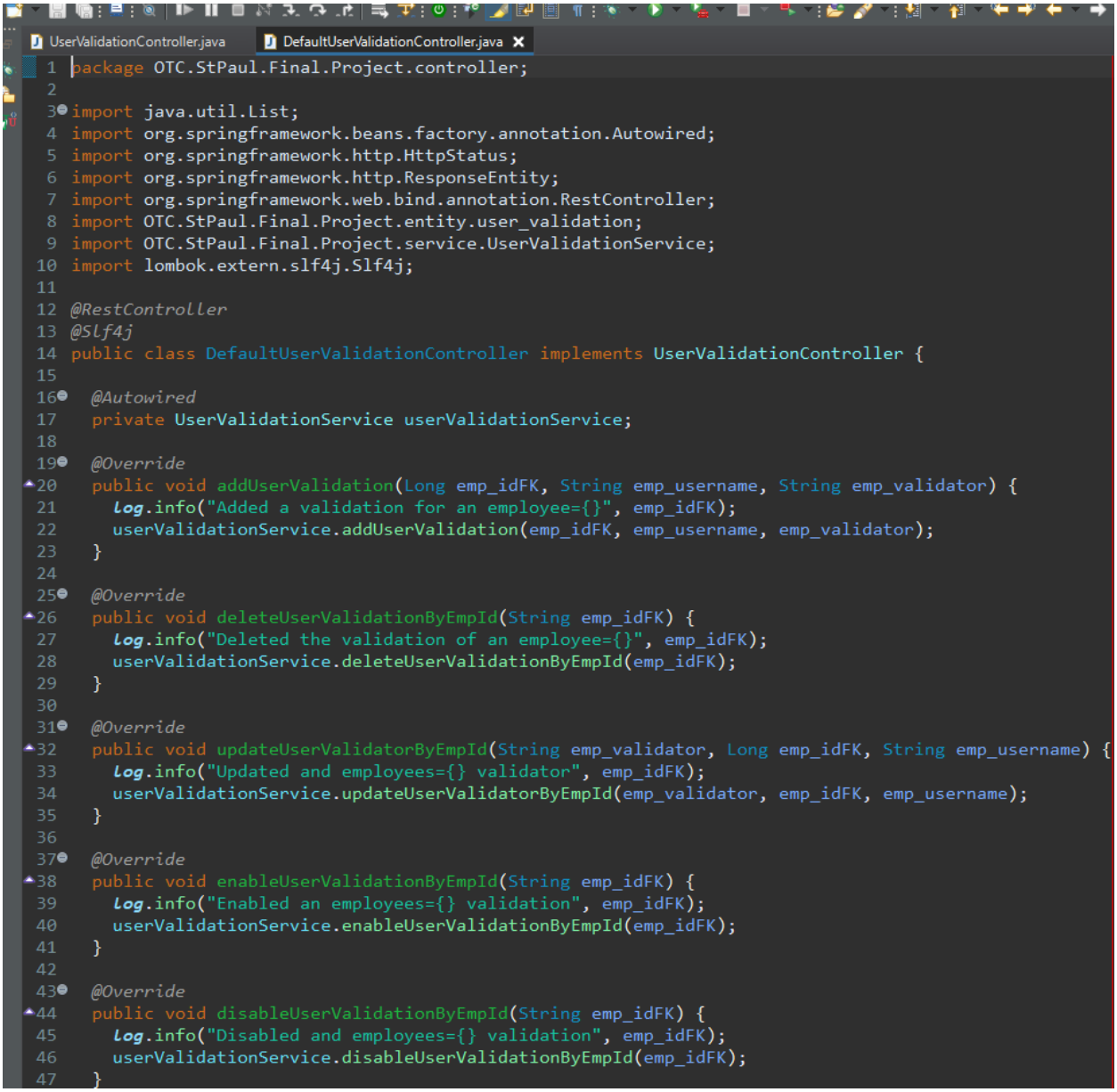

springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/controller/DefaultEmployeesController.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
EmployeesController.java
DefaultEmployeesController.java
package OTC.StPaul.Final.Project.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.RestController;
import OTC.StPaul.Final.Project.entity.employees;
import OTC.StPaul.Final.Project.service.EmployeesService;
import lombok.extern.slf4j.Slf4j;
@RestController
@Slf4j
public class DefaultEmployeesController implements EmployeesController {
@Autowired
private EmployeesService employeesService;
@Override
public ResponseEntity<List<employees>> listAllEmployees() {
log.info("A Request for the complete employees list was made");
List<employees> employees = employeesService.listAllEmployees();
return new ResponseEntity<List<employees>>(employees, HttpStatus.OK);
}
@Override
public ResponseEntity<List<employees>> retrieveEmployeesByStoreID(Long store_idFK) {
log.info("A Request for a stores={} employees was made", store_idFK);
List<employees> storeEmployees = employeesService.retrieveEmployeesByStoreID(store_idFK);
return new ResponseEntity<List<employees>>(storeEmployees, HttpStatus.OK);
}
@Override
public ResponseEntity<List<employees>> retrieveSingleEmployeeByEmployeeId(Long employee_idPK) {
log.info("A Request for a single employees={} information was made", employee_idPK);
List<employees> singleEmployee = employeesService.retrieveSingleEmployeeByEmployeeId(employee_idPK);
return new ResponseEntity<List<employees>>(singleEmployee, HttpStatus.OK);
}
@Override
public void addEmployee(String first_name, String last_name, String address, String phone,
Long store_idFK) {
log.info("Added an employee");
employeesService.addEmployee(first_name, last_name, address, phone, store_idFK);
}

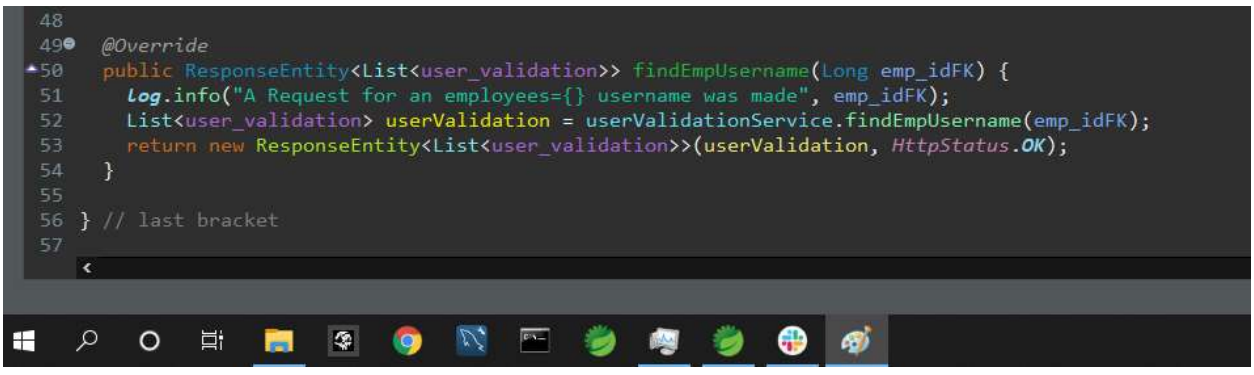


```
46
47  @Override
48  public void deleteEmployeeById(Long employee_idPK) {
49    log.info("Deleted an employee");
50    employeesService.deleteEmployeeById(employee_idPK);
51  }
52
53  @Override
54  public void updateEmployeeFirstNameById(String first_name, Long employee_idPK) {
55    log.info("Updated an employees={} first name", employee_idPK);
56    employeesService.updateEmployeeFirstNameById(first_name, employee_idPK);
57  }
58
59  @Override
60  public void updateEmployeeLastNameById(String last_name, Long employee_idPK) {
61    log.info("Updated an employees={} last name", employee_idPK);
62    employeesService.updateEmployeeLastNameById(last_name, employee_idPK);
63  }
64
65  @Override
66  public void updateEmployeeAddressById(String address, Long employee_idPK) {
67    log.info("Updated an employees={} address", employee_idPK);
68    employeesService.updateEmployeeAddressById(address, employee_idPK);
69  }
70
71  @Override
72  public void updateEmployeePhoneById(String phone, Long employee_idPK) {
73    log.info("Updated an employees={} phone number", employee_idPK);
74    employeesService.updateEmployeePhoneById(phone, employee_idPK);
75  }
76
77 } // last bracket
78
```

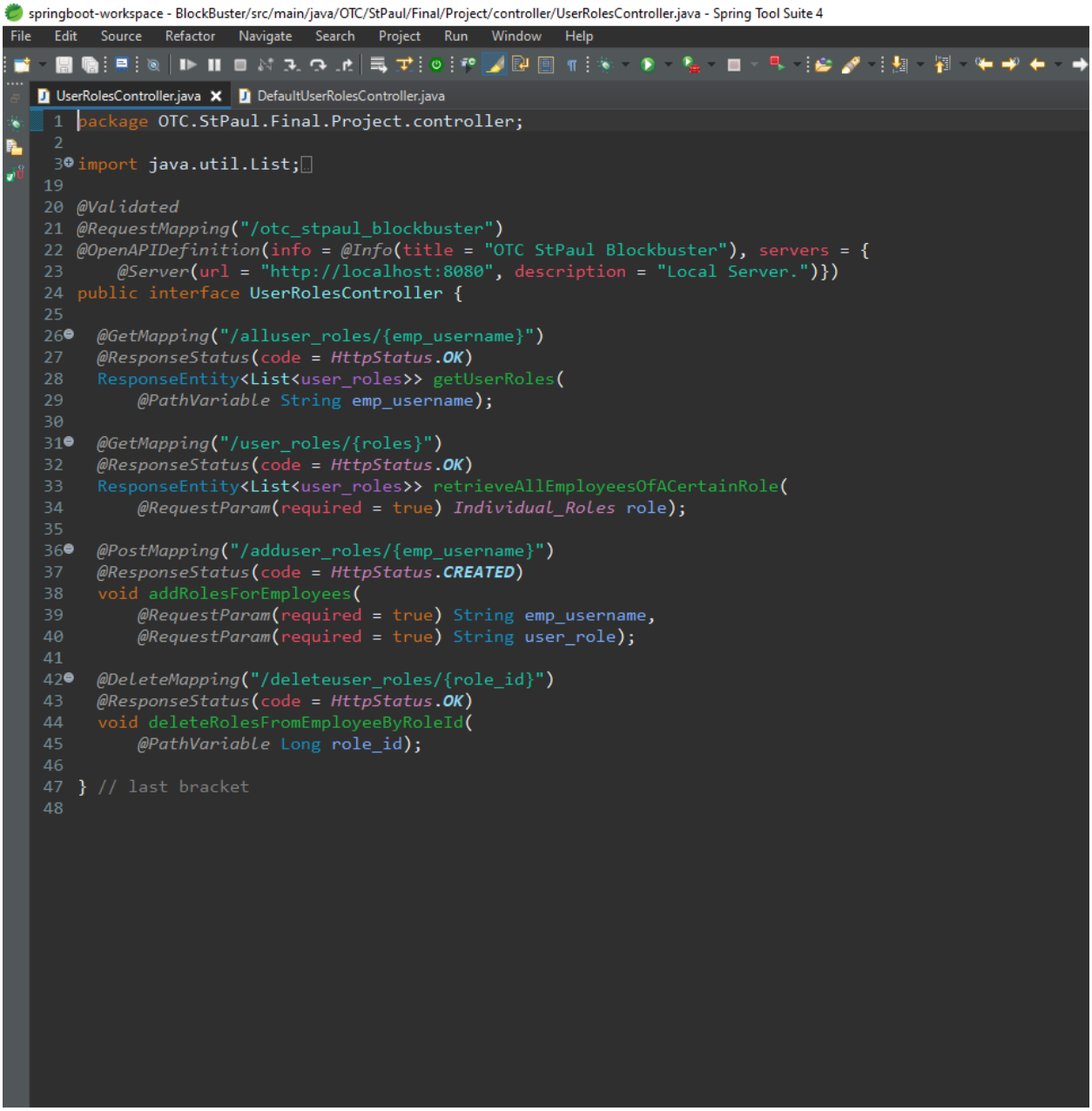
User Validation-

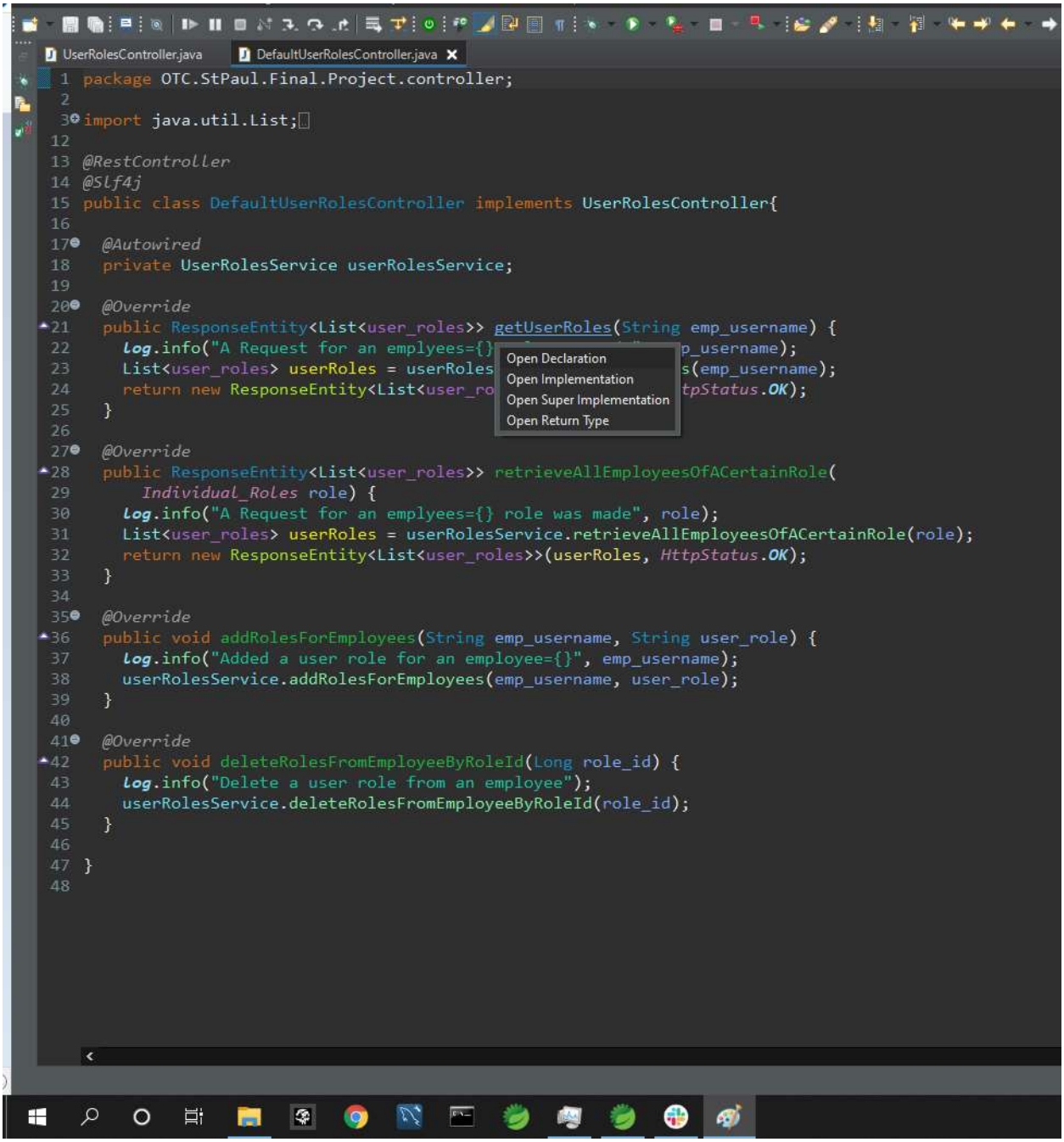



UserValidationController.java
DefaultUserValidationController.java
package OTC.StPaul.Final.Project.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.RestController;
import OTC.StPaul.Final.Project.entity.user_validation;
import OTC.StPaul.Final.Project.service.UserValidationService;
import lombok.extern.slf4j.Slf4j;
@RestController
@Slf4j
public class DefaultUserValidationController implements UserValidationController {
@Autowired
private UserValidationService userValidationService;
@Override
public void addUserValidation(Long emp_idFK, String emp_username, String emp_validator) {
log.info("Added a validation for an employee={}", emp_idFK);
userValidationService.addUserValidation(emp_idFK, emp_username, emp_validator);
}
@Override
public void deleteUserValidationByEmpId(String emp_idFK) {
log.info("Deleted the validation of an employee={}", emp_idFK);
userValidationService.deleteUserValidationByEmpId(emp_idFK);
}
@Override
public void updateUserValidatorByEmpId(String emp_validator, Long emp_idFK, String emp_username) {
log.info("Updated and employees={} validator", emp_idFK);
userValidationService.updateUserValidatorByEmpId(emp_validator, emp_idFK, emp_username);
}
@Override
public void enableUserValidationByEmpId(String emp_idFK) {
log.info("Enabled an employees={} validation", emp_idFK);
userValidationService.enableUserValidationByEmpId(emp_idFK);
}
@Override
public void disableUserValidationByEmpId(String emp_idFK) {
log.info("Disabled and employees={} validation", emp_idFK);
userValidationService.disableUserValidationByEmpId(emp_idFK);
}


@Override
public ResponseEntity<List<user_validation>> findEmpUsername(Long emp_idFK) {
log.info("A Request for an employees={} username was made", emp_idFK);
List<user_validation> userValidation = userValidationService.findEmpUsername(emp_idFK);
return new ResponseEntity<List<user_validation>>(userValidation, HttpStatus.OK);
}
} // last bracket

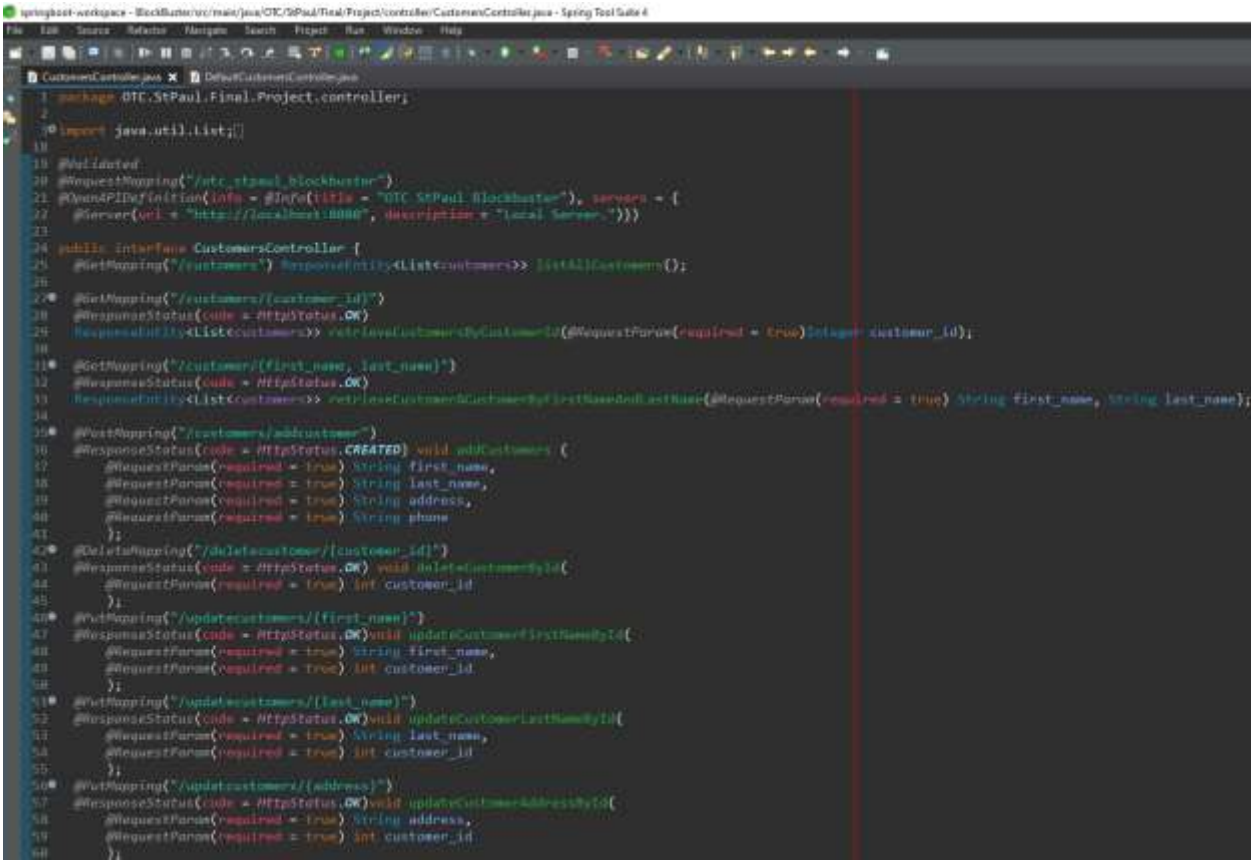
User Roles –


springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/controller/UserRolesController.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
UserRolesController.java
DefaultUserRolesController.java
1 package OTC.StPaul.Final.Project.controller;
2
3 import java.util.List;
19
20 @Validated
21 @RequestMapping("/otc_stpaul_blockbuster")
22 @OpenAPIDefinition(info = @Info(title = "OTC StPaul Blockbuster"), servers = {
23 @Server(url = "http://localhost:8080", description = "Local Server.")})
24 public interface UserRolesController {
25
26 @GetMapping("/alluser_roles/{emp_username}")
27 @ResponseStatus(code = HttpStatus.OK)
28 ResponseEntity<List<user_roles>> getUserRoles(
29 @PathVariable String emp_username);
30
31 @GetMapping("/user_roles/{roles}")
32 @ResponseStatus(code = HttpStatus.OK)
33 ResponseEntity<List<user_roles>> retrieveAllEmployeesOfACertainRole(
34 @RequestParam(required = true) Individual_Roles role);
35
36 @PostMapping("/adduser_roles/{emp_username}")
37 @ResponseStatus(code = HttpStatus.CREATED)
38 void addRolesForEmployees(
39 @RequestParam(required = true) String emp_username,
40 @RequestParam(required = true) String user_role);
41
42 @DeleteMapping("/deleteuser_roles/{role_id}")
43 @ResponseStatus(code = HttpStatus.OK)
44 void deleteRolesFromEmployeeByRoleId(
45 @PathVariable Long role_id);
46
47 } // last bracket
48

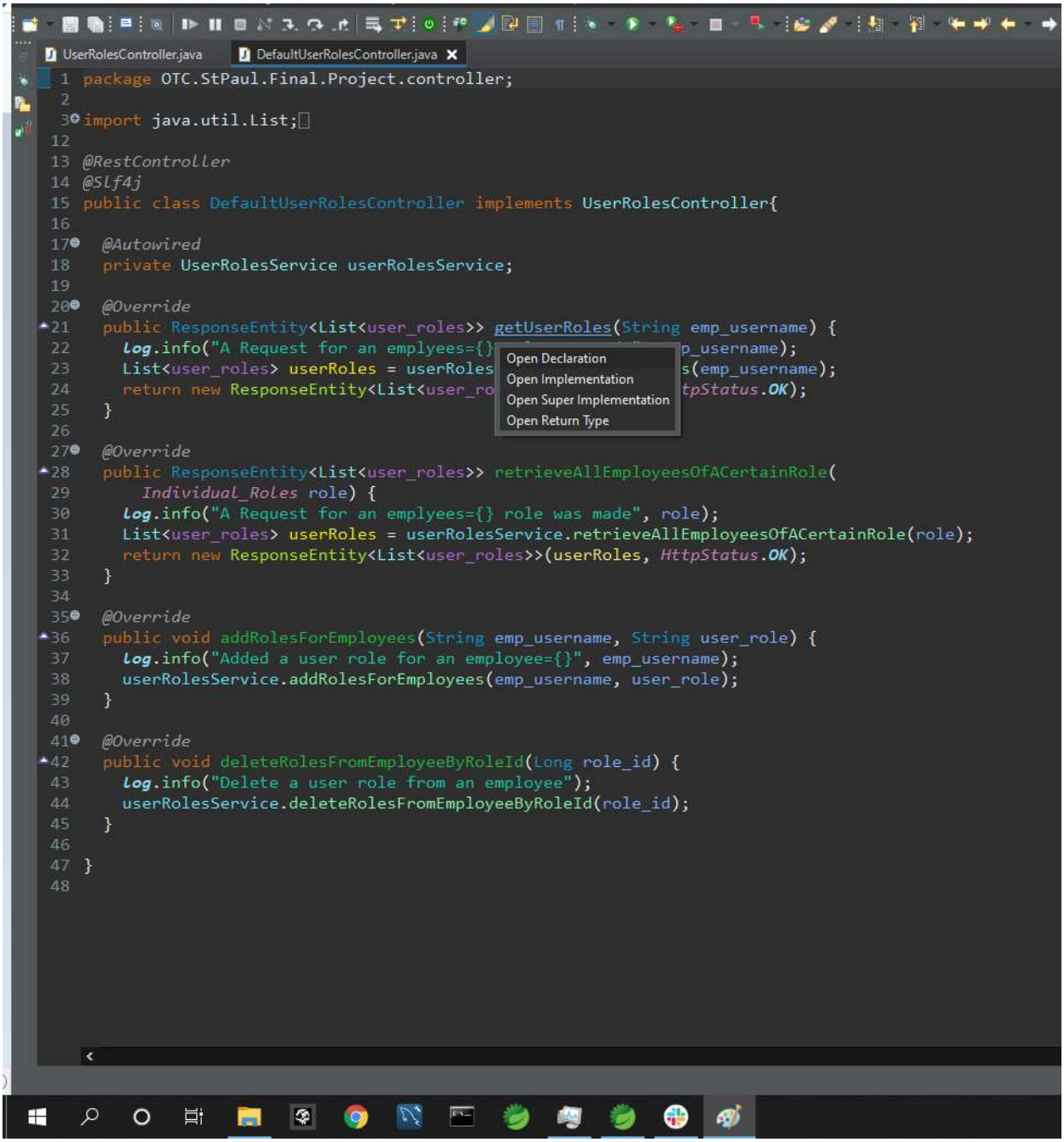



```java
package OTC.StPaul.Final.Project.controller;

import java.util.List;

@RestController
@Slf4j
public class DefaultUserRolesController implements UserRolesController{

  @Autowired
  private UserRolesService userRolesService;

  @Override
  public ResponseEntity<List<user_roles>> getUserRoles(String emp_username) {
    log.info("A Request for an emplyees={}
    List<user_roles> userRoles = userRoles
    return new ResponseEntity<List<user_ro
  }

  @Override
  public ResponseEntity<List<user_roles>> retrieveAllEmployeesOfACertainRole(
      Individual_Roles role) {
    log.info("A Request for an emplyees={} role was made", role);
    List<user_roles> userRoles = userRolesService.retrieveAllEmployeesOfACertainRole(role);
    return new ResponseEntity<List<user_roles>>(userRoles, HttpStatus.OK);
  }

  @Override
  public void addRolesForEmployees(String emp_username, String user_role) {
    log.info("Added a user role for an employee={}", emp_username);
    userRolesService.addRolesForEmployees(emp_username, user_role);
  }

  @Override
  public void deleteRolesFromEmployeeByRoleId(Long role_id) {
    log.info("Delete a user role from an employee");
    userRolesService.deleteRolesFromEmployeeByRoleId(role_id);
  }

}
```

Customers –

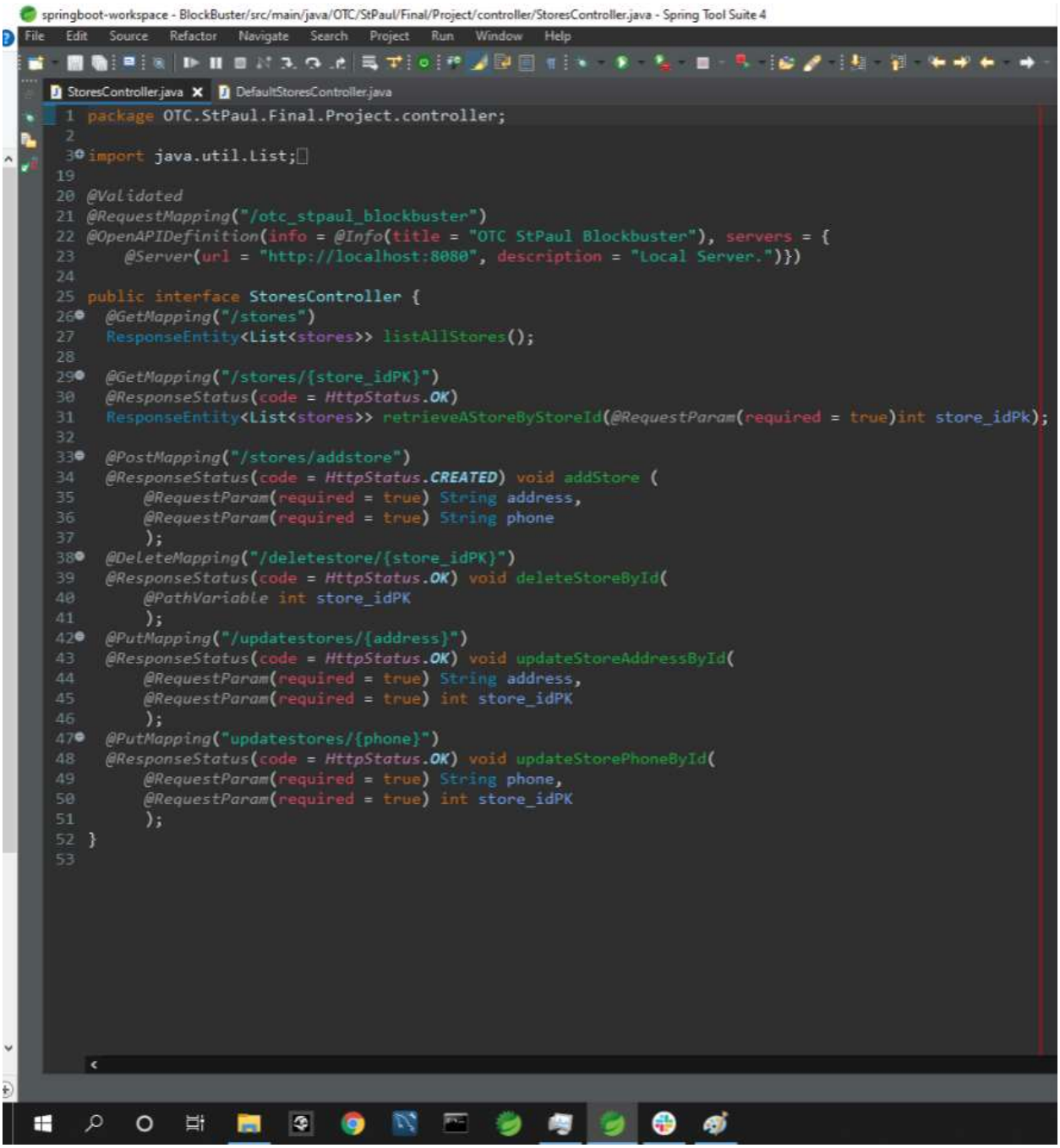


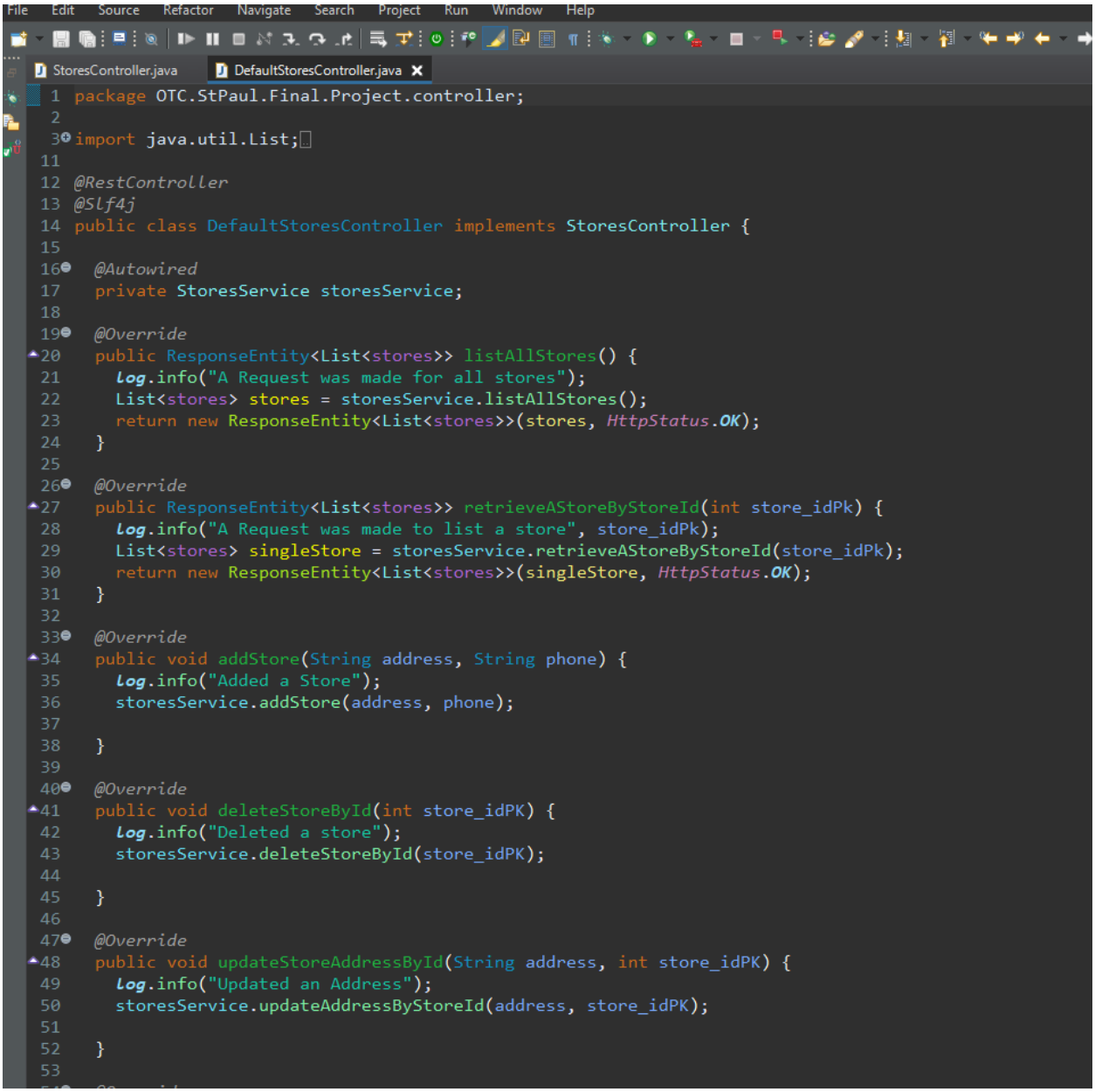
```
61    @PutMapping("/updatecustomers/{phone}")
62    @ResponseStatus(code = HttpStatus.OK)void updateCustomerPhoneById(
63        @RequestParam(required = true) String phone,
64        @RequestParam(required = true) int customer_id);
65
66 } // last bracket
67
```




```java
package OTC.StPaul.Final.Project.controller;

import java.util.List;

@RestController
@Slf4j
public class DefaultUserRolesController implements UserRolesController{

  @Autowired
  private UserRolesService userRolesService;

  @Override
  public ResponseEntity<List<user_roles>> getUserRoles(String emp_username) {
    log.info("A Request for an emplyees={}
    List<user_roles> userRoles = userRoles
    return new ResponseEntity<List<user_ro
  }

  @Override
  public ResponseEntity<List<user_roles>> retrieveAllEmployeesOfACertainRole(
      Individual_Roles role) {
    log.info("A Request for an emplyees={} role was made", role);
    List<user_roles> userRoles = userRolesService.retrieveAllEmployeesOfACertainRole(role);
    return new ResponseEntity<List<user_roles>>(userRoles, HttpStatus.OK);
  }

  @Override
  public void addRolesForEmployees(String emp_username, String user_role) {
    log.info("Added a user role for an employee={}", emp_username);
    userRolesService.addRolesForEmployees(emp_username, user_role);
  }

  @Override
  public void deleteRolesFromEmployeeByRoleId(Long role_id) {
    log.info("Delete a user role from an employee");
    userRolesService.deleteRolesFromEmployeeByRoleId(role_id);
  }

}
```

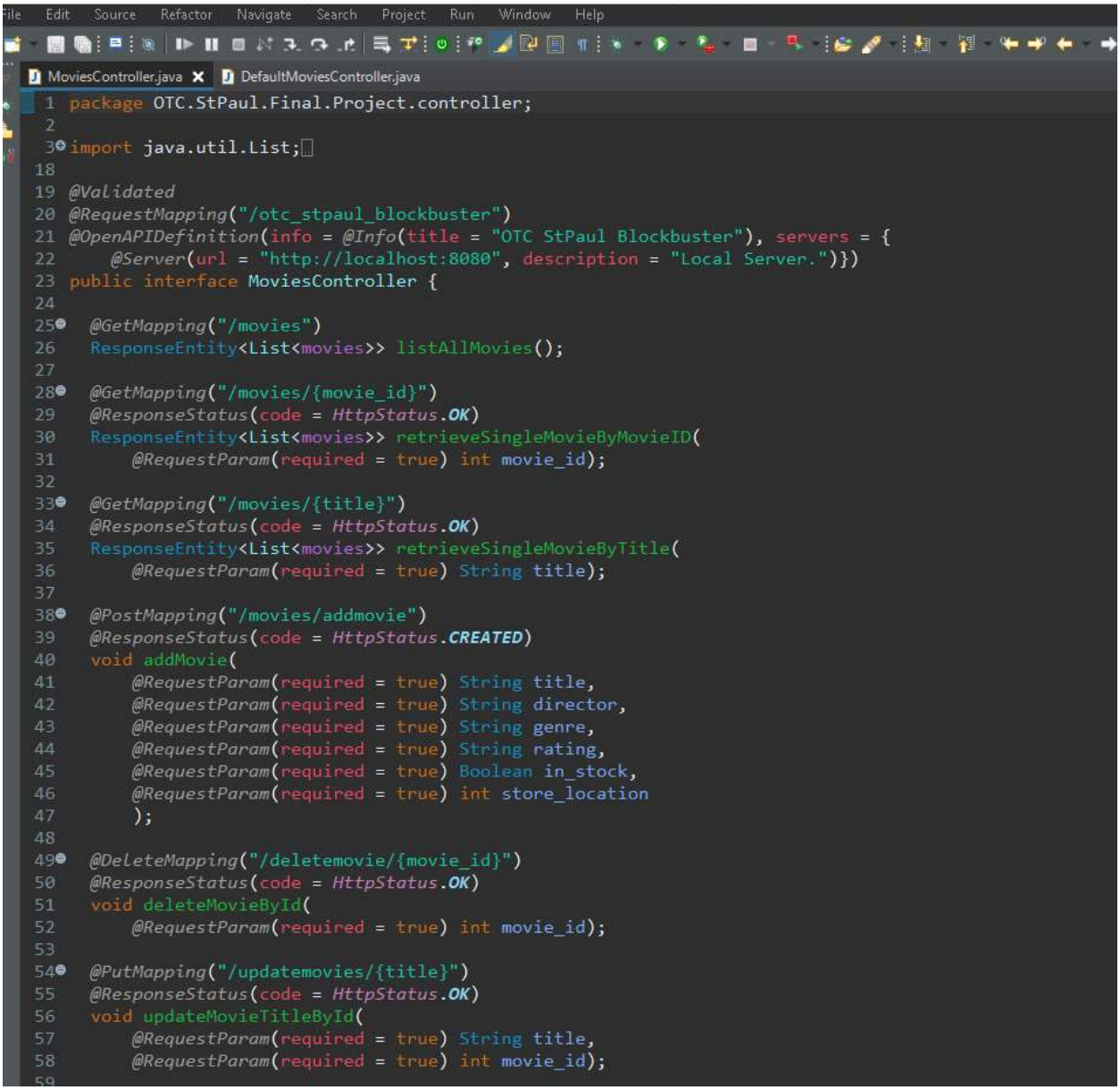
Stores-

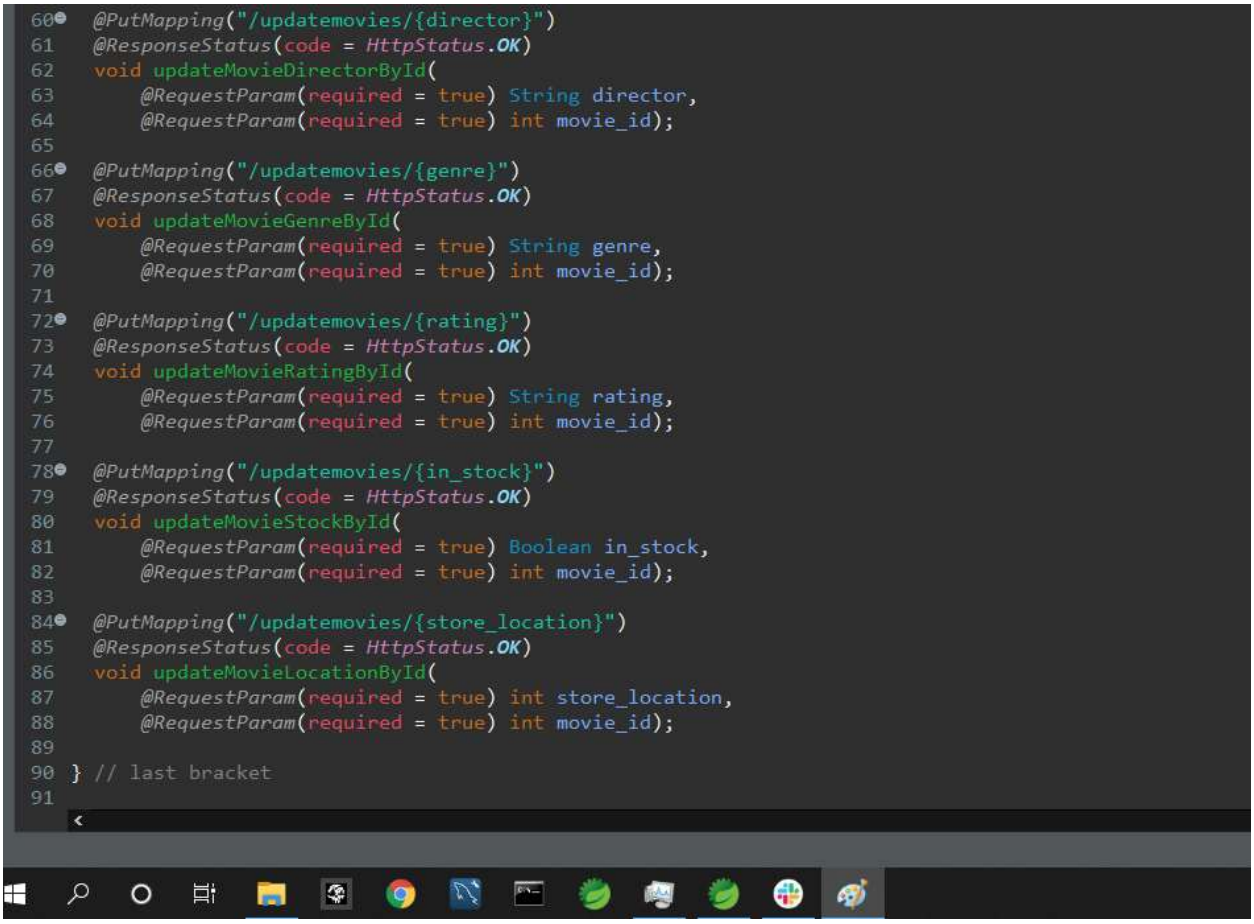

springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/controller/StoresController.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
StoresController.java
DefaultStoresController.java
1 package OTC.StPaul.Final.Project.controller;
2
3 import java.util.List;
19
20 @Validated
21 @RequestMapping("/otc_stpaul_blockbuster")
22 @OpenAPIDefinition(info = @Info(title = "OTC StPaul Blockbuster"), servers = {
23 @Server(url = "http://localhost:8080", description = "Local Server.")})
24
25 public interface StoresController {
26 @GetMapping("/stores")
27 ResponseEntity<List<stores>> listAllStores();
28
29 @GetMapping("/stores/{store_idPK}")
30 @ResponseStatus(code = HttpStatus.OK)
31 ResponseEntity<List<stores>> retrieveAStoreByStoreId(@RequestParam(required = true)int store_idPk);
32
33 @PostMapping("/stores/addstore")
34 @ResponseStatus(code = HttpStatus.CREATED) void addStore (
35 @RequestParam(required = true) String address,
36 @RequestParam(required = true) String phone
37);
38 @DeleteMapping("/deletestore/{store_idPK}")
39 @ResponseStatus(code = HttpStatus.OK) void deleteStoreById(
40 @PathVariable int store_idPK
41);
42 @PutMapping("/updatestores/{address}")
43 @ResponseStatus(code = HttpStatus.OK) void updateStoreAddressById(
44 @RequestParam(required = true) String address,
45 @RequestParam(required = true) int store_idPK
46);
47 @PutMapping("updatestores/{phone}")
48 @ResponseStatus(code = HttpStatus.OK) void updateStorePhoneById(
49 @RequestParam(required = true) String phone,
50 @RequestParam(required = true) int store_idPK
51);
52 }
53



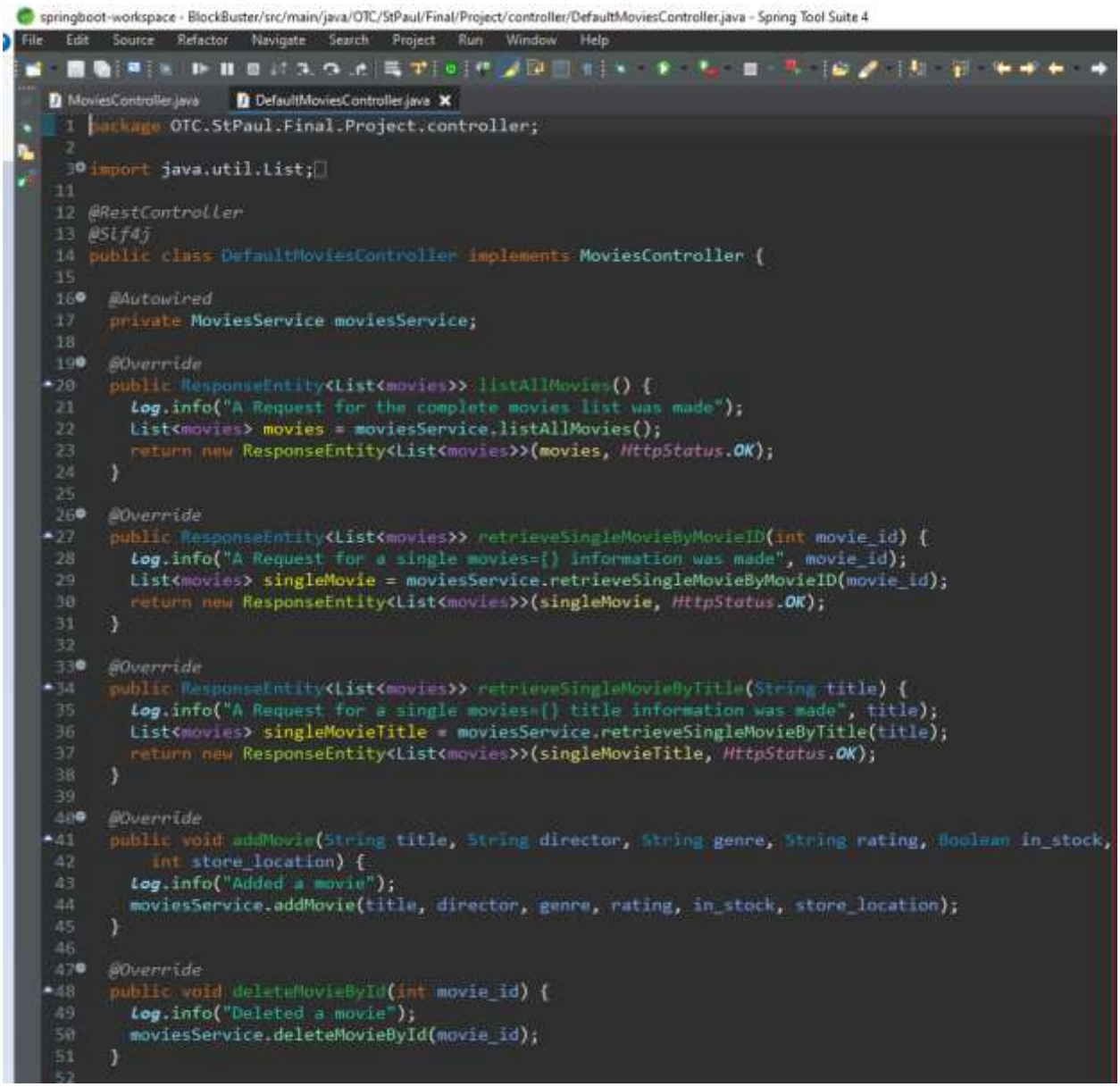
```java
package OTC.StPaul.Final.Project.controller;

import java.util.List;

@RestController
@Slf4j
public class DefaultStoresController implements StoresController {

  @Autowired
  private StoresService storesService;

  @Override
  public ResponseEntity<List<stores>> listAllStores() {
    log.info("A Request was made for all stores");
    List<stores> stores = storesService.listAllStores();
    return new ResponseEntity<List<stores>>(stores, HttpStatus.OK);
  }

  @Override
  public ResponseEntity<List<stores>> retrieveAStoreByStoreId(int store_idPk) {
    log.info("A Request was made to list a store", store_idPk);
    List<stores> singleStore = storesService.retrieveAStoreByStoreId(store_idPk);
    return new ResponseEntity<List<stores>>(singleStore, HttpStatus.OK);
  }

  @Override
  public void addStore(String address, String phone) {
    log.info("Added a Store");
    storesService.addStore(address, phone);

  }

  @Override
  public void deleteStoreById(int store_idPK) {
    log.info("Deleted a store");
    storesService.deleteStoreById(store_idPK);

  }

  @Override
  public void updateStoreAddressById(String address, int store_idPK) {
    log.info("Updated an Address");
    storesService.updateAddressByStoreId(address, store_idPK);

  }

  @Override
  public void updateStorePhoneById(String phone, int store_idPK) {
    log.info("Updated phone");
    storesService.updatePhoneByStroreId(phone, store_idPK);

  }

} // last bracket
```

Movies-


File Edit Source Refactor Navigate Search Project Run Window Help
MoviesController.java
DefaultMoviesController.java
1 package OTC.StPaul.Final.Project.controller;
3 import java.util.List;
18
19 @Validated
20 @RequestMapping("/otc_stpaul_blockbuster")
21 @OpenAPIDefinition(info = @Info(title = "OTC StPaul Blockbuster"), servers = {
22 @Server(url = "http://localhost:8080", description = "Local Server.")})
23 public interface MoviesController {
24
25 @GetMapping("/movies")
26 ResponseEntity<List<movies>> listAllMovies();
27
28 @GetMapping("/movies/{movie_id}")
29 @ResponseStatus(code = HttpStatus.OK)
30 ResponseEntity<List<movies>> retrieveSingleMovieByMovieID(
31 @RequestParam(required = true) int movie_id);
32
33 @GetMapping("/movies/{title}")
34 @ResponseStatus(code = HttpStatus.OK)
35 ResponseEntity<List<movies>> retrieveSingleMovieByTitle(
36 @RequestParam(required = true) String title);
37
38 @PostMapping("/movies/addmovie")
39 @ResponseStatus(code = HttpStatus.CREATED)
40 void addMovie(
41 @RequestParam(required = true) String title,
42 @RequestParam(required = true) String director,
43 @RequestParam(required = true) String genre,
44 @RequestParam(required = true) String rating,
45 @RequestParam(required = true) Boolean in_stock,
46 @RequestParam(required = true) int store_location
47);
48
49 @DeleteMapping("/deletemovie/{movie_id}")
50 @ResponseStatus(code = HttpStatus.OK)
51 void deleteMovieById(
52 @RequestParam(required = true) int movie_id);
53
54 @PutMapping("/updatemovies/{title}")
55 @ResponseStatus(code = HttpStatus.OK)
56 void updateMovieTitleById(
57 @RequestParam(required = true) String title,
58 @RequestParam(required = true) int movie_id);

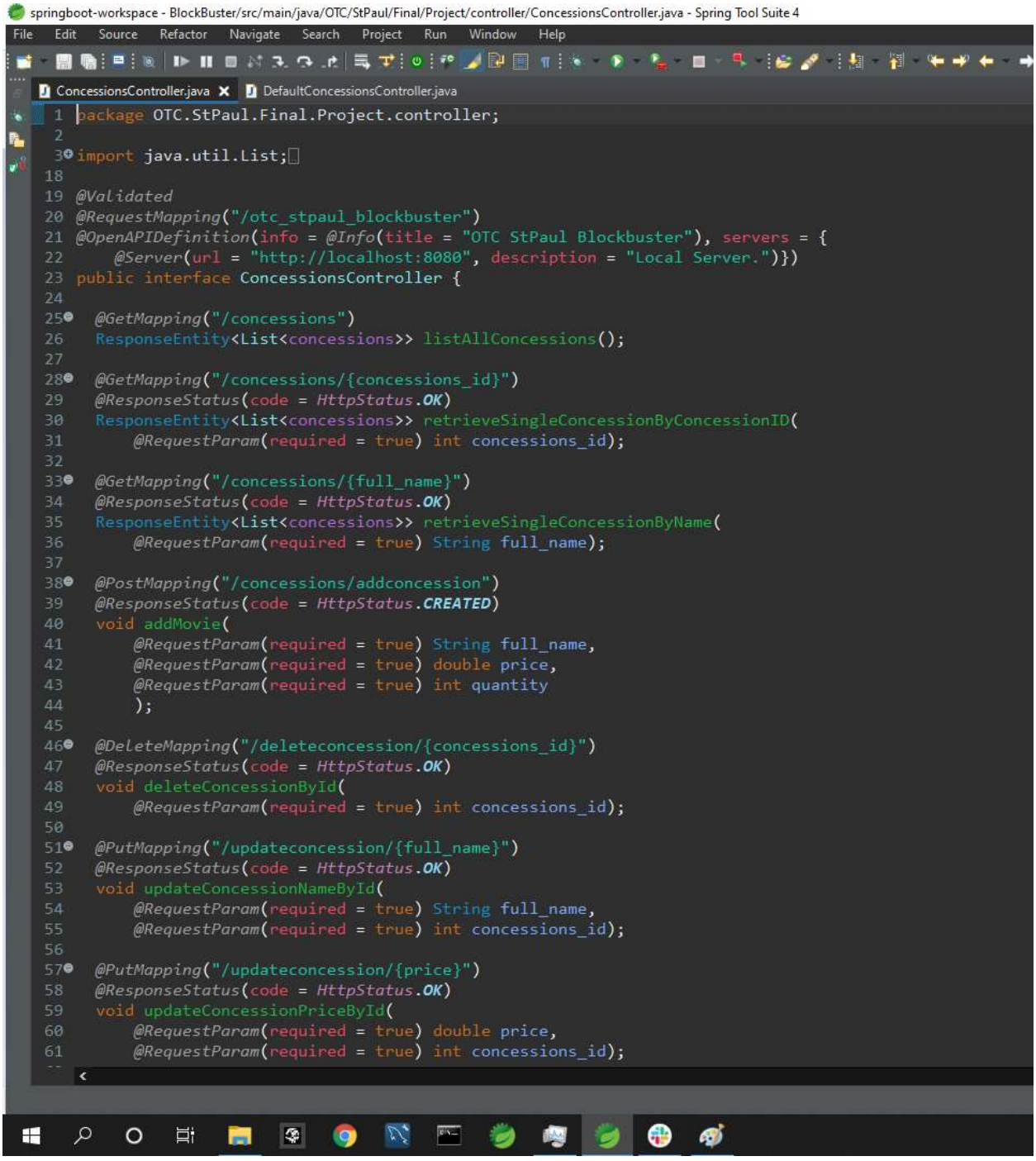


```
@PutMapping("/updatemovies/{director}")
@ResponseStatus(code = HttpStatus.OK)
void updateMovieDirectorById(
    @RequestParam(required = true) String director,
    @RequestParam(required = true) int movie_id);

@PutMapping("/updatemovies/{genre}")
@ResponseStatus(code = HttpStatus.OK)
void updateMovieGenreById(
    @RequestParam(required = true) String genre,
    @RequestParam(required = true) int movie_id);

@PutMapping("/updatemovies/{rating}")
@ResponseStatus(code = HttpStatus.OK)
void updateMovieRatingById(
    @RequestParam(required = true) String rating,
    @RequestParam(required = true) int movie_id);

@PutMapping("/updatemovies/{in_stock}")
@ResponseStatus(code = HttpStatus.OK)
void updateMovieStockById(
    @RequestParam(required = true) Boolean in_stock,
    @RequestParam(required = true) int movie_id);

@PutMapping("/updatemovies/{store_location}")
@ResponseStatus(code = HttpStatus.OK)
void updateMovieLocationById(
    @RequestParam(required = true) int store_location,
    @RequestParam(required = true) int movie_id);

} // last bracket
```


springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/controller/DefaultMoviesController.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
MoviesController.java
DefaultMoviesController.java
1 package OTC.StPaul.Final.Project.controller;
2
3 import java.util.List;
11
12 @RestController
13 @Slf4j
14 public class DefaultMoviesController implements MoviesController {
15
16 @Autowired
17 private MoviesService moviesService;
18
19 @Override
20 public ResponseEntity<List<movies>> listAllMovies() {
21 Log.info("A Request for the complete movies list was made");
22 List<movies> movies = moviesService.listAllMovies();
23 return new ResponseEntity<List<movies>>(movies, HttpStatus.OK);
24 }
25
26 @Override
27 public ResponseEntity<List<movies>> retrieveSingleMovieByMovieID(int movie_id) {
28 Log.info("A Request for a single movies={} information was made", movie_id);
29 List<movies> singleMovie = moviesService.retrieveSingleMovieByMovieID(movie_id);
30 return new ResponseEntity<List<movies>>(singleMovie, HttpStatus.OK);
31 }
32
33 @Override
34 public ResponseEntity<List<movies>> retrieveSingleMovieByTitle(String title) {
35 Log.info("A Request for a single movies={} title information was made", title);
36 List<movies> singleMovieTitle = moviesService.retrieveSingleMovieByTitle(title);
37 return new ResponseEntity<List<movies>>(singleMovieTitle, HttpStatus.OK);
38 }
39
40 @Override
41 public void addMovie(String title, String director, String genre, String rating, Boolean in_stock,
42 int store_location) {
43 Log.info("Added a movie");
44 moviesService.addMovie(title, director, genre, rating, in_stock, store_location);
45 }
46
47 @Override
48 public void deleteMovieById(int movie_id) {
49 Log.info("Deleted a movie");
50 moviesService.deleteMovieById(movie_id);
51 }
52

```
53    @Override
54    public void updateMovieTitleById(String title, int movie_id) {
55     log.info("Update a movies={} title", movie_id);
56     moviesService.updateMovieTitleById(title, movie_id);
57    }
58
59    @Override
60    public void updateMovieDirectorById(String director, int movie_id) {
61      log.info("Update a movies={} director", movie_id);
62      moviesService.updateMovieDirectorById(director, movie_id);
63    }
64
65    @Override
66    public void updateMovieGenreById(String genre, int movie_id) {
67      log.info("Update a movies={} genre", movie_id);
68      moviesService.updateMovieGenreById(genre, movie_id);
69    }
70
71    @Override
72    public void updateMovieRatingById(String rating, int movie_id) {
73      log.info("Update a movies={} rating", movie_id);
74      moviesService.updateMovieRatingById(rating, movie_id);
75    }
76
77    @Override
78    public void updateMovieStockById(Boolean in_stock, int movie_id) {
79      log.info("Update a movies={} stock", movie_id);
80      moviesService.updateMovieStockById(in_stock, movie_id);
81    }
82
83    @Override
84    public void updateMovieLocationById(int store_location, int movie_id) {
85      log.info("Update a movies={} store location number", movie_id);
86      moviesService.updateMovieLocationById(store_location, movie_id);
87    }
88
89  } // last bracket
90
```

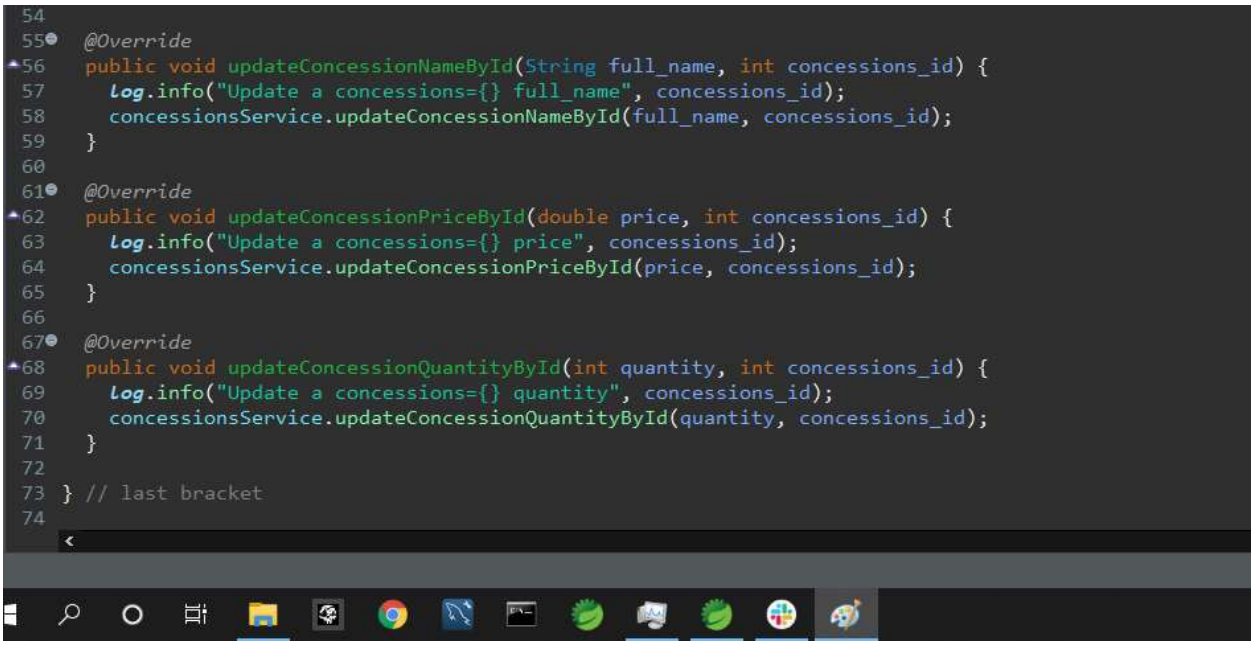
Concessions-



```
62
63  @PutMapping("/updateconcession/{quantity}")
64  @ResponseStatus(code = HttpStatus.OK)
65  void updateConcessionQuantityById(
66      @RequestParam(required = true) int quantity,
67      @RequestParam(required = true) int concessions_id);
68
69 } // last bracket
70
```

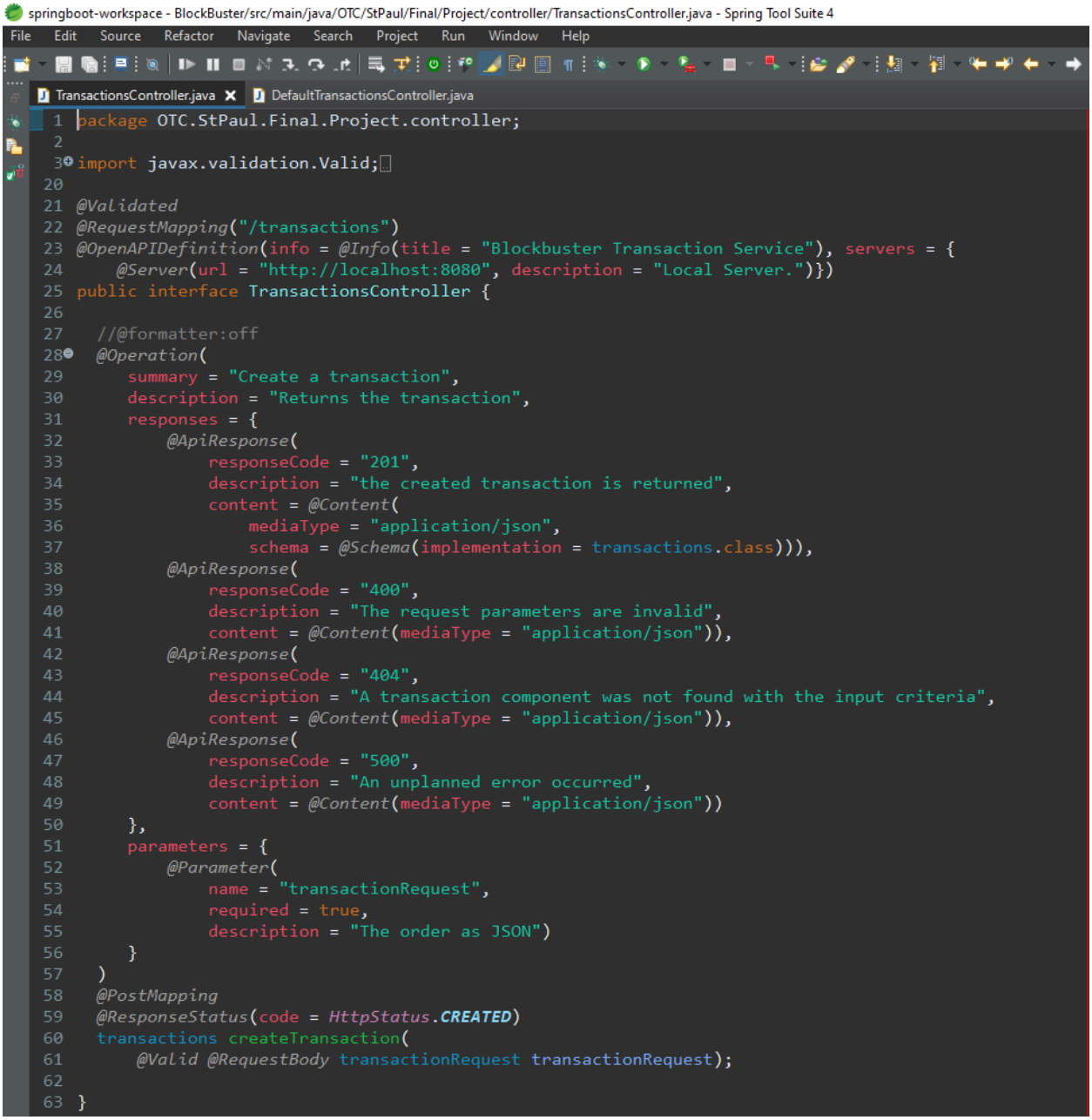
ConcessionsController.java | DefaultConcessionsController.java

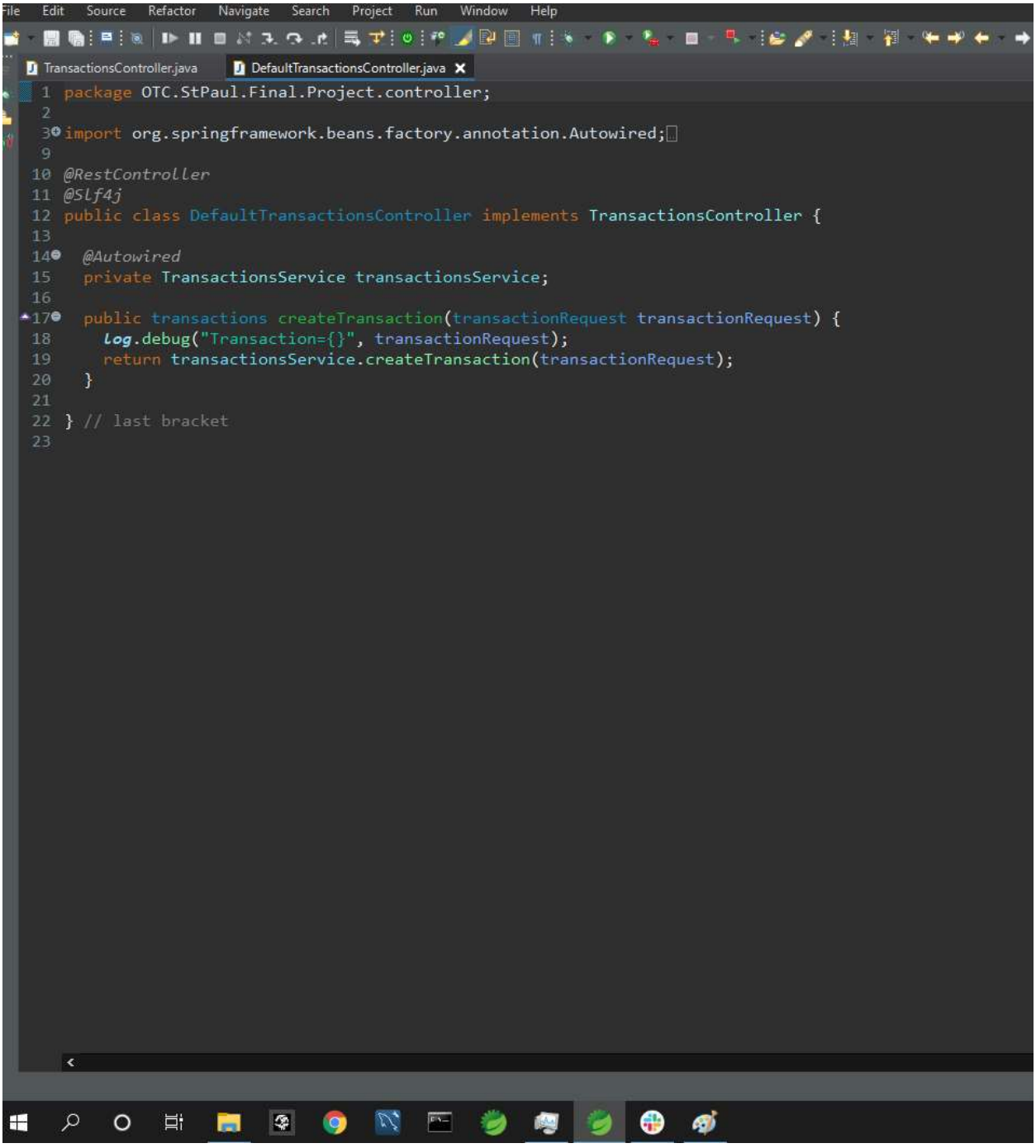
```
1  package OTC.StPaul.Final.Project.controller;
2
3  import java.util.List;
11
12 @RestController
13 @Slf4j
14 public class DefaultConcessionsController implements ConcessionsController {
15
16   @Autowired
17   private ConcessionsService concessionsService;
18
19   @Override
20   public ResponseEntity<List<concessions>> listAllConcessions() {
21     log.info("A Request for the complete concessions list was made");
22     List<concessions> concessions = concessionsService.listAllConcessions();
23     return new ResponseEntity<List<concessions>>(concessions, HttpStatus.OK);
24   }
25
26   @Override
27   public ResponseEntity<List<concessions>> retrieveSingleConcessionByConcessionID(
28       int concessions_id) {
29     log.info("A Request for a single concessions={} information was made", concessions_id);
30     List<concessions> singleConcession = concessionsService
31         .retrieveSingleConcessionByConcessionID(concessions_id);
32     return new ResponseEntity<List<concessions>>(singleConcession, HttpStatus.OK);
33   }
34
35   @Override
36   public ResponseEntity<List<concessions>> retrieveSingleConcessionByName(String full_name) {
37     log.info("A Request for a single concessions={} name information was made", full_name);
38     List<concessions> singleConcessionName = concessionsService
39         .retrieveSingleConcessionByName(full_name);
40     return new ResponseEntity<List<concessions>>(singleConcessionName, HttpStatus.OK);
41   }
42
43   @Override
44   public void addMovie(String full_name, double price, int quantity) {
45     log.info("Added a concession");
46     concessionsService.addConcession(full_name, price, quantity);
47   }
48
49   @Override
50   public void deleteConcessionById(int concessions_id) {
51     log.info("Deleted a concession");
52     concessionsService.deleteConcessionById(concessions_id);
53   }
54
```



```
54
55    @Override
56    public void updateConcessionNameById(String full_name, int concessions_id) {
57      log.info("Update a concessions={} full_name", concessions_id);
58      concessionsService.updateConcessionNameById(full_name, concessions_id);
59    }
60
61    @Override
62    public void updateConcessionPriceById(double price, int concessions_id) {
63      log.info("Update a concessions={} price", concessions_id);
64      concessionsService.updateConcessionPriceById(price, concessions_id);
65    }
66
67    @Override
68    public void updateConcessionQuantityById(int quantity, int concessions_id) {
69      log.info("Update a concessions={} quantity", concessions_id);
70      concessionsService.updateConcessionQuantityById(quantity, concessions_id);
71    }
72
73  } // last bracket
74
```

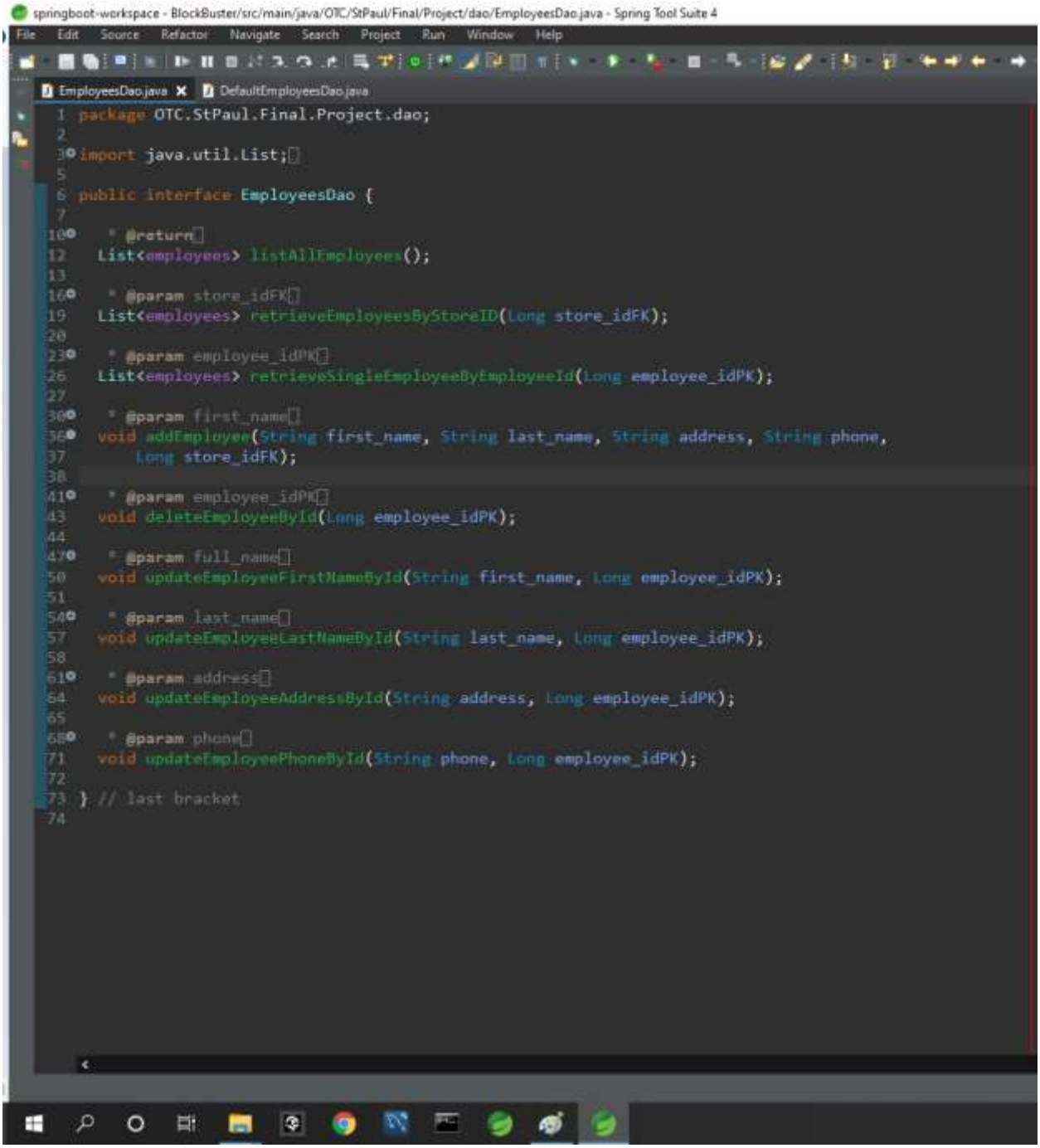
Transactions-

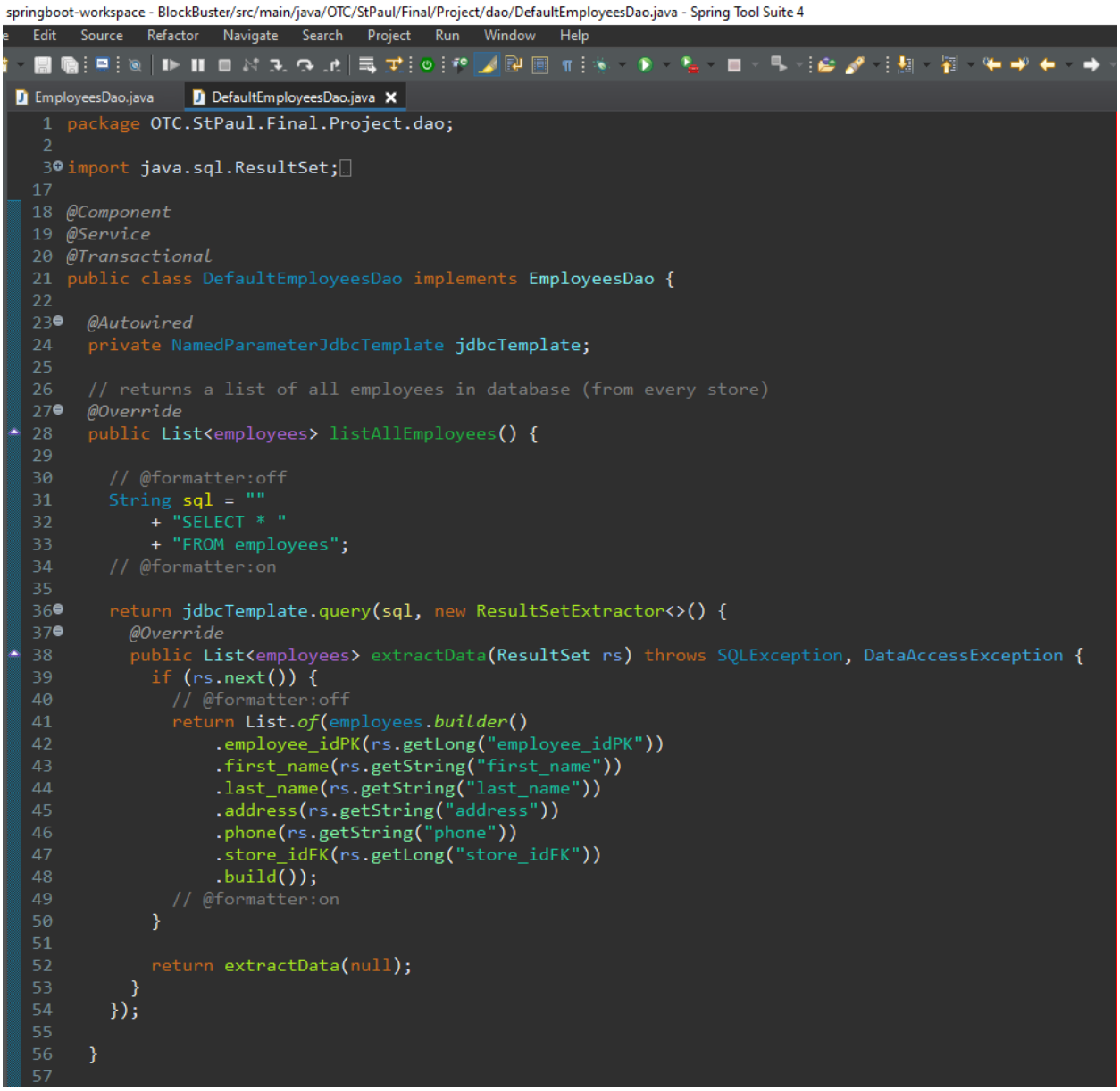



File Edit Source Refactor Navigate Search Project Run Window Help
TransactionsController.java
DefaultTransactionsController.java
1 package OTC.StPaul.Final.Project.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
9
10 @RestController
11 @Slf4j
12 public class DefaultTransactionsController implements TransactionsController {
13
14 @Autowired
15 private TransactionsService transactionsService;
16
17 public transactions createTransaction(transactionRequest transactionRequest) {
18 log.debug("Transaction={}", transactionRequest);
19 return transactionsService.createTransaction(transactionRequest);
20 }
21
22 } // last bracket
23

Dao:

Employees-

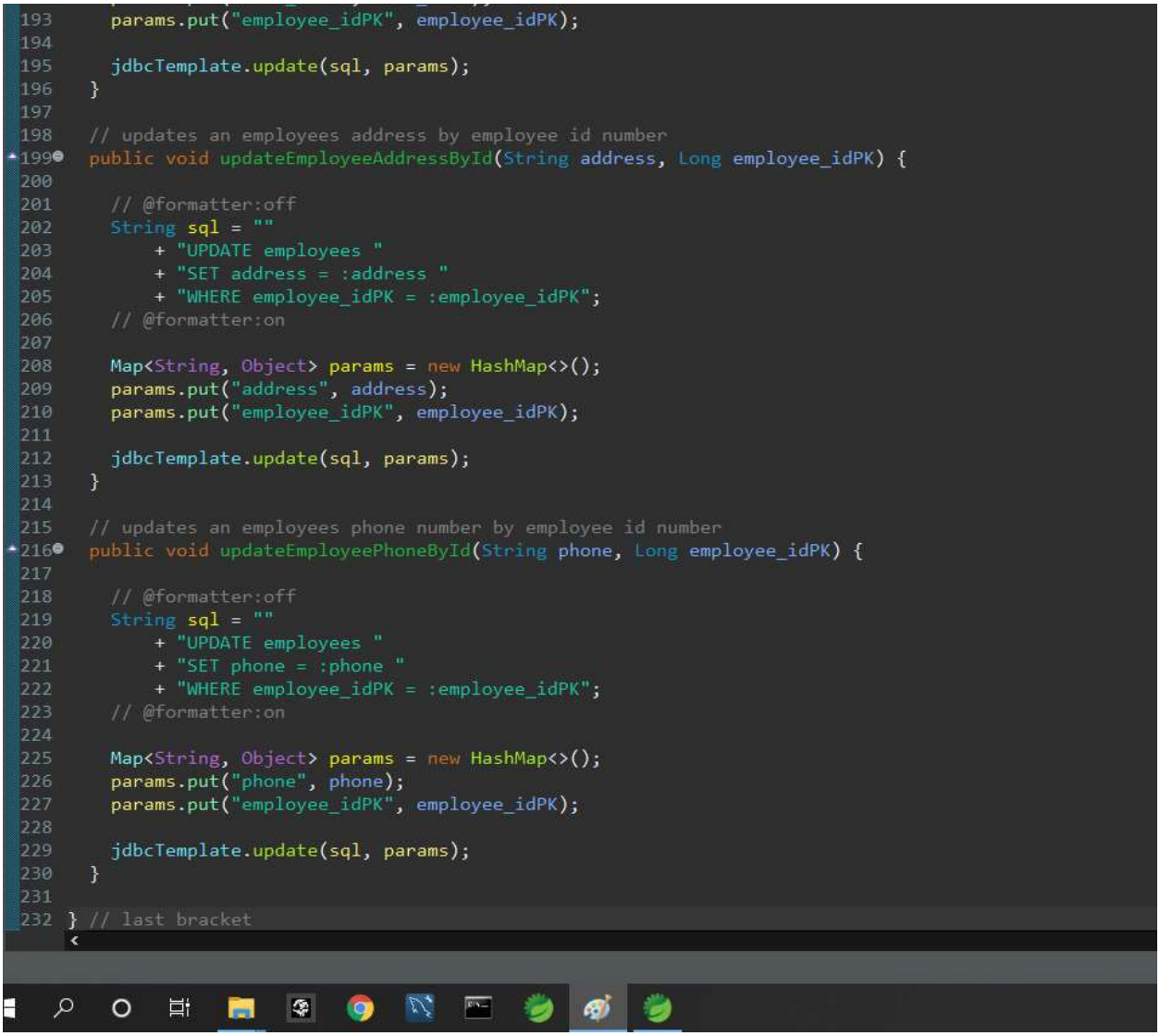



springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/dao/DefaultEmployeesDao.java - Spring Tool Suite 4
Edit Source Refactor Navigate Search Project Run Window Help
EmployeesDao.java
DefaultEmployeesDao.java
1 package OTC.StPaul.Final.Project.dao;
2
3 import java.sql.ResultSet;
17
18 @Component
19 @Service
20 @Transactional
21 public class DefaultEmployeesDao implements EmployeesDao {
22
23 @Autowired
24 private NamedParameterJdbcTemplate jdbcTemplate;
25
26 // returns a list of all employees in database (from every store)
27 @Override
28 public List<employees> listAllEmployees() {
29
30 // @formatter:off
31 String sql = ""
32 + "SELECT * "
33 + "FROM employees";
34 // @formatter:on
35
36 return jdbcTemplate.query(sql, new ResultSetExtractor<>() {
37 @Override
38 public List<employees> extractData(ResultSet rs) throws SQLException, DataAccessException {
39 if (rs.next()) {
40 // @formatter:off
41 return List.of(employees.builder()
42 .employee_idPK(rs.getLong("employee_idPK"))
43 .first_name(rs.getString("first_name"))
44 .last_name(rs.getString("last_name"))
45 .address(rs.getString("address"))
46 .phone(rs.getString("phone"))
47 .store_idFK(rs.getLong("store_idFK"))
48 .build());
49 // @formatter:on
50 }
51
52 return extractData(null);
53 }
54 });
55
56 }
57

```java
// returns all employees for a specific store using the store id number
@Override
public List<employees> retrieveEmployeesByStoreID(Long store_idFK) {

  // @formatter:off
  String sql = ""
      + "SELECT * "
      + "FROM employees "
      + "WHERE store_idFK = :store_idFK";
  // @formatter:on

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

  return jdbcTemplate.query(sql, params, new ResultSetExtractor<>() {
    @Override
    public List<employees> extractData(ResultSet rs) throws SQLException, DataAccessException {
      if (rs.next()) {
        // @formatter:off
        return List.of(employees.builder()
            .employee_idPK(rs.getLong("employee_idPK"))
            .first_name(rs.getString("first_name"))
            .last_name(rs.getString("last_name"))
            .address(rs.getString("address"))
            .phone(rs.getString("phone"))
            .store_idFK(rs.getLong("store_idFK"))
            .build());
        // @formatter:on
      }

      return extractData(null);
    }
  });
}

// returns a single employee by by employee id number
@Override
public List<employees> retrieveSingleEmployeeByEmployeeId(Long employee_idPK) {

  // @formatter:off
  String sql = ""
      + "SELECT * "
      + "FROM employees "
      + "WHERE employee_idPK = :employee_idPK";
  // @formatter:on
```

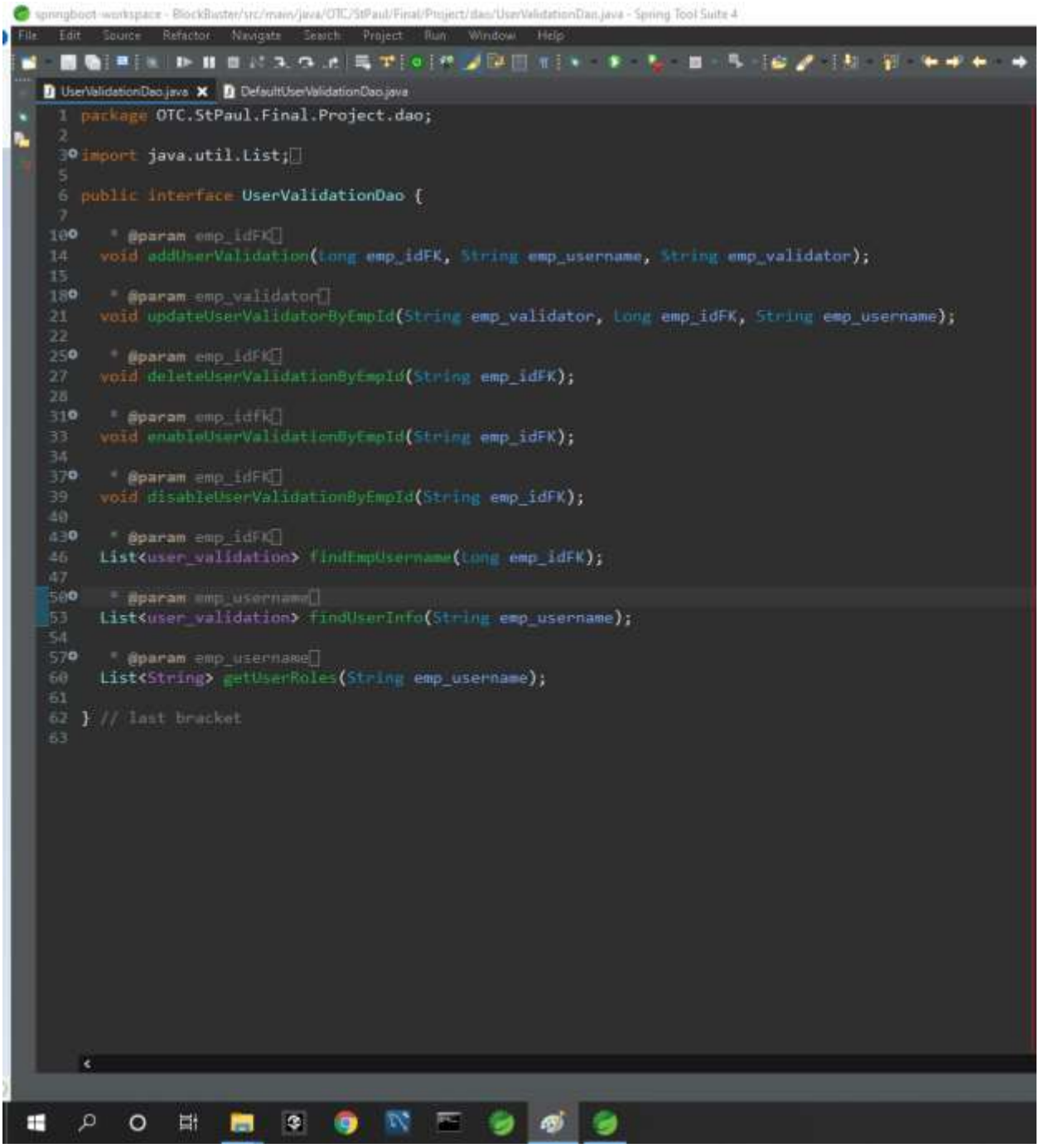
```
103
104     Map<String, Object> params = new HashMap<>();
105     params.put("employee_idPK", employee_idPK);
106
107     return jdbcTemplate.query(sql, params, new RowMapper<>() {
108
109       @Override
110       public employees mapRow(ResultSet rs, int rowNum) throws SQLException {
111
112         // @formatter:off
113           return employees.builder()
114               .employee_idPK(rs.getLong("employee_idPK"))
115               .first_name(rs.getString("first_name"))
116               .last_name(rs.getString("last_name"))
117               .address(rs.getString("address"))
118               .phone(rs.getString("phone"))
119               .store_idFK(rs.getLong("store_idFK"))
120               .build();
121           // @formatter:on
122       }
123     });
124   }
125
126   // adds an employee to the database
127   @Override
128   public void addEmployee(String first_name, String last_name, String address, String phone,
129       Long store_idFK) {
130     // @formatter:off
131     String sql = ""
132         + "INSERT INTO employees ("
133         + "first_name, last_name, address, phone, store_idFK"
134         + ") VALUES ("
135         + ":first_name, :last_name, :address, :phone, :store_idFK"
136         + ")";
137     // @formatter:on
138     Map<String, Object> params = new HashMap<>();
139
140     params.put("first_name", first_name);
141     params.put("last_name", last_name);
142     params.put("address", address);
143     params.put("phone", phone);
144     params.put("store_idFK", store_idFK);
145
146     jdbcTemplate.update(sql, params);
147   }
```

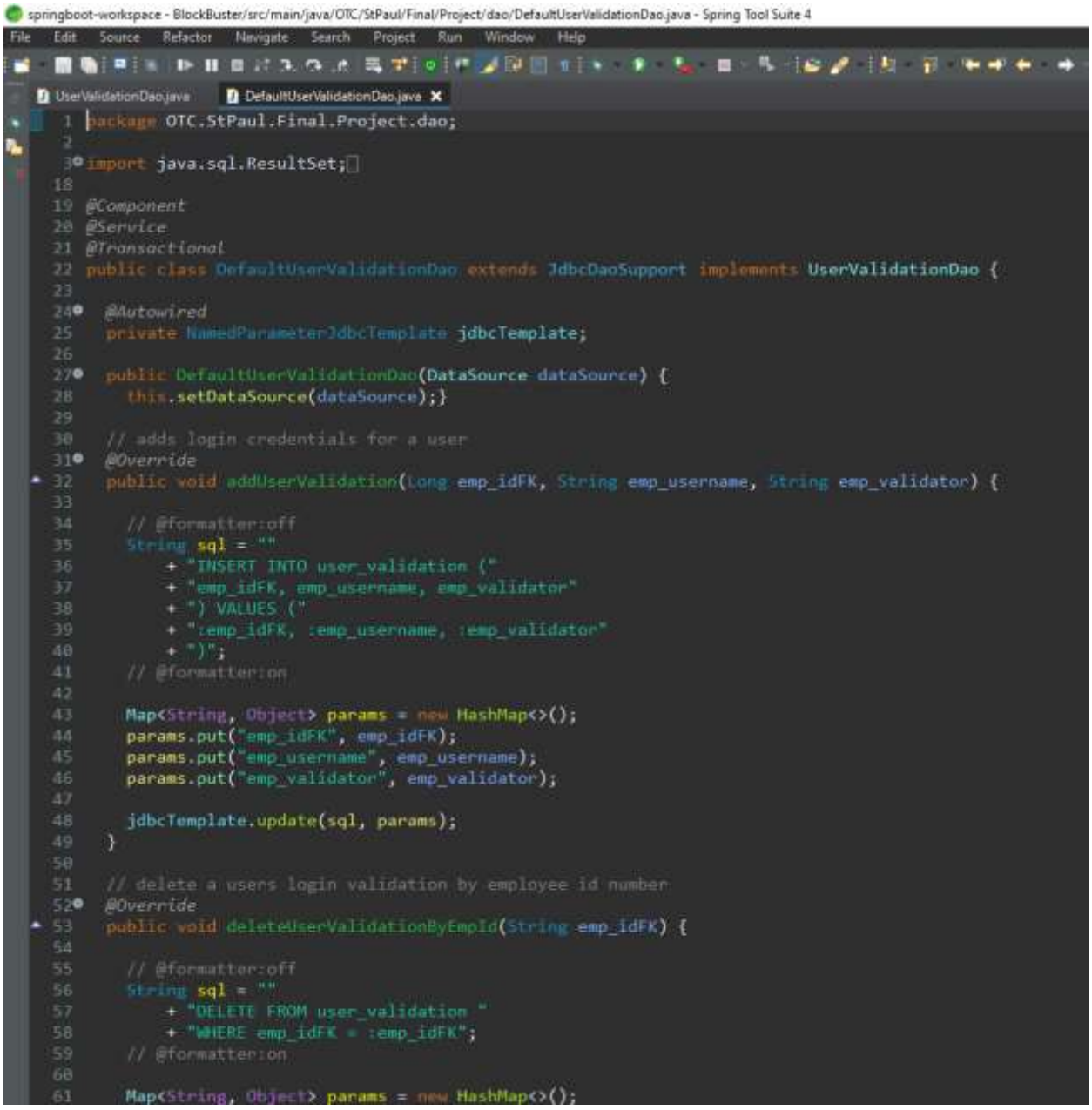
```java

  // deletes and employee by employee id number
  public void deleteEmployeeById(Long employee_idPK) {

    // @formatter:off
    String sql = ""
        + "DELETE FROM employees "
        + "WHERE employee_idPK = :employee_idPK";
    // @formatter:on

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

    jdbcTemplate.update(sql, params);
  }

  // updates an employees first name by employee id number
  public void updateEmployeeFirstNameById(String first_name, Long employee_idPK) {

    // @formatter:off
    String sql = ""
        + "UPDATE employees "
        + "SET first_name = :first_name "
        + "WHERE employee_idPK = :employee_idPK";
    // @formatter:on

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

    jdbcTemplate.update(sql, params);
  }

  // updates an employees last name by employee id number
  public void updateEmployeeLastNameById(String last_name, Long employee_idPK) {

    // @formatter:off
    String sql = ""
        + "UPDATE employees "
        + "SET last_name = :last_name "
        + "WHERE employee_idPK = :employee_idPK";
    // @formatter:on

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



```
193     params.put("employee_idPK", employee_idPK);
194
195     jdbcTemplate.update(sql, params);
196   }
197
198   // updates an employees address by employee id number
199   public void updateEmployeeAddressById(String address, Long employee_idPK) {
200
201     // @formatter:off
202     String sql = ""
203         + "UPDATE employees "
204         + "SET address = :address "
205         + "WHERE employee_idPK = :employee_idPK";
206     // @formatter:on
207
208     Map<String, Object> params = new HashMap<>();
209     params.put("address", address);
210     params.put("employee_idPK", employee_idPK);
211
212     jdbcTemplate.update(sql, params);
213   }
214
215   // updates an employees phone number by employee id number
216   public void updateEmployeePhoneById(String phone, Long employee_idPK) {
217
218     // @formatter:off
219     String sql = ""
220         + "UPDATE employees "
221         + "SET phone = :phone "
222         + "WHERE employee_idPK = :employee_idPK";
223     // @formatter:on
224
225     Map<String, Object> params = new HashMap<>();
226     params.put("phone", phone);
227     params.put("employee_idPK", employee_idPK);
228
229     jdbcTemplate.update(sql, params);
230   }
231
232 } // last bracket
```

User Validation-




springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/dao/DefaultUserValidationDao.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
UserValidationDao.java
DefaultUserValidationDao.java
1 package OTC.StPaul.Final.Project.dao;
2
3 import java.sql.ResultSet;
18
19 @Component
20 @Service
21 @Transactional
22 public class DefaultUserValidationDao extends JdbcDaoSupport implements UserValidationDao {
23
24 @Autowired
25 private NamedParameterJdbcTemplate jdbcTemplate;
26
27 public DefaultUserValidationDao(DataSource dataSource) {
28 this.setDataSource(dataSource);}
29
30 // adds login credentials for a user
31 @Override
32 public void addUserValidation(Long emp_idFK, String emp_username, String emp_validator) {
33
34 // @formatter:off
35 String sql = ""
36 + "INSERT INTO user_validation ("
37 + "emp_idFK, emp_username, emp_validator"
38 + ") VALUES ("
39 + ":emp_idFK, :emp_username, :emp_validator"
40 + ")";
41 // @formatter:on
42
43 Map<String, Object> params = new HashMap<>();
44 params.put("emp_idFK", emp_idFK);
45 params.put("emp_username", emp_username);
46 params.put("emp_validator", emp_validator);
47
48 jdbcTemplate.update(sql, params);
49 }
50
51 // delete a users login validation by employee id number
52 @Override
53 public void deleteUserValidationByEmpId(String emp_idFK) {
54
55 // @formatter:off
56 String sql = ""
57 + "DELETE FROM user_validation "
58 + "WHERE emp_idFK = :emp_idFK";
59 // @formatter:on
60
61 Map<String, Object> params = new HashMap<>();

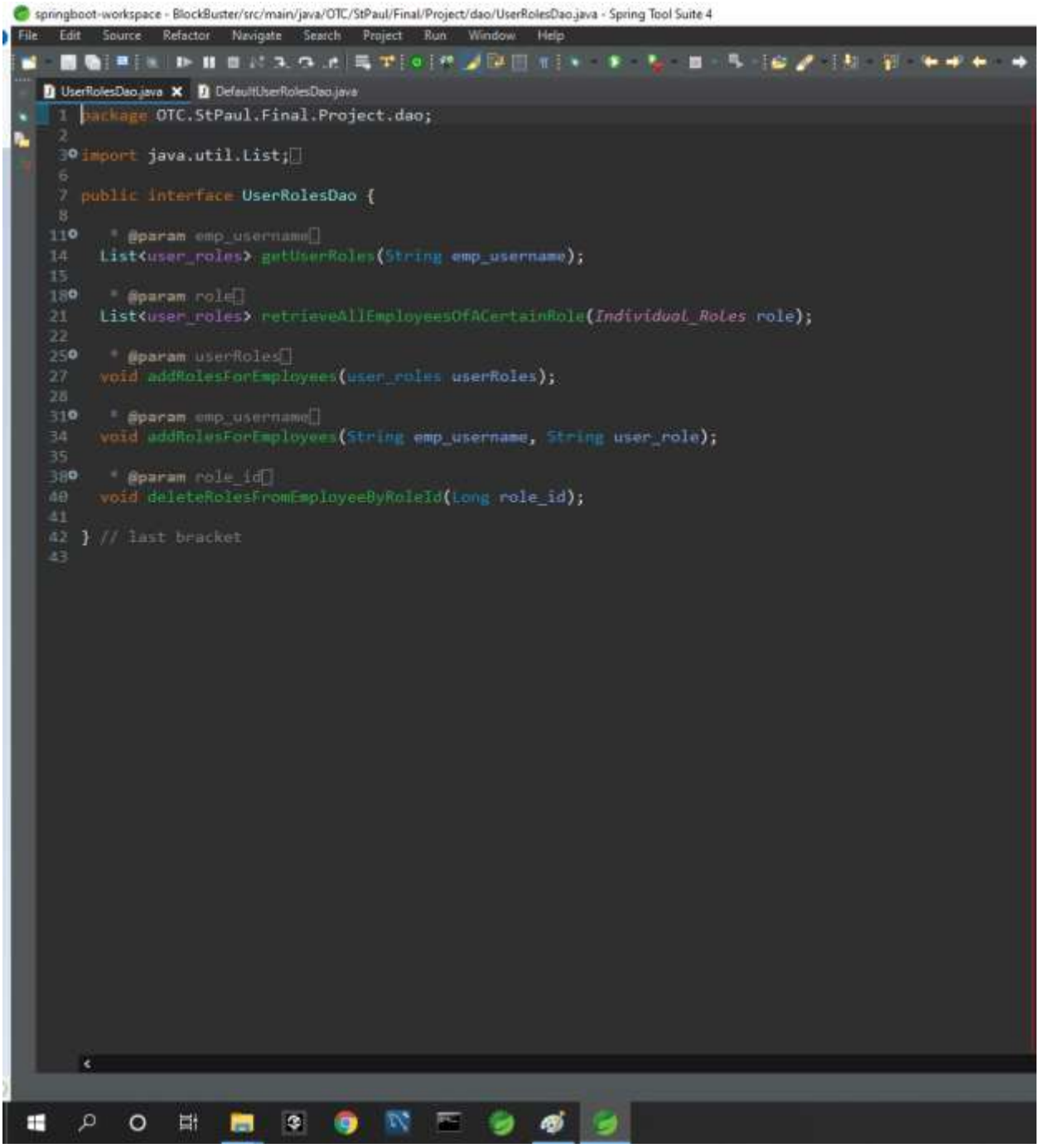
```
62      params.put("emp_idFK", emp_idFK);
63
64      jdbcTemplate.update(sql, params);
65    }
66
67    // updates an employees emp_validator (password) by employee id number & username
68    @Override
69    public void updateUserValidatorByEmpId(String emp_validator, Long emp_idFK, String emp_username) {
70
71     // @formatter:off
72     String sql = ""
73         + "UPDATE user_validation "
74         + "SET emp_validator = :emp_validator "
75         + "WHERE emp_idFK = :emp_idFK "
76         + "AND emp_username = :emp_username";
77     // @formatter:on
78
79     Map<String, Object> params = new HashMap<>();
80     params.put("emp_validator", emp_validator);
81     params.put("emp_idFK", emp_idFK);
82     params.put("emp_username", emp_username);
83
84     jdbcTemplate.update(sql, params);
85   }
86
87   // Enables a Users Validation
88   @Override
89   public void enableUserValidationByEmpId(String emp_idFK) {
90
91      // @formatter:off
92      String sql = ""
93          + "Update user_validation "
94          + "SET ENABLED = true "
95          + "WHERE emp_idFK = :emp_idFK";
96      // @formatter:on
97
98      Map<String, Object> params = new HashMap<>();
99      params.put("emp_idFK", emp_idFK);
100
101     jdbcTemplate.update(sql, params);
102   }
103
104   // Disables a Users Validation
105   @Override
106   public void disableUserValidationByEmpId(String emp_idFK) {
```

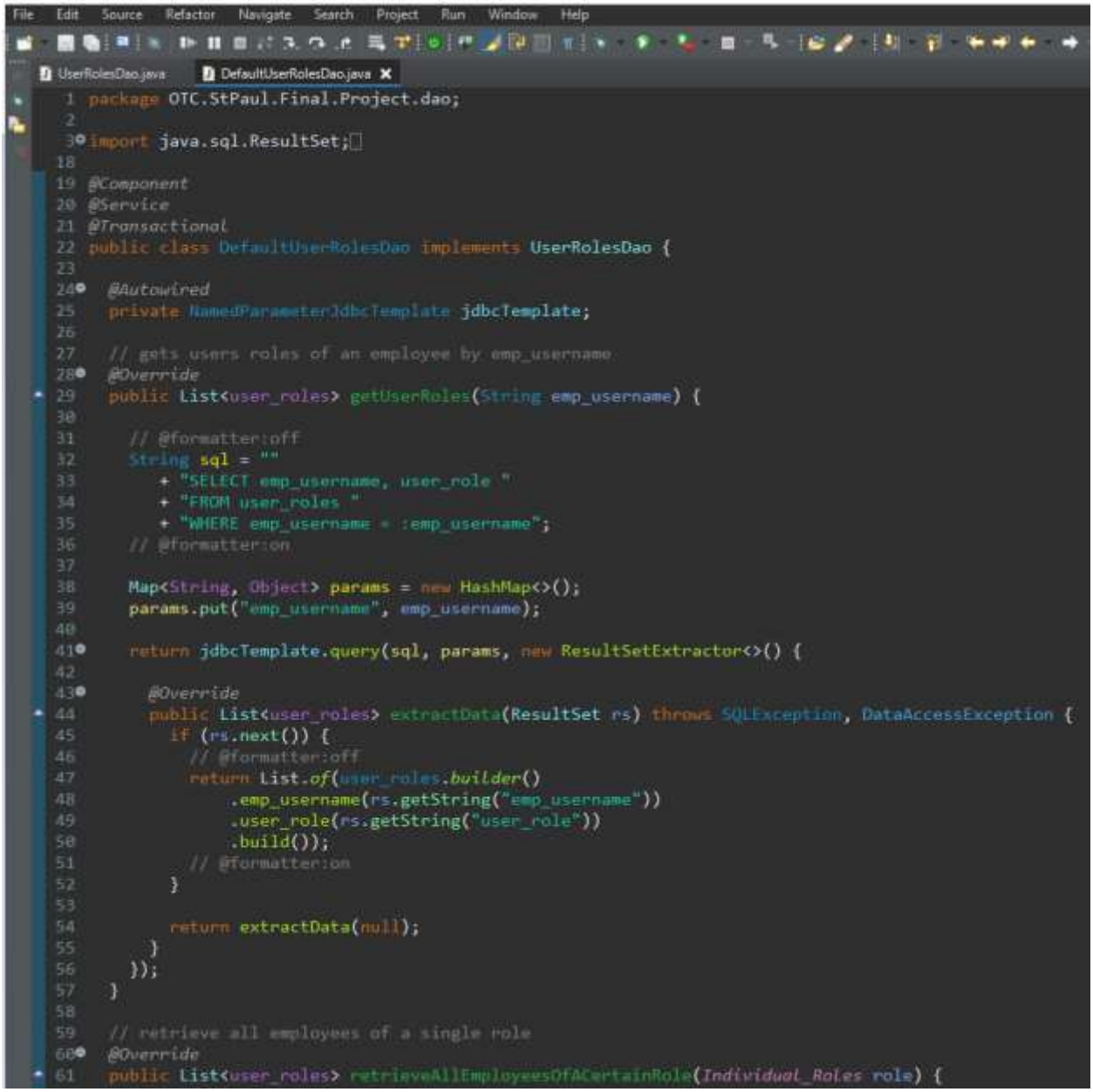
```
107
108     // @formatter:off
109     String sql = ""
110         + "Update user_validation "
111         + "SET ENABLED = false "
112         + "WHERE emp_idFK = :emp_idFK";
113     // @formatter:on
114
115     Map<String, Object> params = new HashMap<>();
116     params.put("emp_idFK", emp_idFK);
117
118     jdbcTemplate.update(sql, params);
119   }
120
121   // finds an employees emp_username by emp_idFK
122   @Override
123   public List<user_validation> findEmpUsername(Long emp_idFK) {
124
125     // @formatter:off
126     String sql = ""
127         + "SELECT emp_idFK, emp_username "
128         + "FROM user_validation "
129         + "WHERE emp_idFk = :emp_idFK";
130     // @formatter:on
131
132     Map<String, Object> params = new HashMap<>();
133     params.put("emp_idFK", emp_idFK);
134
135     return jdbcTemplate.query(sql, params, new ResultSetExtractor<>() {
136       @Override
137       public List<user_validation> extractData(ResultSet rs) throws SQLException, DataAccessException {
138         if (rs.next()) {
139           // @formatter:off
140           return List.of(user_validation.builder()
141               .emp_idFK(rs.getLong("emp_idFK"))
142               .emp_username(rs.getString("emp_username"))
143               .build());
144           // @formatter:on
145         }
146
147         return extractData(null);
148       }
149     });
150
151   }
```

```
152
153     // finds a users password for verification purposes
154     @Override
155     public List<user_validation> findUserInfo(String emp_username) {
156
157       // @formatter:off
158       String sql = ""
159           + "SELECT emp_username, emp_validator "
160           + "FROM user_validation "
161           + "WHERE emp_username = :emp_username";
162       // @formatter:on
163
164       Map<String, Object> params = new HashMap<>();
165       params.put("emp_username", emp_username);
166
167       return jdbcTemplate.query(sql, params, new ResultSetExtractor<>() {
168         @Override
169         public List<user_validation> extractData(ResultSet rs) throws SQLException, DataAccessExc
170           if (rs.next()) {
171             // @formatter:off
172             return List.of(user_validation.builder()
173                 .emp_username(rs.getString("emp_username"))
174                 .emp_validator(rs.getString("emp_validator"))
175                 .build());
176             // @formatter:on
177           }
178
179           return extractData(null);
180         }
181       });
182     }
183
184     // gets users roles for login type
185     @SuppressWarnings("deprecation")
186     @Override
187     public List<String> getUserRoles(String emp_username) {
188
189       // @formatter:off
190       String sql = ""
191           + "SELECT user_role "
192           + "FROM user_roles "
193           + "WHERE emp_username = ?";
194       // @formatter:on
```

```
195
196       Object[] params = new Object[] {emp_username};
197
198       List<String> roles = this.getJdbcTemplate().queryForList(sql, params, String.class);
199
200       return roles;
201     }
202
203 } // last bracket
204
```

User Roles –

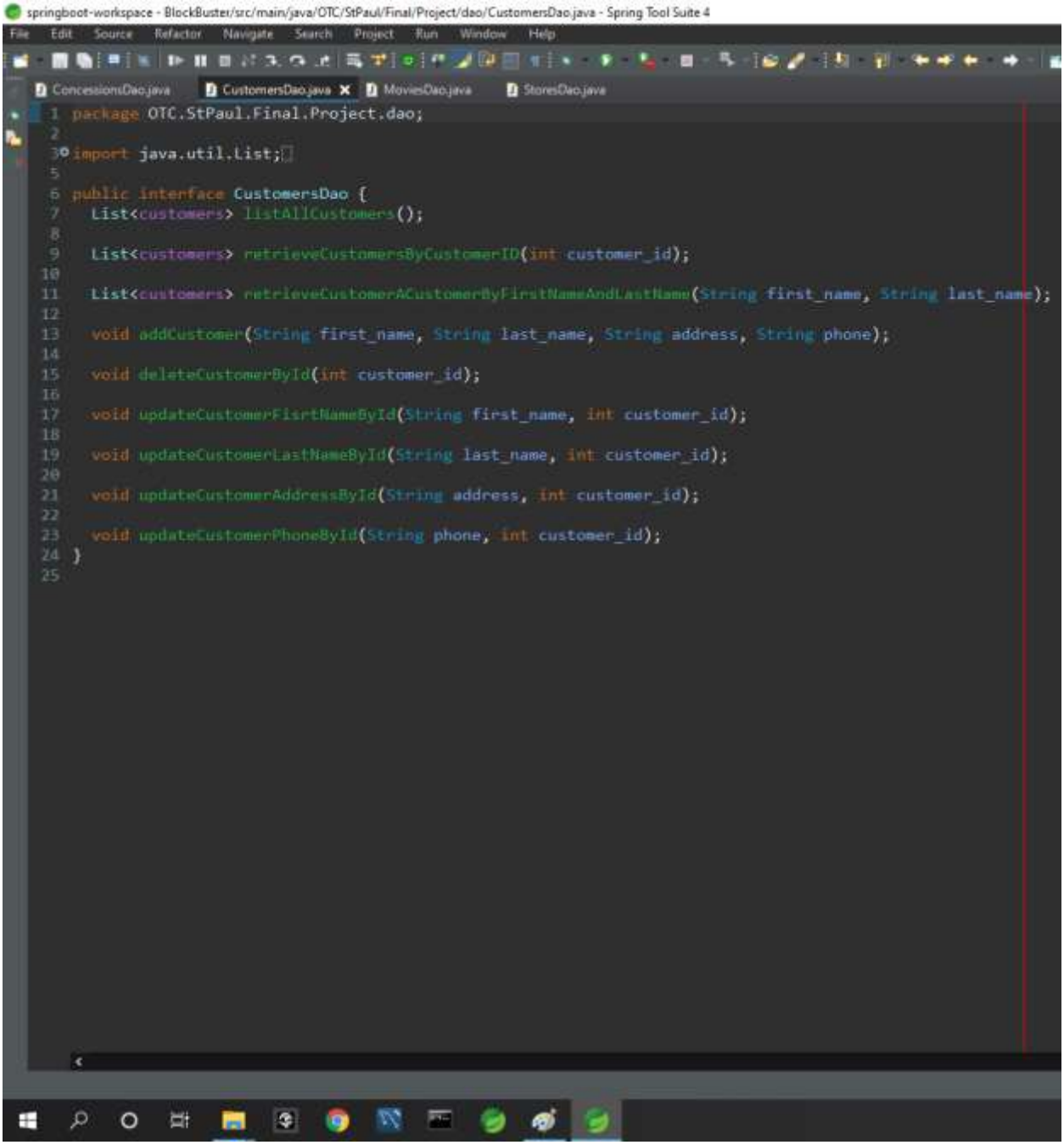

springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/dao/UserRolesDao.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
UserRolesDao.java
DefaultUserRolesDao.java
1 package OTC.StPaul.Final.Project.dao;
2
3 import java.util.List;
6
7 public interface UserRolesDao {
8
11 * @param emp_username
14 List<user_roles> getUserRoles(String emp_username);
15
18 * @param role
21 List<user_roles> retrieveAllEmployeesOfACertainRole(Individual_Roles role);
22
25 * @param userRoles
27 void addRolesForEmployees(user_roles userRoles);
28
31 * @param emp_username
34 void addRolesForEmployees(String emp_username, String user_role);
35
38 * @param role_id
40 void deleteRolesFromEmployeeByRoleId(Long role_id);
41
42 } // last bracket
43

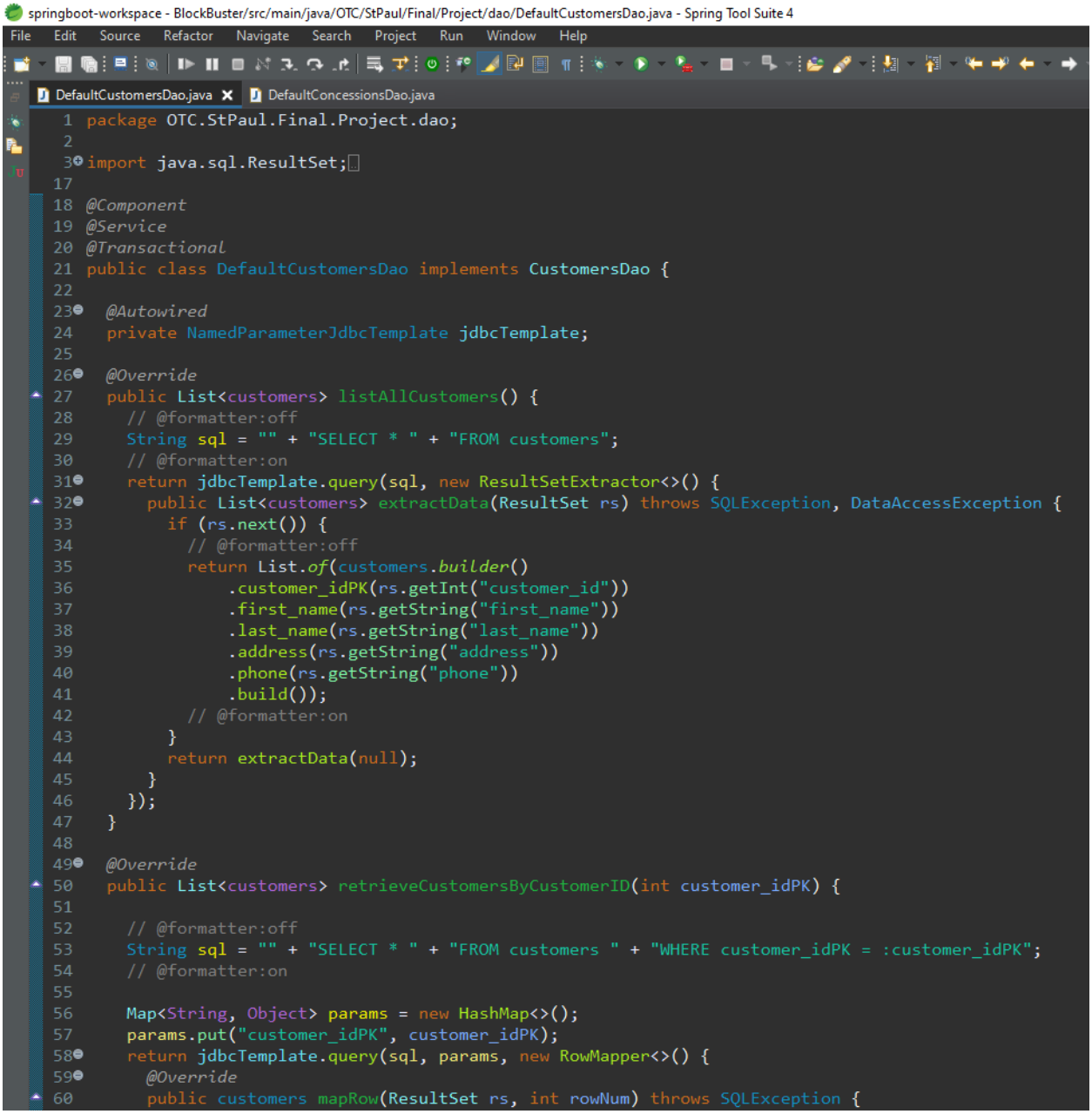

File Edit Source Refactor Navigate Search Project Run Window Help
UserRolesDao.java
DefaultUserRolesDao.java
1 package OTC.StPaul.Final.Project.dao;
3 import java.sql.ResultSet;
19 @Component
20 @Service
21 @Transactional
22 public class DefaultUserRolesDao implements UserRolesDao {
24 @Autowired
25 private NamedParameterJdbcTemplate jdbcTemplate;
27 // gets users roles of an employee by emp_username
28 @Override
29 public List<user_roles> getUserRoles(String emp_username) {
31 // @formatter:off
32 String sql = ""
33 + "SELECT emp_username, user_role "
34 + "FROM user_roles "
35 + "WHERE emp_username = :emp_username";
36 // @formatter:on
38 Map<String, Object> params = new HashMap<>();
39 params.put("emp_username", emp_username);
41 return jdbcTemplate.query(sql, params, new ResultSetExtractor<>() {
43 @Override
44 public List<user_roles> extractData(ResultSet rs) throws SQLException, DataAccessException {
45 if (rs.next()) {
46 // @formatter:off
47 return List.of(user_roles.builder()
48 .emp_username(rs.getString("emp_username"))
49 .user_role(rs.getString("user_role"))
50 .build());
51 // @formatter:on
52 }
54 return extractData(null);
55 }
56 });
57 }
59 // retrieve all employees of a single role
60 @Override
61 public List<user_roles> retrieveAllEmployeesOfACertainRole(Individual_Roles role) {

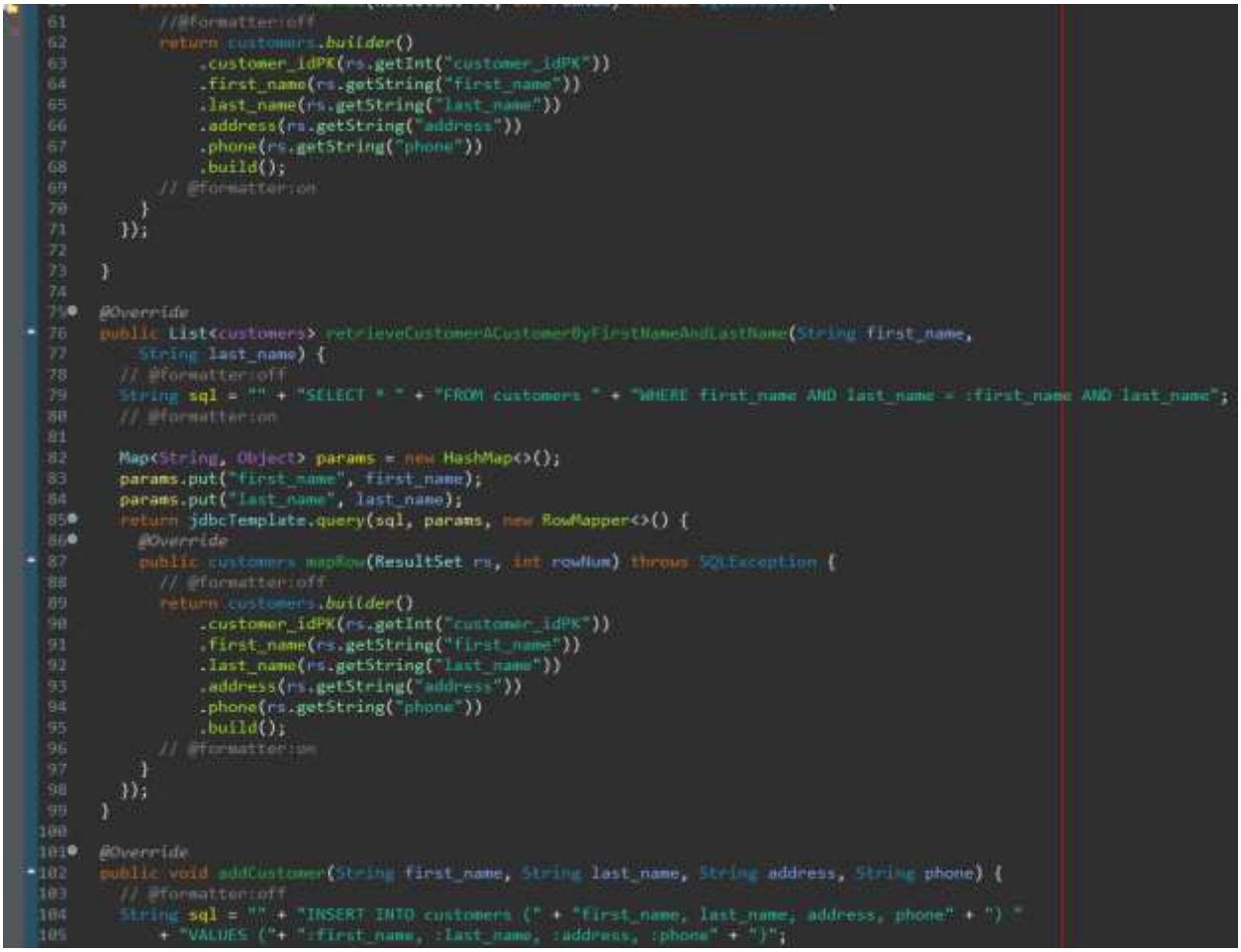
```java

    // @formatter:off
    String sql = ""
        + "SELECT * "
        + "FROM user_roles "
        + "WHERE user_role = :user_role";
    // @formatter:on

    Map<String, Object> params = new HashMap<>();
    params.put("user_role", role.toString());

    return jdbcTemplate.query(sql, params, new RowMapper<>() {

      @Override
      public user_roles mapRow(ResultSet rs, int rowNum) throws SQLException {

        // @formatter:off
          return user_roles.builder()
              .role_id(rs.getLong("role_id"))
              .emp_username(rs.getString("emp_username"))
              .user_role(rs.getString("user_role"))
              .build();
          // @formatter:on
      }
    });
  }

  // add a role for an employee
  @Override
  public void addRolesForEmployees(user_roles userRoles) {

    // @formatter:off
    String sql = ""
        + "INSERT INTO user_roles ("
        + "emp_username, user_role"
        + ") VALUES ("
        + ":emp_username, :user_role"
        + ")";
    // @formatter:on

    Map<String, Object> params = new HashMap<>();
    params.put("emp_username", userRoles.getEmp_username());
    params.put("user_role", userRoles.getUser_role());

    jdbcTemplate.update(sql, params);
```

```
    jdbcTemplate.update(sql, params);
  }

  // delete a role for an employee
  @Override
  public void deleteRolesFromEmployeeByRoleId(Long role_id) {

    // @formatter:off
    String sql = ""
        + "DELETE FROM user_roles "
        + "WHERE role_id = :role_id";
    // @formatter:on

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

    jdbcTemplate.update(sql, params);
  }

  @Override
  public void addRolesForEmployees(String emp_username, String user_role) {
    // @formatter:off
    String sql = ""
        + "INSERT INTO user_roles ("
        + "emp_username, user_role"
        + ") VALUES ("
        + ":emp_username, :user_role"
        + ")";
    // @formatter:on

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

    jdbcTemplate.update(sql, params);
  }

} // last bracket
```

Customers –

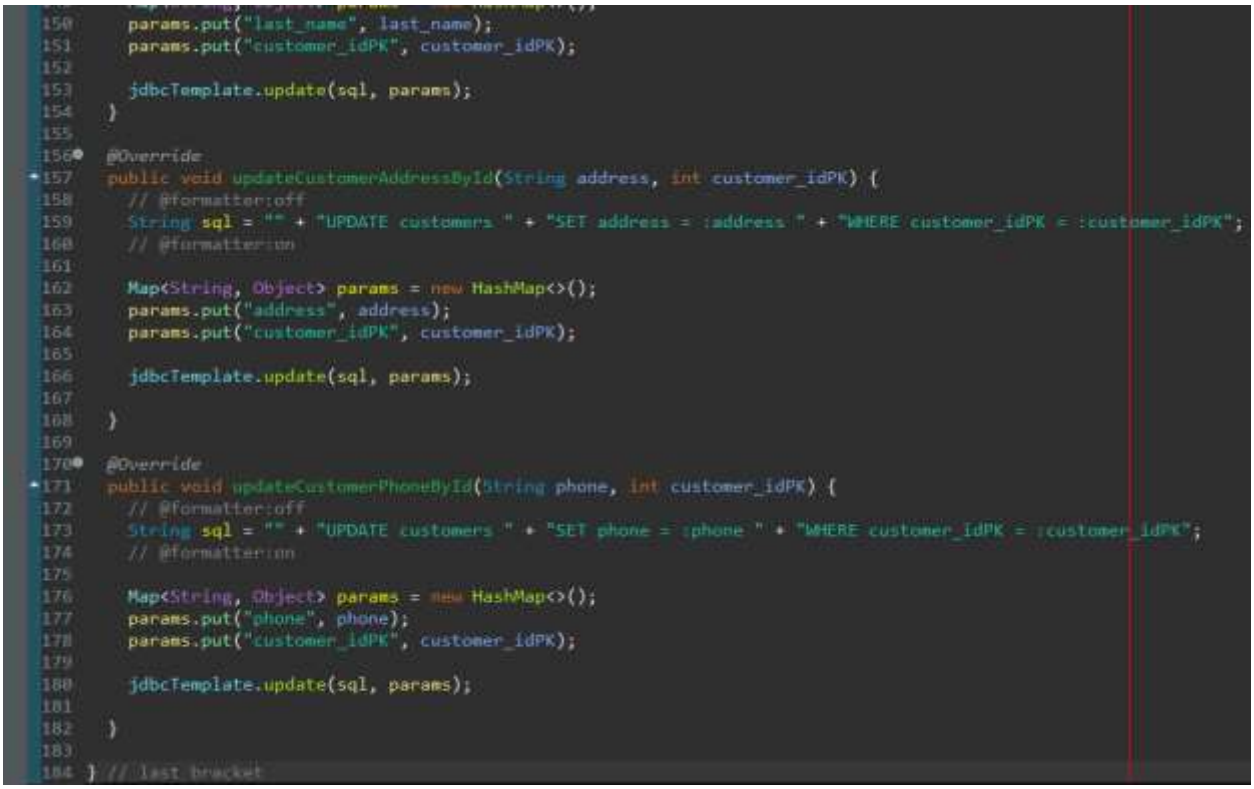

springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/dao/CustomersDao.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
ConcessionsDao.java
CustomersDao.java
MoviesDao.java
StoresDao.java
1 package OTC.StPaul.Final.Project.dao;
2
3 import java.util.List;
5
6 public interface CustomersDao {
7 List<customers> listAllCustomers();
8
9 List<customers> retrieveCustomersByCustomerID(int customer_id);
10
11 List<customers> retrieveCustomerACustomerByFirstNameAndLastName(String first_name, String last_name);
12
13 void addCustomer(String first_name, String last_name, String address, String phone);
14
15 void deleteCustomerById(int customer_id);
16
17 void updateCustomerFisrtNameById(String first_name, int customer_id);
18
19 void updateCustomerLastNameById(String last_name, int customer_id);
20
21 void updateCustomerAddressById(String address, int customer_id);
22
23 void updateCustomerPhoneById(String phone, int customer_id);
24 }
25


springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/dao/DefaultCustomersDao.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
DefaultCustomersDao.java
DefaultConcessionsDao.java
1 package OTC.StPaul.Final.Project.dao;
2
3 import java.sql.ResultSet;
17
18 @Component
19 @Service
20 @Transactional
21 public class DefaultCustomersDao implements CustomersDao {
22
23 @Autowired
24 private NamedParameterJdbcTemplate jdbcTemplate;
25
26 @Override
27 public List<customers> listAllCustomers() {
28 // @formatter:off
29 String sql = "" + "SELECT * " + "FROM customers";
30 // @formatter:on
31 return jdbcTemplate.query(sql, new ResultSetExtractor<>() {
32 public List<customers> extractData(ResultSet rs) throws SQLException, DataAccessException {
33 if (rs.next()) {
34 // @formatter:off
35 return List.of(customers.builder()
36 .customer_idPK(rs.getInt("customer_id"))
37 .first_name(rs.getString("first_name"))
38 .last_name(rs.getString("last_name"))
39 .address(rs.getString("address"))
40 .phone(rs.getString("phone"))
41 .build());
42 // @formatter:on
43 }
44 return extractData(null);
45 }
46 });
47 }
48
49 @Override
50 public List<customers> retrieveCustomersByCustomerID(int customer_idPK) {
51
52 // @formatter:off
53 String sql = "" + "SELECT * " + "FROM customers " + "WHERE customer_idPK = :customer_idPK";
54 // @formatter:on
55
56 Map<String, Object> params = new HashMap<>();
57 params.put("customer_idPK", customer_idPK);
58 return jdbcTemplate.query(sql, params, new RowMapper<>() {
59 @Override
60 public customers mapRow(ResultSet rs, int rowNum) throws SQLException {



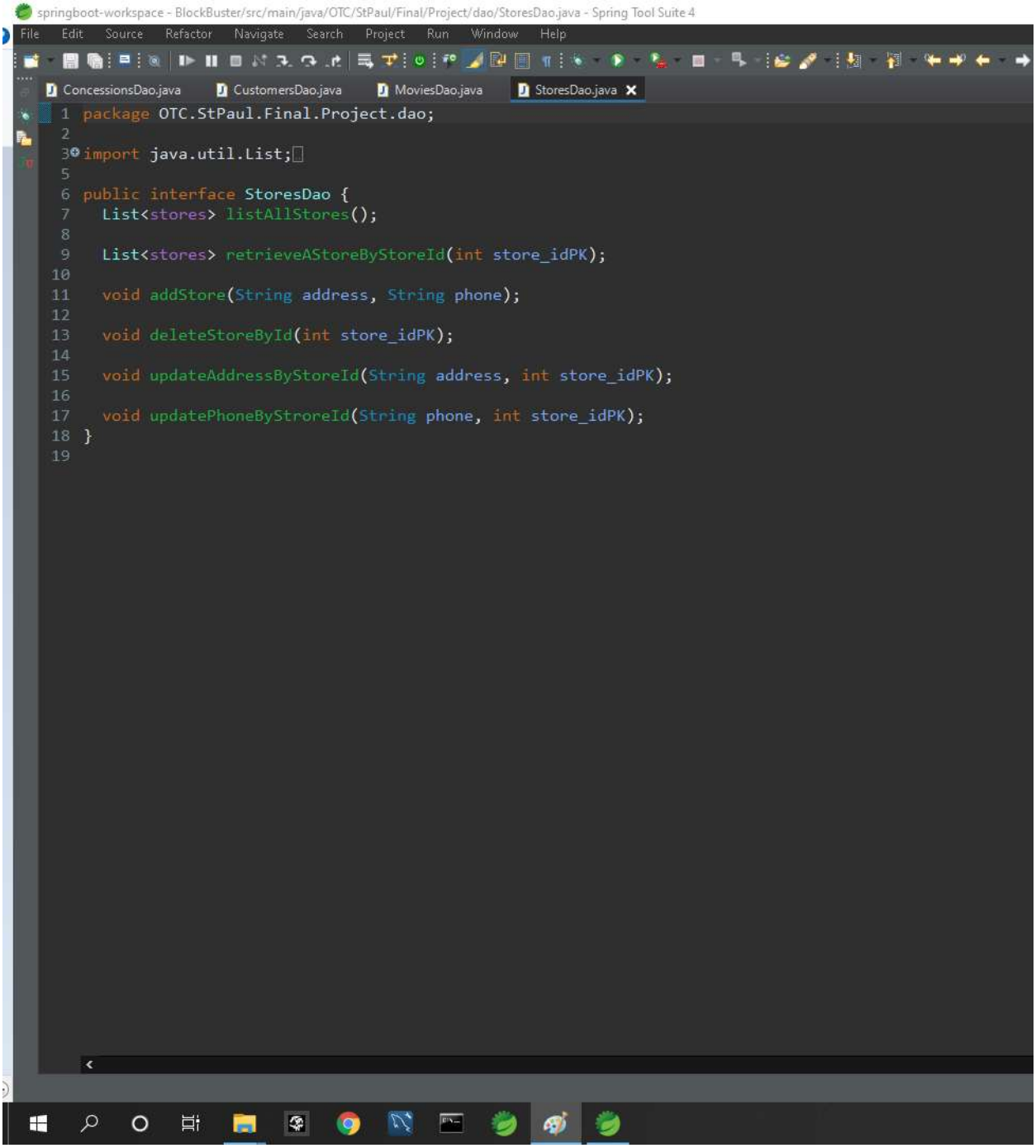
```
        //@formatter:off
        return customers.builder()
            .customer_idPK(rs.getInt("customer_idPK"))
            .first_name(rs.getString("first_name"))
            .last_name(rs.getString("last_name"))
            .address(rs.getString("address"))
            .phone(rs.getString("phone"))
            .build();
        // @formatter:on
      }
    });

  }
  @Override
  public List<customers> retrieveCustomerACustomerByFirstNameAndLastName(String first_name,
      String last_name) {
    // @formatter:off
    String sql = "" + "SELECT * " + "FROM customers " + "WHERE first_name AND last_name = :first_name AND last_name";
    // @formatter:on

    Map<String, Object> params = new HashMap<>();
    params.put("first_name", first_name);
    params.put("last_name", last_name);
    return jdbcTemplate.query(sql, params, new RowMapper<>() {
      @Override
      public customers mapRow(ResultSet rs, int rowNum) throws SQLException {
        // @formatter:off
        return customers.builder()
            .customer_idPK(rs.getInt("customer_idPK"))
            .first_name(rs.getString("first_name"))
            .last_name(rs.getString("last_name"))
            .address(rs.getString("address"))
            .phone(rs.getString("phone"))
            .build();
        // @formatter:on
      }
    });
  }

  @Override
  public void addCustomer(String first_name, String last_name, String address, String phone) {
    // @formatter:off
    String sql = "" + "INSERT INTO customers (" + "first_name, last_name, address, phone" + ") "
        + "VALUES ("+ ":first_name, :last_name, :address, :phone" + ")";
```

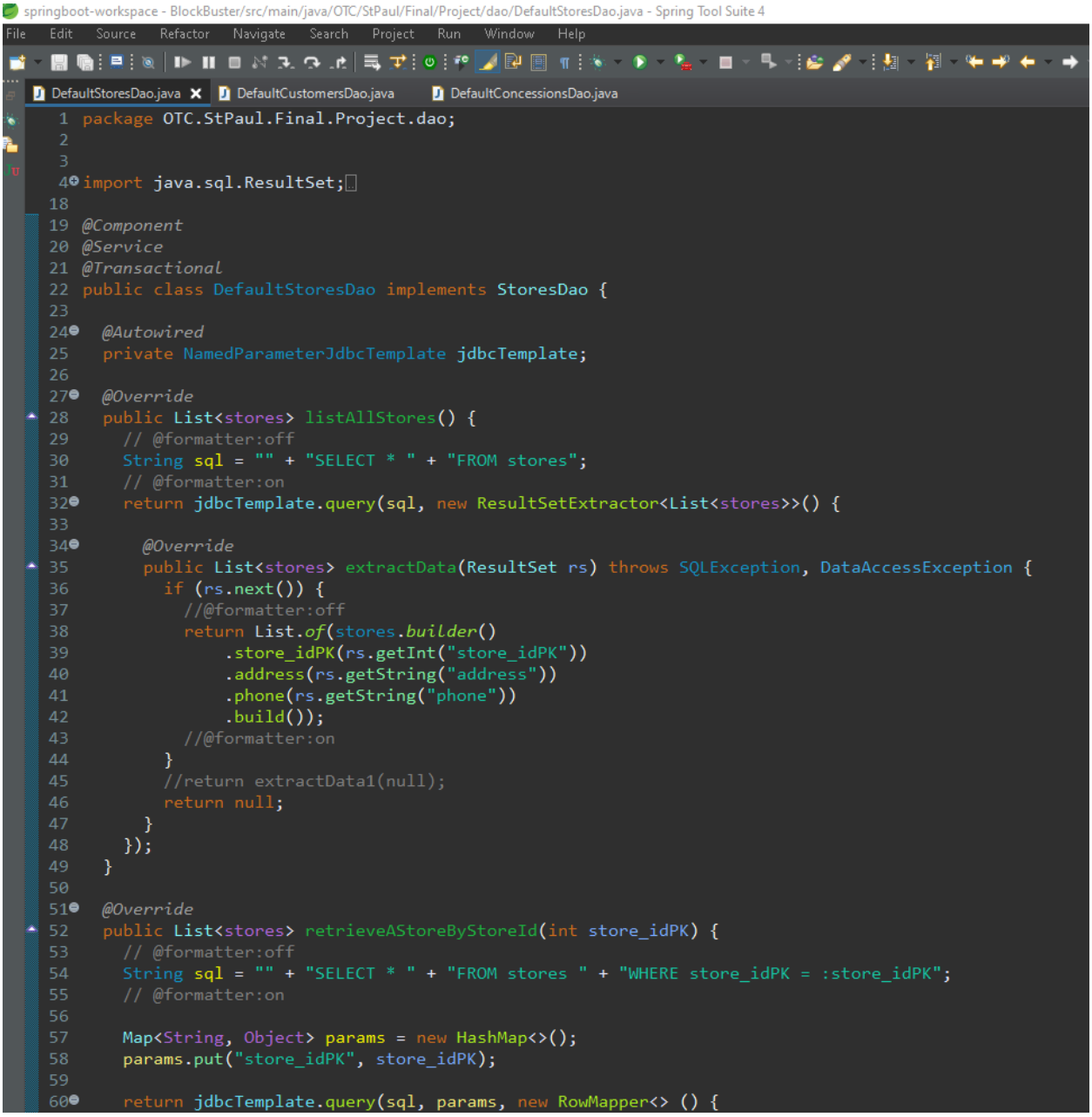
```
106     // @formatter:on
107     Map<String, Object> params = new HashMap<>();
108     params.put("first_name", first_name);
109     params.put("last_name", last_name);
110     params.put("address", address);
111     params.put("phone", phone);
112
113     jdbcTemplate.update(sql, params);
114   }
115
116   @Override
117   public void deleteCustomerById(int customer_idPK) {
118     // @formatter:off
119     String sql = "" + "DELETE FROM customers " + "WHERE customer_idPK = :customer_idPK";
120     // @formatter:on
121
122     Map<String, Object> params = new HashMap<>();
123     params.put("customer_idPK", customer_idPK);
124
125     jdbcTemplate.update(sql, params);
126
127   }
128
129   @Override
130   public void updateCustomerFisrtNameById(String first_name, int customer_idPK) {
131     // @formatter:off
132     String sql = "" + "UPDATE customers " + "SET first_name = :first_name " + "WHERE customer_idPK = :customer_idPK";
133     // @formatter:on
134
135     Map<String, Object> params = new HashMap<>();
136     params.put("first_name", first_name);
137     params.put("customer_idPK", customer_idPK);
138
139     jdbcTemplate.update(sql, params);
140
141   }
142
143   @Override
144   public void updateCustomerLastNameById(String last_name, int customer_idPK) {
145     // @formatter:off
146     String sql = "" + "UPDATE customers " + "SET last_name = :last_name " + "WHERE customer_idPK = :customer_idPK";
147     // @formatter:on
148
149     Map<String, Object> params = new HashMap<>();
150     params.put("last_name", last_name);
```



```
    params.put("last_name", last_name);
    params.put("customer_idPK", customer_idPK);

    jdbcTemplate.update(sql, params);
  }

  @Override
  public void updateCustomerAddressById(String address, int customer_idPK) {
    // @formatter:off
    String sql = "" + "UPDATE customers " + "SET address = :address " + "WHERE customer_idPK = :customer_idPK";
    // @formatter:on

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

    jdbcTemplate.update(sql, params);

  }

  @Override
  public void updateCustomerPhoneById(String phone, int customer_idPK) {
    // @formatter:off
    String sql = "" + "UPDATE customers " + "SET phone = :phone " + "WHERE customer_idPK = :customer_idPK";
    // @formatter:on

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

    jdbcTemplate.update(sql, params);

  }

} // last bracket
```

Stores-

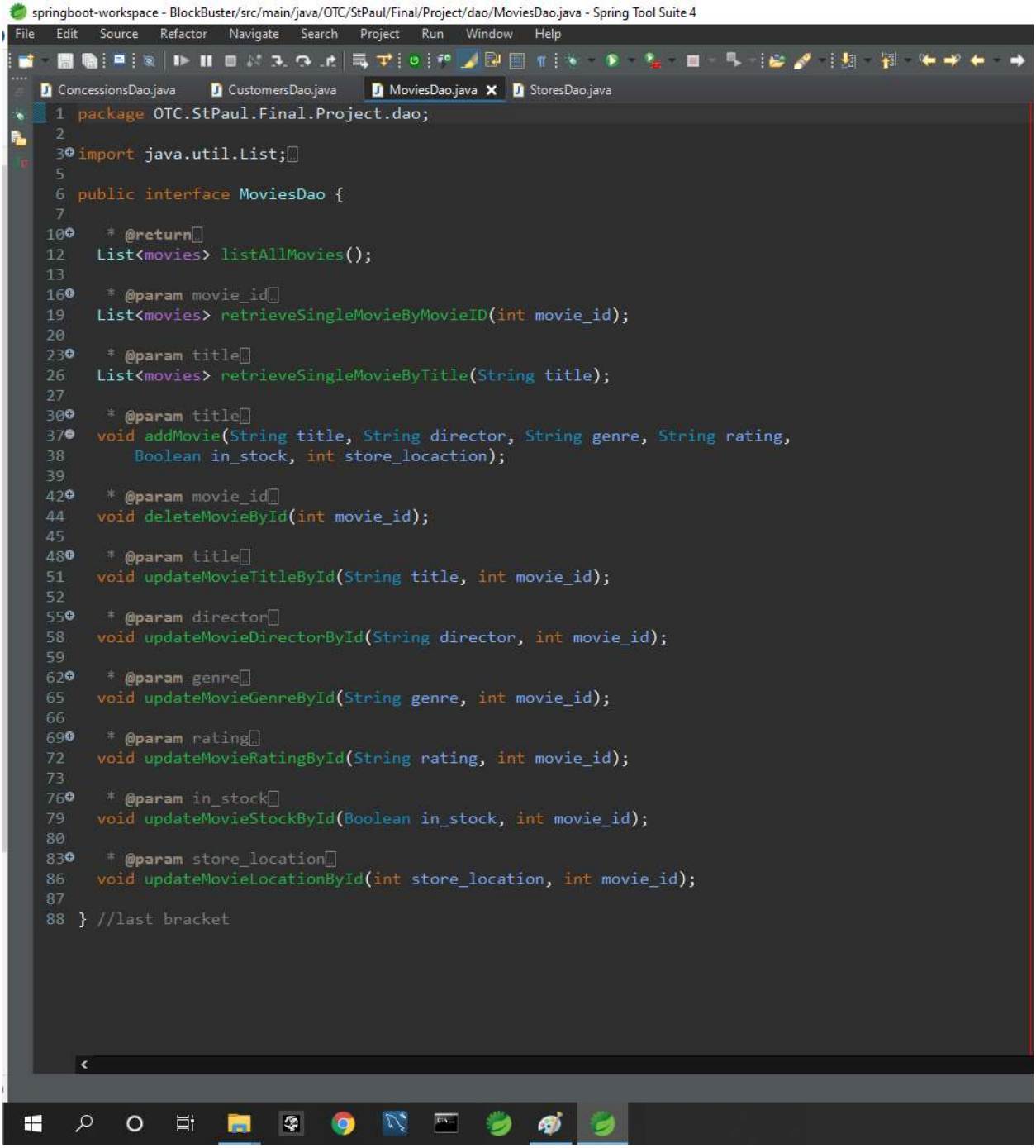

springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/dao/StoresDao.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
ConcessionsDao.java
CustomersDao.java
MoviesDao.java
StoresDao.java
1 package OTC.StPaul.Final.Project.dao;
2
3 import java.util.List;
5
6 public interface StoresDao {
7 List<stores> listAllStores();
8
9 List<stores> retrieveAStoreByStoreId(int store_idPK);
10
11 void addStore(String address, String phone);
12
13 void deleteStoreById(int store_idPK);
14
15 void updateAddressByStoreId(String address, int store_idPK);
16
17 void updatePhoneByStroreId(String phone, int store_idPK);
18 }
19

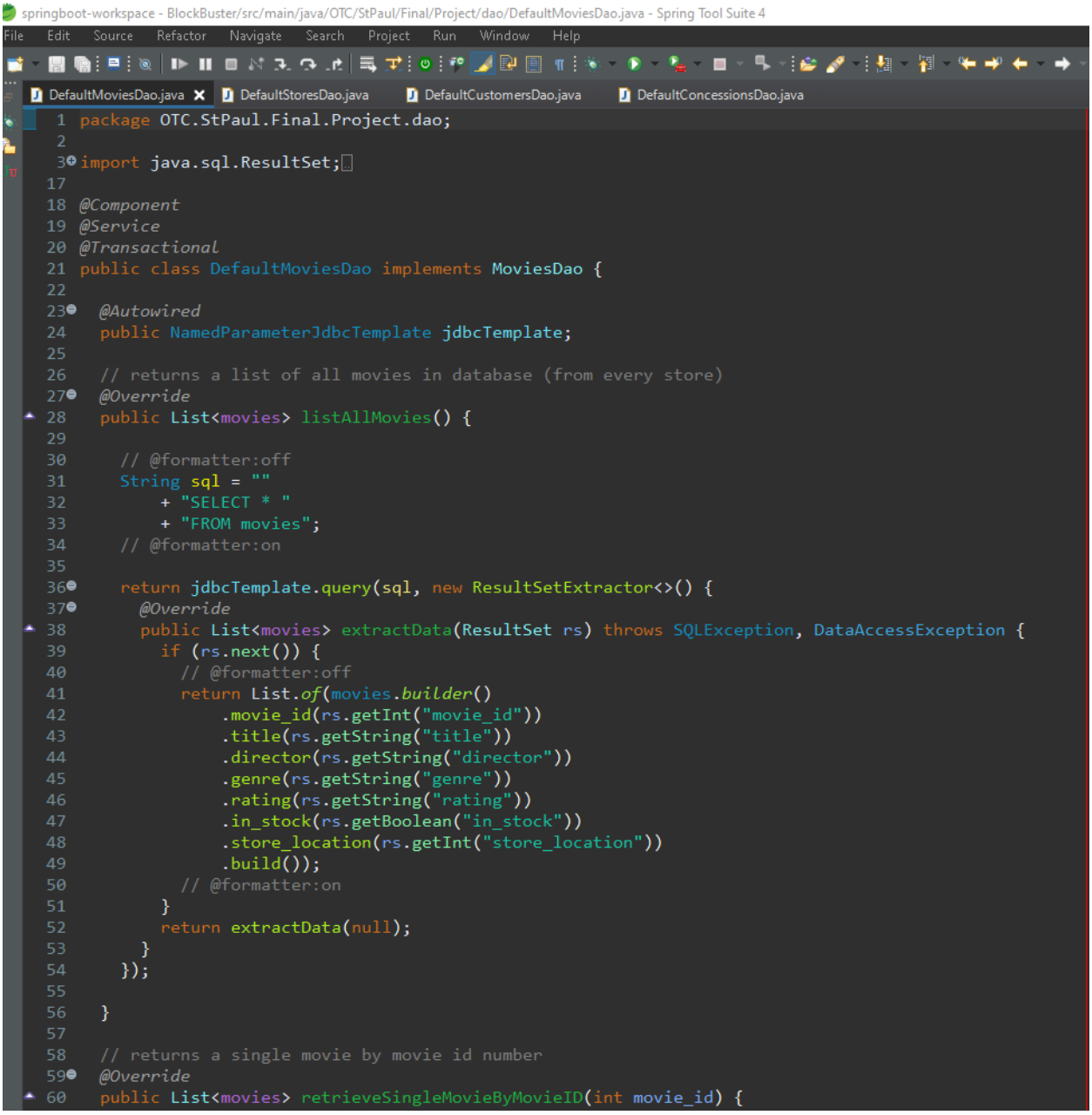

springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/dao/DefaultStoresDao.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
DefaultStoresDao.java
DefaultCustomersDao.java
DefaultConcessionsDao.java
1 package OTC.StPaul.Final.Project.dao;
4 import java.sql.ResultSet;
19 @Component
20 @Service
21 @Transactional
22 public class DefaultStoresDao implements StoresDao {
24 @Autowired
25 private NamedParameterJdbcTemplate jdbcTemplate;
27 @Override
28 public List<stores> listAllStores() {
29 // @formatter:off
30 String sql = "" + "SELECT * " + "FROM stores";
31 // @formatter:on
32 return jdbcTemplate.query(sql, new ResultSetExtractor<List<stores>>() {
34 @Override
35 public List<stores> extractData(ResultSet rs) throws SQLException, DataAccessException {
36 if (rs.next()) {
37 //@formatter:off
38 return List.of(stores.builder()
39 .store_idPK(rs.getInt("store_idPK"))
40 .address(rs.getString("address"))
41 .phone(rs.getString("phone"))
42 .build());
43 //@formatter:on
44 }
45 //return extractData1(null);
46 return null;
47 }
48 });
49 }
51 @Override
52 public List<stores> retrieveAStoreByStoreId(int store_idPK) {
53 // @formatter:off
54 String sql = "" + "SELECT * " + "FROM stores " + "WHERE store_idPK = :store_idPK";
55 // @formatter:on
57 Map<String, Object> params = new HashMap<>();
58 params.put("store_idPK", store_idPK);
60 return jdbcTemplate.query(sql, params, new RowMapper<> () {

```
62        public stores mapRow(ResultSet rs, int rowNum) throws SQLException, DataAccessException {
63            // @formatter:off
64            return stores.builder()
65                .store_idPK(rs.getInt("store_idPK"))
66                .address(rs.getString("address"))
67                .phone(rs.getString("phone"))
68                .build();
69            // @formatter:on
70        }
71    });
72  }
73
74  @Override
75  public void addStore(String address, String phone) {
76    // @formatter:off
77    String sql = "" + "INSERT INTO stores (" + "address, phone" + ") "
78        + "VALUES (" + ":address, :phone" + ")";
79    // @formatter:on
80
81    Map<String, Object> params = new HashMap<> ();
82    params.put("address", address);
83    params.put("phone", phone);
84
85    jdbcTemplate.update(sql, params);
86  }
87
88  @Override
89  public void deleteStoreById(int store_idPK) {
90    // @formatter:off
91    String sql = "" + "DELETE FROM stores "
92    + "WHERE store_idPK = :store_idPK";
93    // @formatter:on
94
95    Map<String, Object> params = new HashMap<>();
96    params.put("store_idPK", store_idPK);
97
98    jdbcTemplate.update(sql, params);
99
100 }
101
102 @Override
103 public void updateAddressByStoreId(String address, int store_idPK) {
104   // @formatter:off
105   String sql = "" + "UPDATE stores " + "SET address = :address " + "WHERE store_idPK = :store_idPK";
```

```
106   // @formatter:on
107
108   Map<String, Object> params = new HashMap<>();
109   params.put("address", address);
110   params.put("store_idPK", store_idPK);
111
112   jdbcTemplate.update(sql, params);
113
114 }
115
116 @Override
117 public void updatePhoneByStroreId(String phone, int store_idPK) {
118   // @formatter:off
119   String sql = "" + "UPDATE stores " + "SET phone = :phone " + "WHERE store_idPK = :store_idPK";
120   // @formatter:on
121
122   Map<String, Object> params = new HashMap<>();
123   params.put("phone", phone);
124   params.put("store_idPK", store_idPK);
125
126   jdbcTemplate.update(sql, params);
127
128 }
129
130 } // last bracket
```

Movies-


springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/dao/MoviesDao.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
ConcessionsDao.java
CustomersDao.java
MoviesDao.java
StoresDao.java
1 package OTC.StPaul.Final.Project.dao;
2
3 import java.util.List;
5
6 public interface MoviesDao {
7
10 * @return
12 List<movies> listAllMovies();
13
16 * @param movie_id
19 List<movies> retrieveSingleMovieByMovieID(int movie_id);
20
23 * @param title
26 List<movies> retrieveSingleMovieByTitle(String title);
27
30 * @param title
37 void addMovie(String title, String director, String genre, String rating,
38 Boolean in_stock, int store_locaction);
39
42 * @param movie_id
44 void deleteMovieById(int movie_id);
45
48 * @param title
51 void updateMovieTitleById(String title, int movie_id);
52
55 * @param director
58 void updateMovieDirectorById(String director, int movie_id);
59
62 * @param genre
65 void updateMovieGenreById(String genre, int movie_id);
66
69 * @param rating
72 void updateMovieRatingById(String rating, int movie_id);
73
76 * @param in_stock
79 void updateMovieStockById(Boolean in_stock, int movie_id);
80
83 * @param store_location
86 void updateMovieLocationById(int store_location, int movie_id);
87
88 } //last bracket

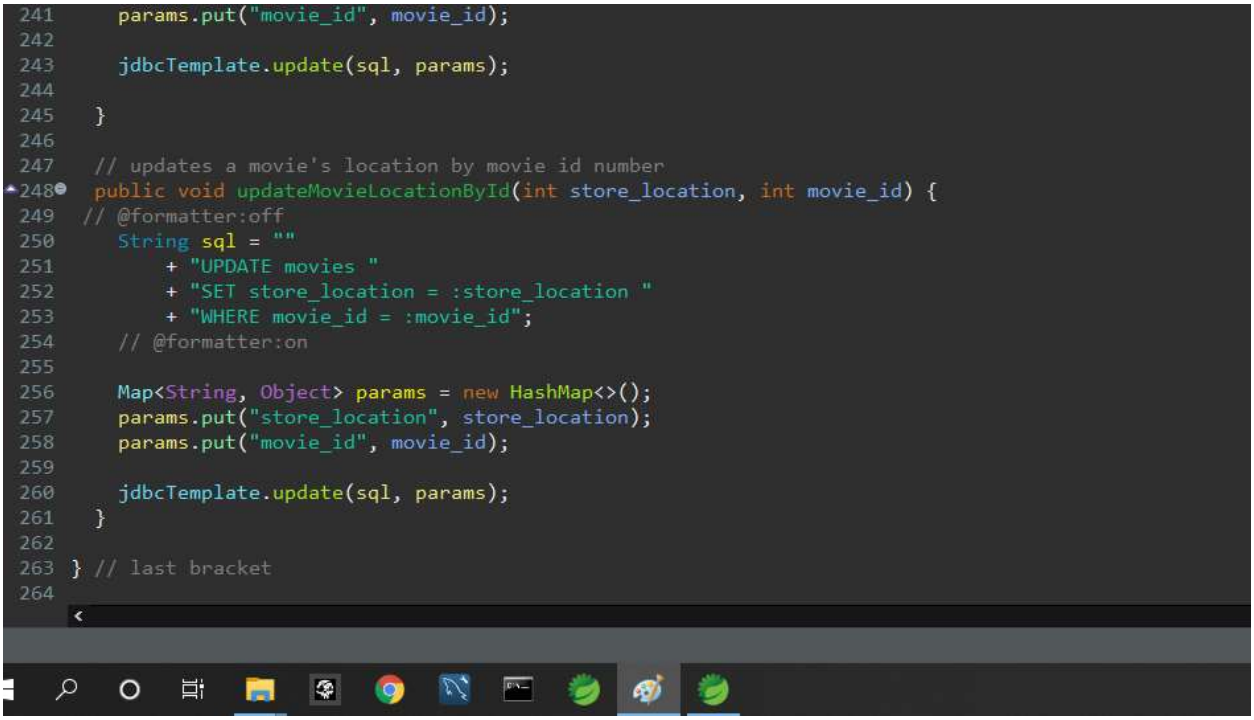

springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/dao/DefaultMoviesDao.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
DefaultMoviesDao.java
DefaultStoresDao.java
DefaultCustomersDao.java
DefaultConcessionsDao.java
1 package OTC.StPaul.Final.Project.dao;
2
3 import java.sql.ResultSet;
17
18 @Component
19 @Service
20 @Transactional
21 public class DefaultMoviesDao implements MoviesDao {
22
23 @Autowired
24 public NamedParameterJdbcTemplate jdbcTemplate;
25
26 // returns a list of all movies in database (from every store)
27 @Override
28 public List<movies> listAllMovies() {
29
30 // @formatter:off
31 String sql = ""
32 + "SELECT * "
33 + "FROM movies";
34 // @formatter:on
35
36 return jdbcTemplate.query(sql, new ResultSetExtractor<>() {
37 @Override
38 public List<movies> extractData(ResultSet rs) throws SQLException, DataAccessException {
39 if (rs.next()) {
40 // @formatter:off
41 return List.of(movies.builder()
42 .movie_id(rs.getInt("movie_id"))
43 .title(rs.getString("title"))
44 .director(rs.getString("director"))
45 .genre(rs.getString("genre"))
46 .rating(rs.getString("rating"))
47 .in_stock(rs.getBoolean("in_stock"))
48 .store_location(rs.getInt("store_location"))
49 .build());
50 // @formatter:on
51 }
52 return extractData(null);
53 }
54 });
55
56 }
57
58 // returns a single movie by movie id number
59 @Override
60 public List<movies> retrieveSingleMovieByMovieID(int movie_id) {

```
61
62      // @formatter:off
63      String sql = ""
64          + "SELECT * "
65          + "FROM movies "
66          + "WHERE movie_id = :movie_id";
67      // @formatter:on
68
69      Map<String, Object> params = new HashMap<>();
70      params.put("movie_id", movie_id);
71
72      return jdbcTemplate.query(sql, params, new RowMapper<>() {
73
74        @Override
75        public movies mapRow(ResultSet rs, int rowNum) throws SQLException {
76
77          // @formatter:off
78            return movies.builder()
79                .movie_id(rs.getInt("movie_id"))
80                .title(rs.getString("title"))
81                .director(rs.getString("director"))
82                .genre(rs.getString("genre"))
83                .rating(rs.getString("rating"))
84                .in_stock(rs.getBoolean("in_stock"))
85                .store_location(rs.getInt("store_location"))
86                .build();
87            // @formatter:on
88        }
89      });
90    }
91
92    // returns a single movie by title
93    @Override
94    public List<movies> retrieveSingleMovieByTitle(String title) {
95   // @formatter:off
96      String sql = ""
97          + "SELECT * "
98          + "FROM movies "
99          + "WHERE title = :title";
100     // @formatter:on
101
102     Map<String, Object> params = new HashMap<>();
103     params.put("title", title);
104
105     return jdbcTemplate.query(sql, params, new RowMapper<>() {
```

```
105     return jdbcTemplate.query(sql, params, new RowMapper<>() {
106
107       @Override
108       public movies mapRow(ResultSet rs, int rowNum) throws SQLException {
109
110         // @formatter:off
111           return movies.builder()
112               .movie_id(rs.getInt("movie_id"))
113               .title(rs.getString("title"))
114               .director(rs.getString("director"))
115               .genre(rs.getString("genre"))
116               .rating(rs.getString("rating"))
117               .in_stock(rs.getBoolean("in_stock"))
118               .store_location(rs.getInt("store_location"))
119               .build();
120           // @formatter:on
121       }
122     });
123   }
124
125   @Override
126   public void addMovie(String title, String director, String genre, String rating, Boolean in_stock,
127       int store_location) {
128     // @formatter:off
129     String sql = ""
130         + "INSERT INTO movies ("
131         + "title, director, genre, rating, in_stock, store_location"
132         + ")VALUES ("
133         + ":title, :director, :genre, :rating, :in_stock, :store_location"
134         + ")";
135     // @formatter:on
136     Map<String, Object> params = new HashMap<>();
137
138     params.put("title", title);
139     params.put("director", director);
140     params.put("genre", genre);
141     params.put("rating", rating);
142     params.put("in_stock", in_stock);
143     params.put("store_location", store_location);
144
145     jdbcTemplate.update(sql, params);
146   }
147
148   // deletes a movie by id number
149   public void deleteMovieById(int movie_id) {
150     // @formatter:off
```

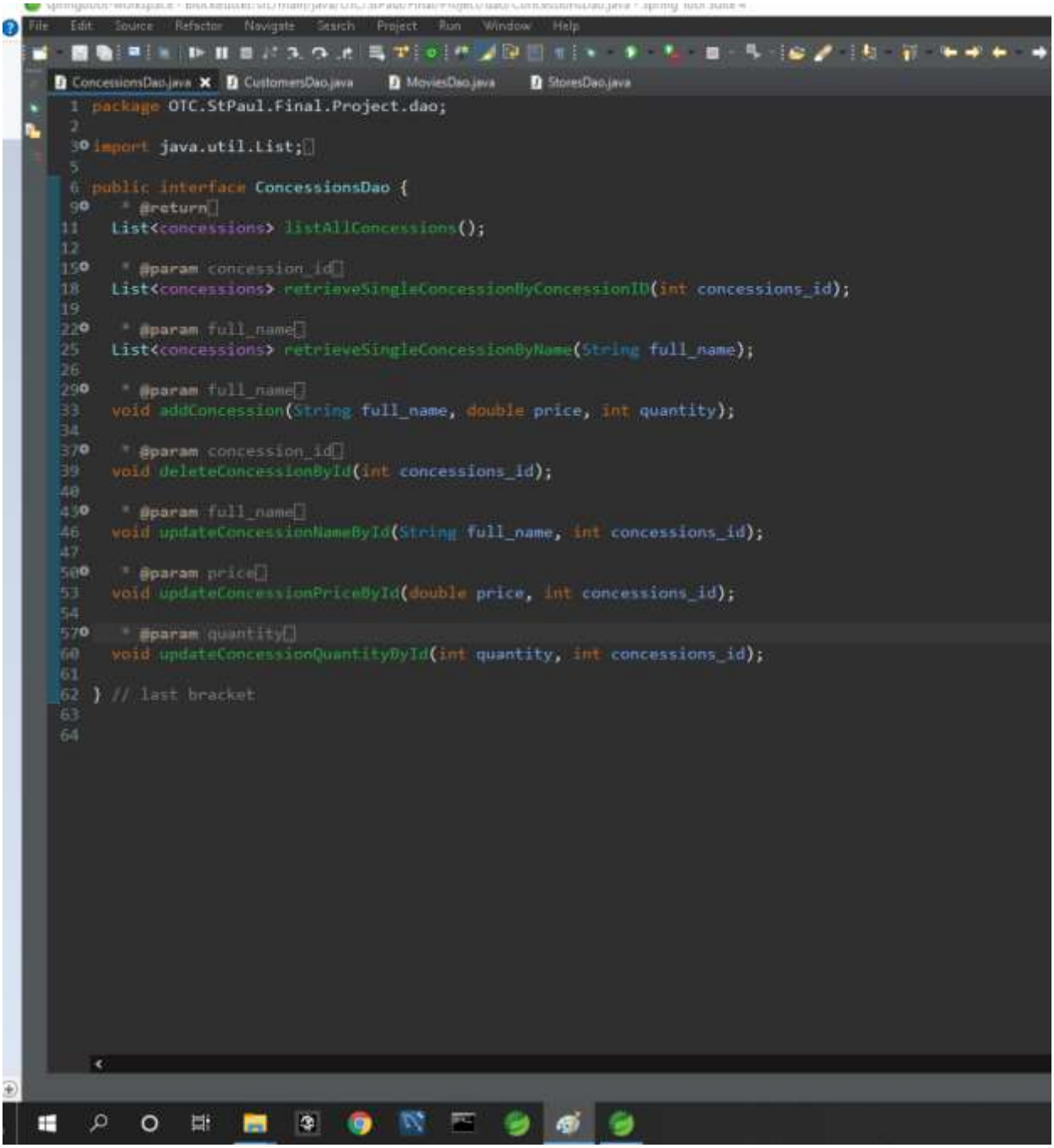
```
151     String sql = ""
152         + "DELETE FROM movies "
153         + "WHERE movie_id = :movie_id";
154     // @formatter:on
155 
156     Map<String, Object> params = new HashMap<>();
157     params.put("movie_id", movie_id);
158 
159     jdbcTemplate.update(sql,  params);
160   }
161 
162   // updates a movie title by movie id number
163   public void updateMovieTitleById(String title, int movie_id) {
164   // @formatter:off
165     String sql = ""
166         + "UPDATE movies "
167         + "SET title = :title "
168         + "WHERE movie_id = :movie_id";
169     // @formatter:on
170 
171     Map<String, Object> params = new HashMap<>();
172     params.put("title", title);
173     params.put("movie_id", movie_id);
174 
175     jdbcTemplate.update(sql, params);
176 
177   }
178 
179   // updates a movie director by movie id number
180   public void updateMovieDirectorById(String director, int movie_id) {
181   // @formatter:off
182     String sql = ""
183         + "UPDATE movies "
184         + "SET director = :director "
185         + "WHERE movie_id = :movie_id";
186     // @formatter:on
187 
188     Map<String, Object> params = new HashMap<>();
189     params.put("director", director);
190     params.put("movie_id", movie_id);
191 
192     jdbcTemplate.update(sql, params);
193 
194   }
195 
```

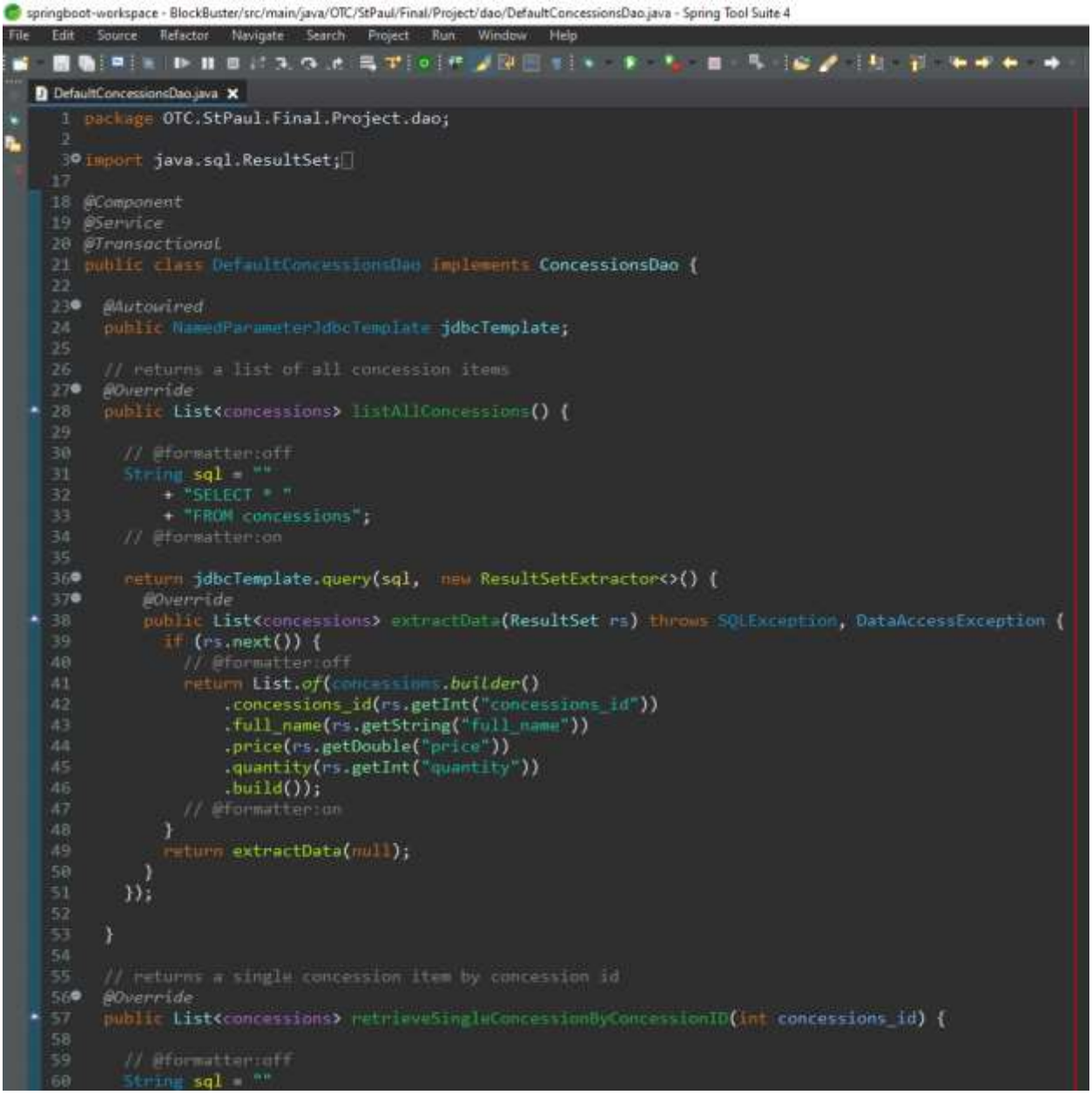
```java
    // updates a movie genre by movie id number
    public void updateMovieGenreById(String genre, int movie_id) {
    // @formatter:off
        String sql = ""
            + "UPDATE movies "
            + "SET genre = :genre "
            + "WHERE movie_id = :movie_id";
        // @formatter:on

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

        jdbcTemplate.update(sql, params);

    }

    // updates a movie rating by movie id number
    public void updateMovieRatingById(String rating, int movie_id) {
    // @formatter:off
        String sql = ""
            + "UPDATE movies "
            + "SET rating = :rating "
            + "WHERE movie_id = :movie_id";
        // @formatter:on

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

        jdbcTemplate.update(sql, params);

    }

    // updates a movie stock by movie id number
    public void updateMovieStockById(Boolean in_stock, int movie_id) {
    // @formatter:off
        String sql = ""
            + "UPDATE movies "
            + "SET in_stock = :in_stock "
            + "WHERE movie_id = :movie_id";
        // @formatter:on

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



```
241     params.put("movie_id", movie_id);
242
243     jdbcTemplate.update(sql, params);
244
245   }
246
247   // updates a movie's location by movie id number
248   public void updateMovieLocationById(int store_location, int movie_id) {
249  // @formatter:off
250     String sql = ""
251         + "UPDATE movies "
252         + "SET store_location = :store_location "
253         + "WHERE movie_id = :movie_id";
254     // @formatter:on
255
256     Map<String, Object> params = new HashMap<>();
257     params.put("store_location", store_location);
258     params.put("movie_id", movie_id);
259
260     jdbcTemplate.update(sql, params);
261   }
262
263 } // last bracket
264
```

Concessions-




springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/dao/DefaultConcessionsDao.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
DefaultConcessionsDao.java
package OTC.StPaul.Final.Project.dao;
import java.sql.ResultSet;
@Component
@Service
@Transactional
public class DefaultConcessionsDao implements ConcessionsDao {
@Autowired
public NamedParameterJdbcTemplate jdbcTemplate;
// returns a list of all concession items
@Override
public List<concessions> listAllConcessions() {
// @formatter:off
String sql = ""
+ "SELECT * "
+ "FROM concessions";
// @formatter:on
return jdbcTemplate.query(sql, new ResultSetExtractor<>() {
@Override
public List<concessions> extractData(ResultSet rs) throws SQLException, DataAccessException {
if (rs.next()) {
// @formatter:off
return List.of(concessions.builder()
.concessions_id(rs.getInt("concessions_id"))
.full_name(rs.getString("full_name"))
.price(rs.getDouble("price"))
.quantity(rs.getInt("quantity"))
.build());
// @formatter:on
}
return extractData(null);
}
});
}
// returns a single concession item by concession id
@Override
public List<concessions> retrieveSingleConcessionByConcessionID(int concessions_id) {
// @formatter:off
String sql = ""

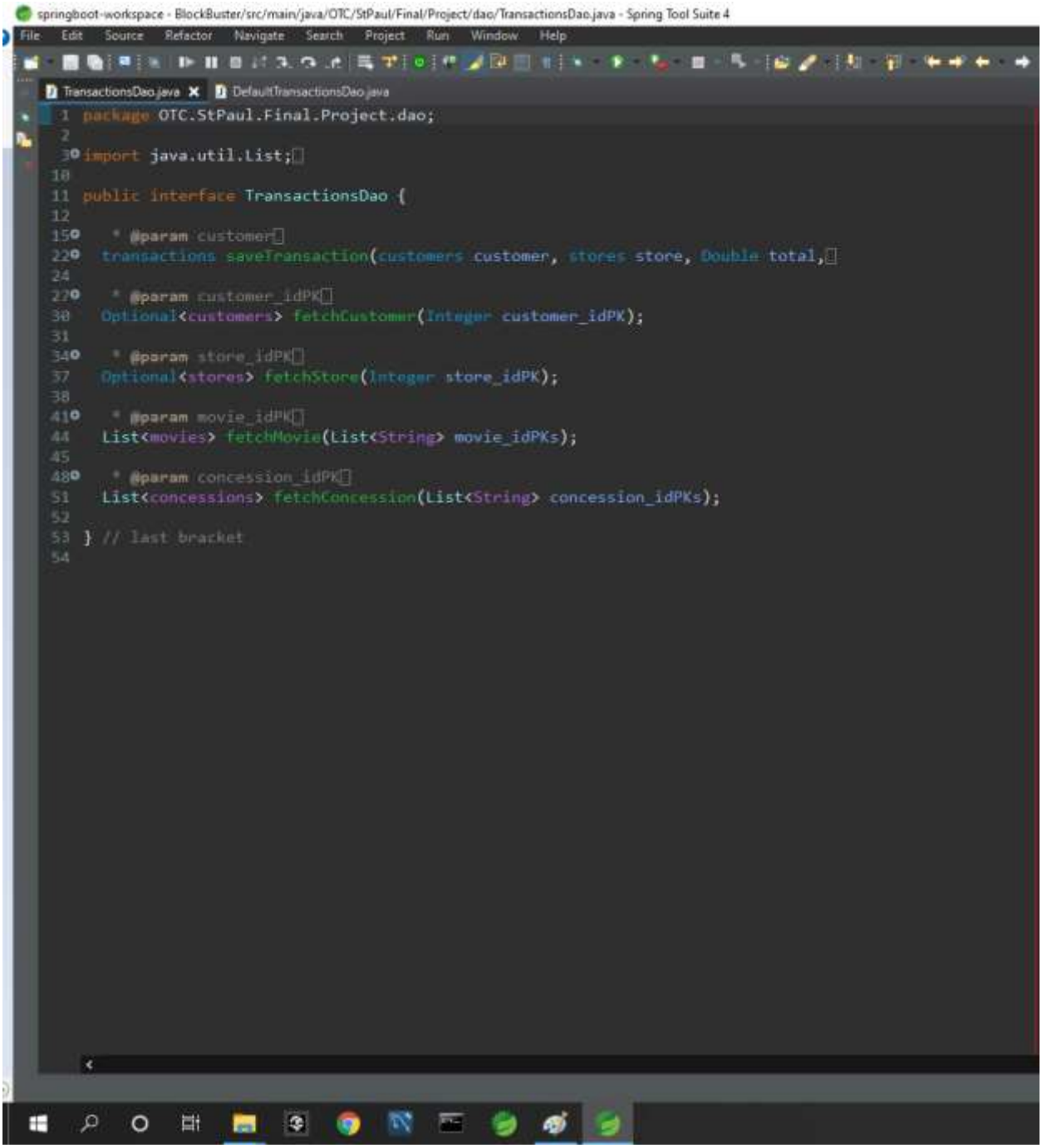
```
61          + "SELECT * "
62          + "FROM concessions "
63          + "WHERE concessions_id = :concessions_id";
64      // @formatter:on
65
66      Map<String, Object> params = new HashMap<>();
67      params.put("concessions_id", concessions_id);
68
69      return jdbcTemplate.query(sql, params, new RowMapper<>() {
70
71        @Override
72        public concessions mapRow(ResultSet rs, int rowNum) throws SQLException {
73
74        // @formatter:off
75          return concessions.builder()
76              .concessions_id(rs.getInt("concessions_id"))
77              .full_name(rs.getString("full_name"))
78              .price(rs.getDouble("price"))
79              .quantity(rs.getInt("quantity"))
80              .build();
81          // @formatter:on
82        }
83      });
84    }
85
86    // returns a single concession item by name
87    @Override
88    public List<concessions> retrieveSingleConcessionByName(String full_name) {
89
90      // @formatter:off
91      String sql = ""
92          + "SELECT * "
93          + "FROM concessions "
94          + "WHERE full_name = :full_name";
95      // @formatter:on
96
97      Map<String, Object> params = new HashMap<>();
98      params.put("full_name", full_name);
99
100     return jdbcTemplate.query(sql, params, new RowMapper<>() {
101
102       @Override
103       public concessions mapRow(ResultSet rs, int rowNum) throws SQLException {
104
105       // @formatter:off
```

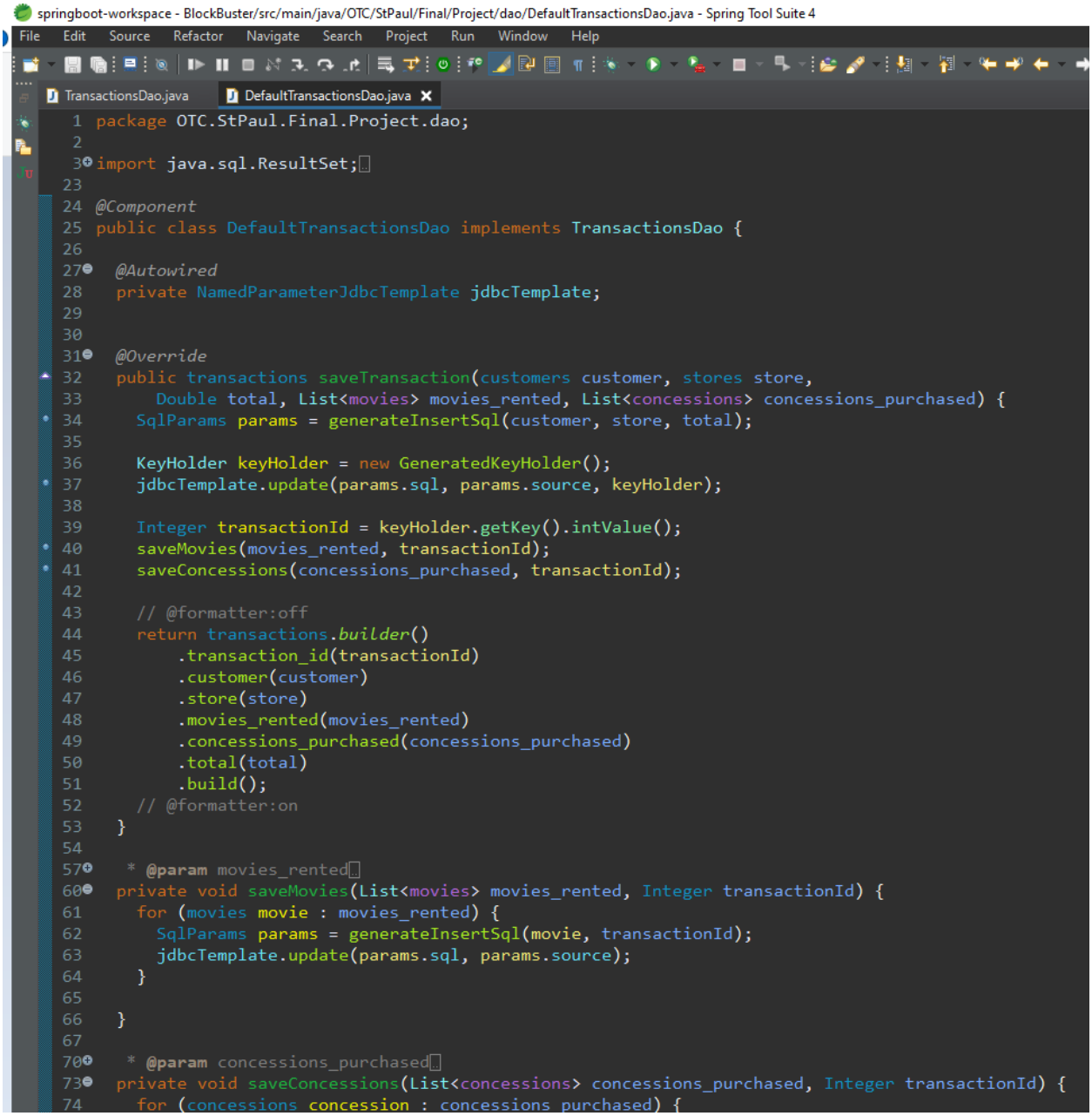
```
106         return concessions.builder()
107             .concessions_id(rs.getInt("concessions_id"))
108             .full_name(rs.getString("full_name"))
109             .price(rs.getDouble("price"))
110             .quantity(rs.getInt("quantity"))
111             .build();
112         // @formatter:on
113       }
114     });
115   }
116
117   @Override
118   public void addConcession(String full_name, double price, int quantity) {
119   // @formatter:off
120     String sql = ""
121         + "INSERT INTO concessions ("
122         + "full_name, price, quantity"
123         + ")VALUES ("
124         + ":full_name, :price, :quantity"
125         + ")";
126     // @formatter:on
127     Map<String, Object> params = new HashMap<>();
128
129     params.put("full_name", full_name);
130     params.put("price", price);
131     params.put("quantity", quantity);
132
133     jdbcTemplate.update(sql, params);
134   }
135
136   // deletes a concessions item by id
137   @Override
138   public void deleteConcessionById(int concessions_id) {
139     // @formatter:off
140     String sql = ""
141         + "DELETE FROM concessions "
142         + "WHERE concessions_id = :concessions_id";
143     // @formatter:on
144
145     Map<String, Object> params = new HashMap<>();
146     params.put("concessions_id", concessions_id);
147
148     jdbcTemplate.update(sql,  params);
149   }
150
```

```java
    // updates a concession name by concession id number
    @Override
    public void updateConcessionNameById(String full_name, int concessions_id) {
   // @formatter:off
      String sql = ""
          + "UPDATE concessions "
          + "SET full_name = :full_name "
          + "WHERE concessions_id = :concessions_id";
      // @formatter:on

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

      jdbcTemplate.update(sql, params);
    }

    // updates a concession price by concession id
    @Override
    public void updateConcessionPriceById(double price, int concessions_id) {
   // @formatter:off
      String sql = ""
          + "UPDATE concessions "
          + "SET price = :price "
          + "WHERE concessions_id = :concessions_id";
      // @formatter:on

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

      jdbcTemplate.update(sql, params);
    }

    @Override
    public void updateConcessionQuantityById(int quantity, int concessions_id) {
   // @formatter:off
      String sql = ""
          + "UPDATE concessions "
          + "SET quantity = :quantity "
          + "WHERE concessions_id = :concessions_id";
      // @formatter:on

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

```java
      params.put("concessions_id", concessions_id);

      jdbcTemplate.update(sql, params);
    }

} // last bracket
```

Transactions-




springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/dao/DefaultTransactionsDao.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
TransactionsDao.java
DefaultTransactionsDao.java
1 package OTC.StPaul.Final.Project.dao;
2
3 import java.sql.ResultSet;
23
24 @Component
25 public class DefaultTransactionsDao implements TransactionsDao {
26
27 @Autowired
28 private NamedParameterJdbcTemplate jdbcTemplate;
29
30
31 @Override
32 public transactions saveTransaction(customers customer, stores store,
33 Double total, List<movies> movies_rented, List<concessions> concessions_purchased) {
34 SqlParams params = generateInsertSql(customer, store, total);
35
36 KeyHolder keyHolder = new GeneratedKeyHolder();
37 jdbcTemplate.update(params.sql, params.source, keyHolder);
38
39 Integer transactionId = keyHolder.getKey().intValue();
40 saveMovies(movies_rented, transactionId);
41 saveConcessions(concessions_purchased, transactionId);
42
43 // @formatter:off
44 return transactions.builder()
45 .transaction_id(transactionId)
46 .customer(customer)
47 .store(store)
48 .movies_rented(movies_rented)
49 .concessions_purchased(concessions_purchased)
50 .total(total)
51 .build();
52 // @formatter:on
53 }
54
57 * @param movies_rented
60 private void saveMovies(List<movies> movies_rented, Integer transactionId) {
61 for (movies movie : movies_rented) {
62 SqlParams params = generateInsertSql(movie, transactionId);
63 jdbcTemplate.update(params.sql, params.source);
64 }
65
66 }
67
70 * @param concessions_purchased
73 private void saveConcessions(List<concessions> concessions_purchased, Integer transactionId) {
74 for (concessions concession : concessions_purchased) {

```
75        SqlParams params = generateInsertSql(concession, transactionId);
76        jdbcTemplate.update(params.sql, params.source);
77      }
78
79    }
80
83     * @param customer
88    private SqlParams generateInsertSql(customers customer, stores store, Double total) {
89      // @formatter:off
90      String sql = ""
91          + "INSERT INTO transactions ("
92          + "customer_idFK, store_idFK, total"
93          + ") VALUES ("
94          + ":customer_idFK, :store_idFK, :total"
95          + ")";
96      // @formatter:on
97
98      SqlParams params = new SqlParams();
99
100     params.sql = sql;
101     params.source.addValue("customer_idFK", customer.getCustomer_idPK());
102     params.source.addValue("store_idFK", store.getStore_idPK());
103     params.source.addValue("total", total);
104
105     return params;
106   }
107
110    * @param movie
114   private SqlParams generateInsertSql(movies movie, Integer transactionId) {
115     SqlParams params = new SqlParams();
116
117     // @formatter:off
118     params.sql = ""
119         + "INSERT INTO transaction_movies ("
120         + "movie_idFK, transaction_idFK"
121         + ") VALUES ("
122         + ":movie_idFK, :transaction_idFK"
123         + ")";
124     // @formatter:on
125
126     params.source.addValue("movie_idFK", movie.getMovie_id());
127     params.source.addValue("transaction_idFK", transactionId);
128
129     return params;
130   }
```

```
131
134     * @param concession
138   private SqlParams generateInsertSql(concessions concession, Integer transactionId) {
139     SqlParams params = new SqlParams();
140
141     // @formatter:off
142     params.sql = ""
143         + "INSERT INTO transaction_concessions ("
144         + "concessions_idFK, transaction_idFK"
145         + ") VALUES ("
146         + ":concessions_idFK, :transaction_idFK"
147         + ")";
148     // @formatter:on
149
150     params.source.addValue("concessions_idFK", concession.getConcessions_id());
151     params.source.addValue("transaction_idFK", transactionId);
152
153     return params;
154   }
155
156   @Override
157   public Optional<customers> fetchCustomer(Integer customer_idPK) {
158     // @formatter:off
159     String sql = ""
160         + "SELECT * "
161         + "FROM customers "
162         + "WHERE customer_idPK = :customer_idPK";
163     // @formatter:on
164
165     Map<String, Object> params = new HashMap<>();
166     params.put("customer_idPK", customer_idPK);
167
168     return Optional.ofNullable(jdbcTemplate.query(sql, params, new CustomerResultSetExtractor()));
169   }
170
171   public Optional<stores> fetchStore(Integer store_idPK) {
172     // @formatter:off
173     String sql = ""
174         + "SELECT * "
175         + "FROM stores "
176         + "WHERE store_idPK = :store_idPK";
177     // @formatter:on
178
179     Map<String, Object> params = new HashMap<>();
180     params.put("store_idPK", store_idPK);
```

```
181
182     return Optional.ofNullable(jdbcTemplate.query(sql, params, new StoreResultSetExtractor()));
183   }
184
185   public List<movies> fetchMovie(List<String> movie_ids) {
186     if (movie_ids.isEmpty()) {
187       return new LinkedList<>();
188     }
189
190     Map<String, Object> params = new HashMap<>();
191
192     // @formatter:off
193     String sql = ""
194         + "SELECT * "
195         + "FROM movies "
196         + "WHERE movie_id IN(";
197     // @formatter:on
198
199     for (int index = 0; index < movie_ids.size(); index++) {
200       String key = "movie_" + index;
201       sql += ":" + key + ", ";
202       params.put(key, movie_ids.get(index));
203     }
204
205     sql = sql.substring(0, sql.length() - 2);
206     sql += ")";
207
208     return jdbcTemplate.query(sql, params, new RowMapper<movies>() {
209       @Override
210       public movies mapRow(ResultSet rs, int rowNum) throws SQLException {
211         // @formatter:off
212         return movies.builder()
213             .movie_id(rs.getInt("movie_id"))
214             .title(rs.getString("title"))
215             .build();
216         // @formatter:on
217       }
218     });
219   }
220
221   public List<concessions> fetchConcession(List<String> concessions_ids) {
222     if (concessions_ids.isEmpty()) {
223       return new LinkedList<>();
224     }
225
```

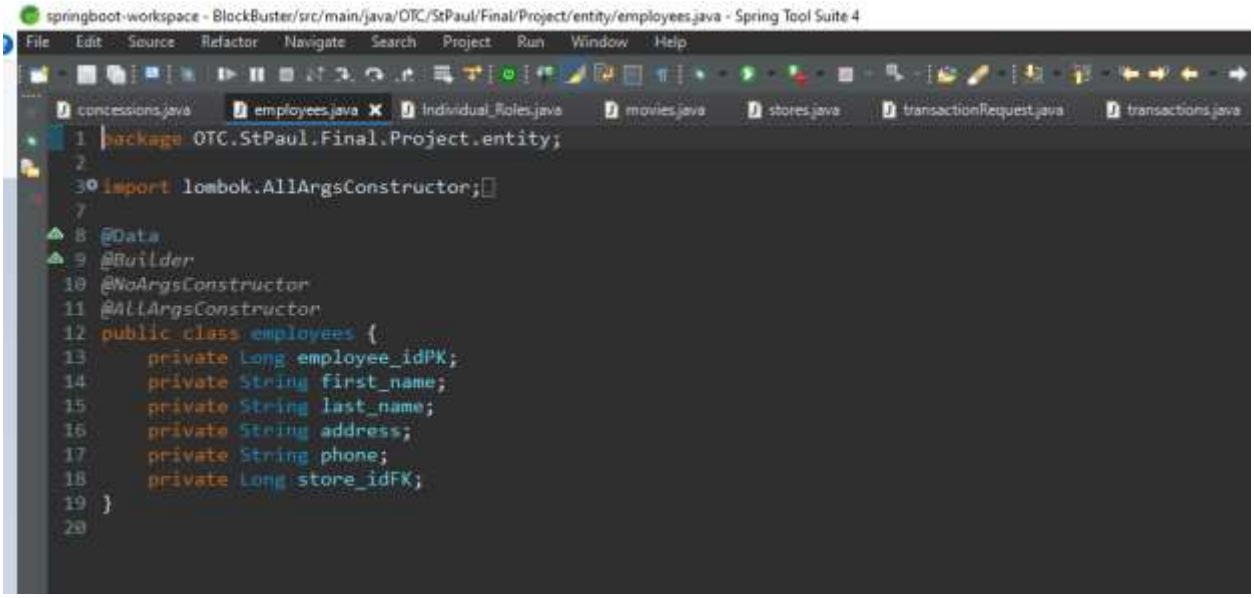
```java
    Map<String, Object> params = new HashMap<>();

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

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

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

    return jdbcTemplate.query(sql, params, new RowMapper<concessions>() {
      @Override
      public concessions mapRow(ResultSet rs, int rowNum) throws SQLException {
       // @formatter:off
        return concessions.builder()
            .concessions_id(rs.getInt("concessions_id"))
            .full_name(rs.getString("full_name"))
            .price(rs.getDouble("price"))
            .build();
       // @formatter:on
      }
    });
  }

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

      // @formatter:off
      return customers.builder()
          .customer_idPK(rs.getInt("customer_idPK"))
          .first_name(rs.getString("first_name"))
          .last_name(rs.getString("last_name"))
          .build();
      // @formatter:on
    }
```

```
271   }
272
273   class StoreResultSetExtractor implements ResultSetExtractor<stores> {
274     @Override
275     public stores extractData(ResultSet rs) throws SQLException {
276       rs.next();
277
278       // @formatter:off
279       return stores.builder()
280           .store_idPK(rs.getInt("store_idPK"))
281           .address(rs.getString("address"))
282           .phone(rs.getString("phone"))
283           .build();
284       // @formatter:on
285     }
286   }
287
288   class MovieResultSetExtractor implements ResultSetExtractor<movies> {
289     @Override
290     public movies extractData(ResultSet rs) throws SQLException {
291       rs.next();
292
293       // @formatter:off
294       return movies.builder()
295           .title(rs.getString("title"))
296           .build();
297       // @formatter:on
298     }
299   }
300
301   class ConcessionResultSetExtractor implements ResultSetExtractor<concessions> {
302     @Override
303     public concessions extractData(ResultSet rs) throws SQLException {
304       rs.next();
305
306       // @formatter:off
307       return concessions.builder()
308           .full_name(rs.getString("full_name"))
309           .price(rs.getDouble("price"))
310           .build();
311       // @formatter:on
312     }
313   }
314
315   class SqlParams {
```

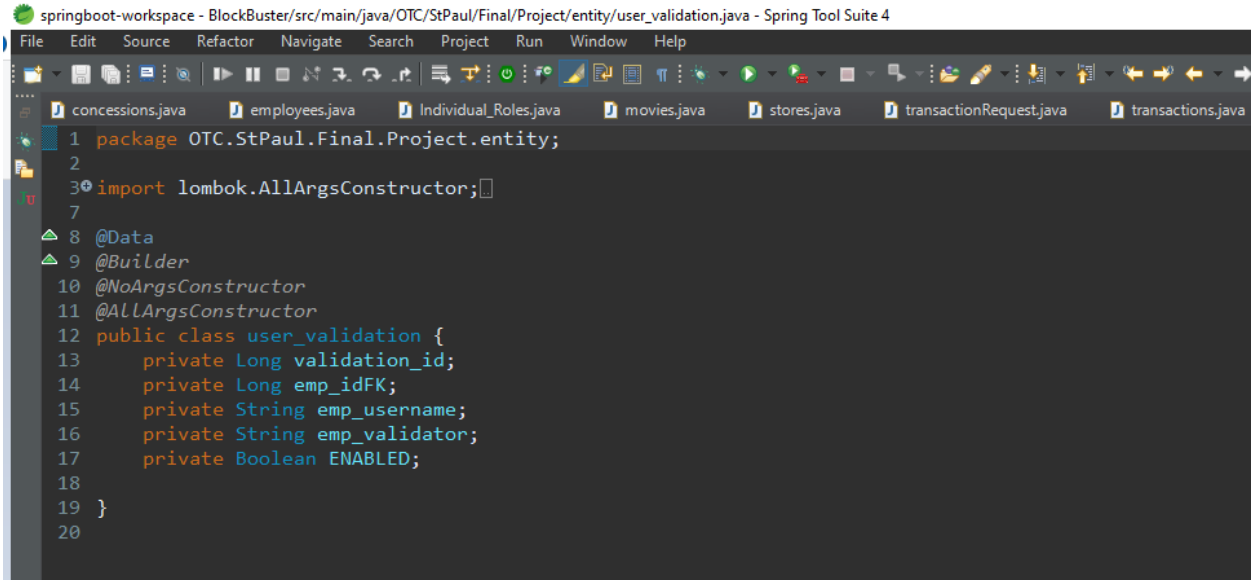
```
316     String sql;
317     MapSqlParameterSource source = new MapSqlParameterSource();
318   }
319
320 } // last bracket
```

Entity:

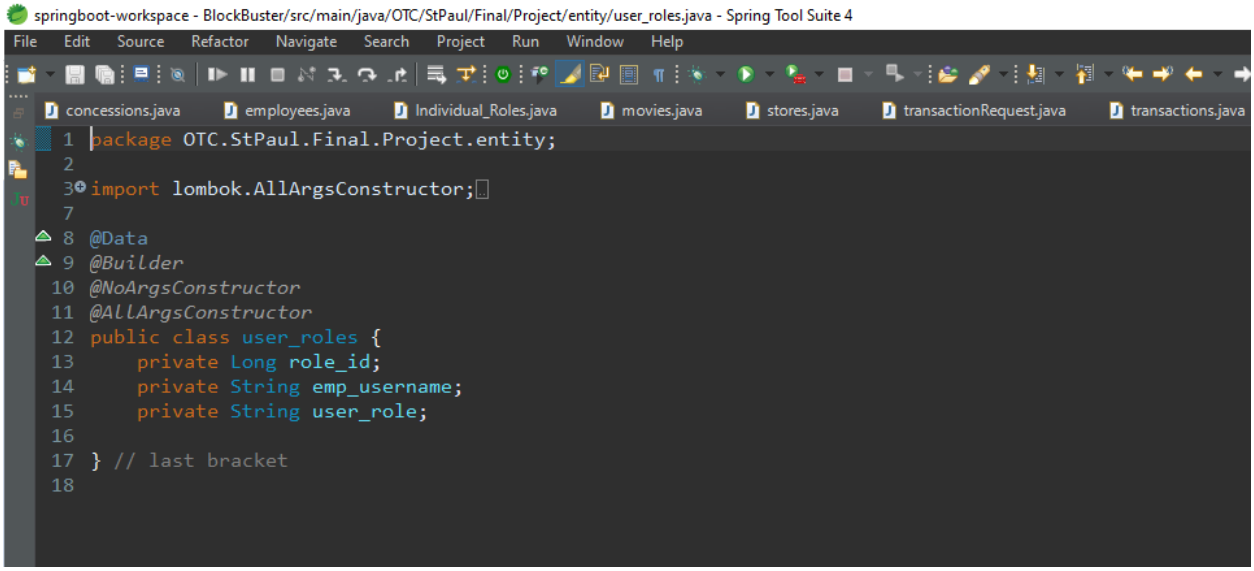
Employees-

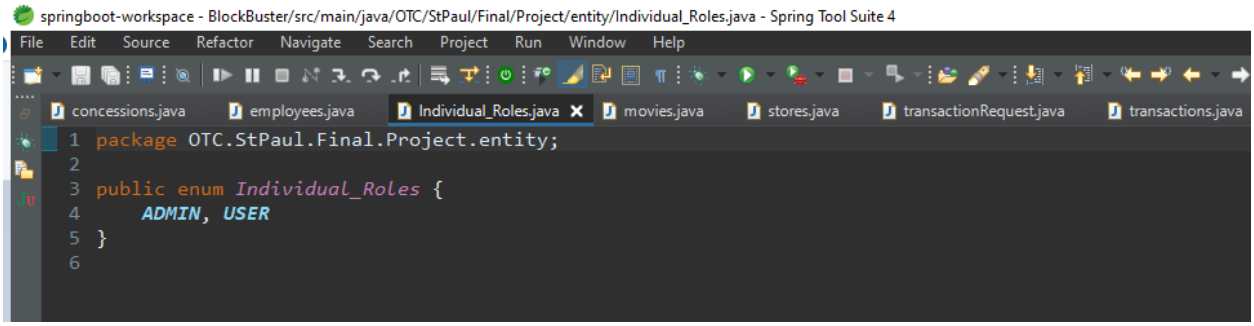


User Validation-

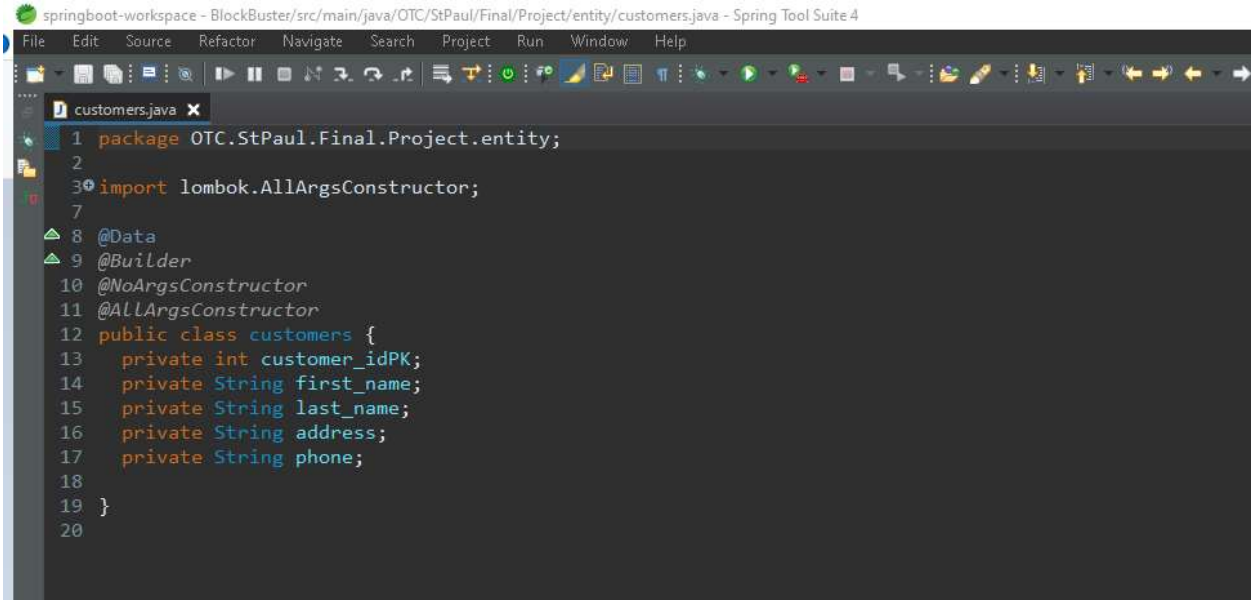


User Roles –

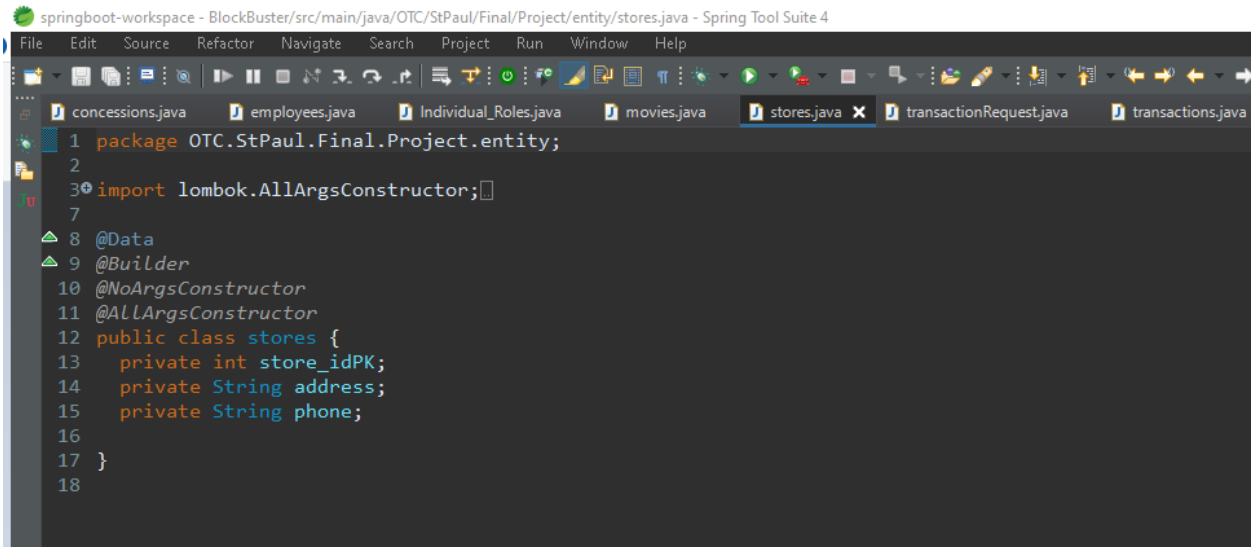




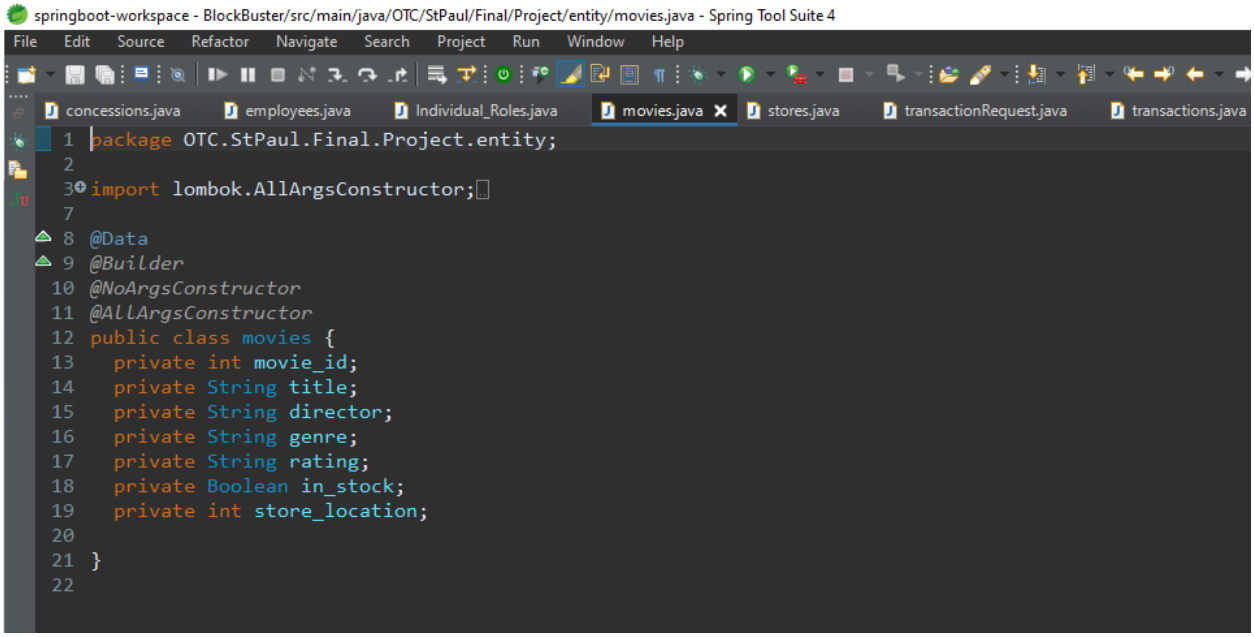
Customers –



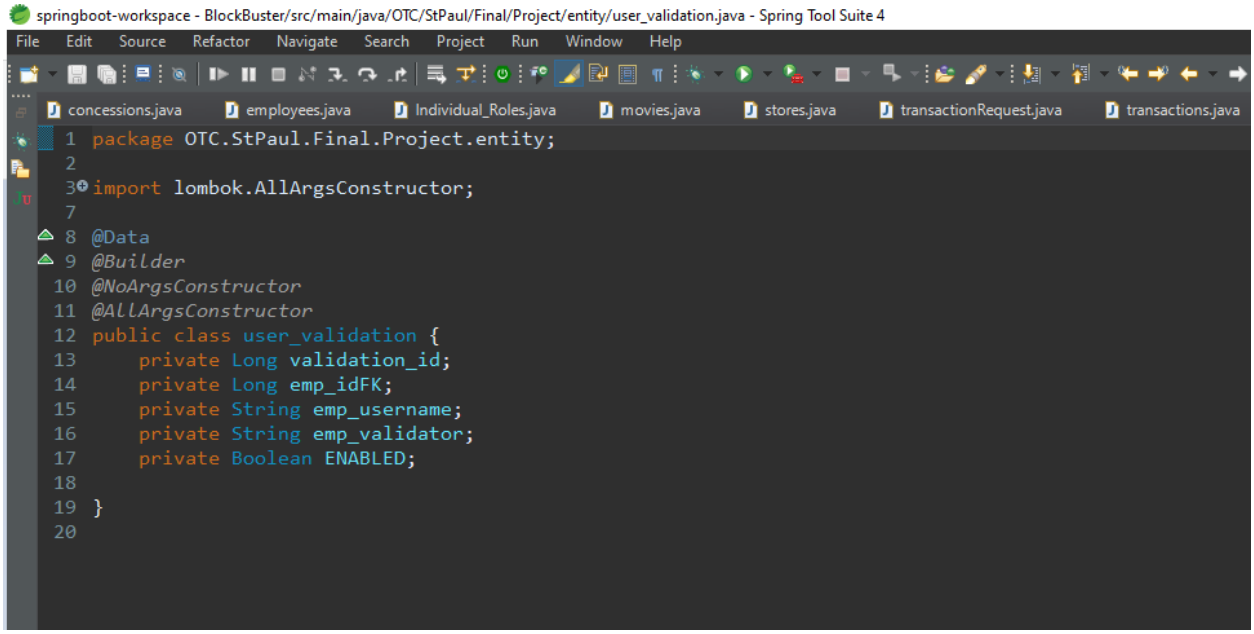
Stores-



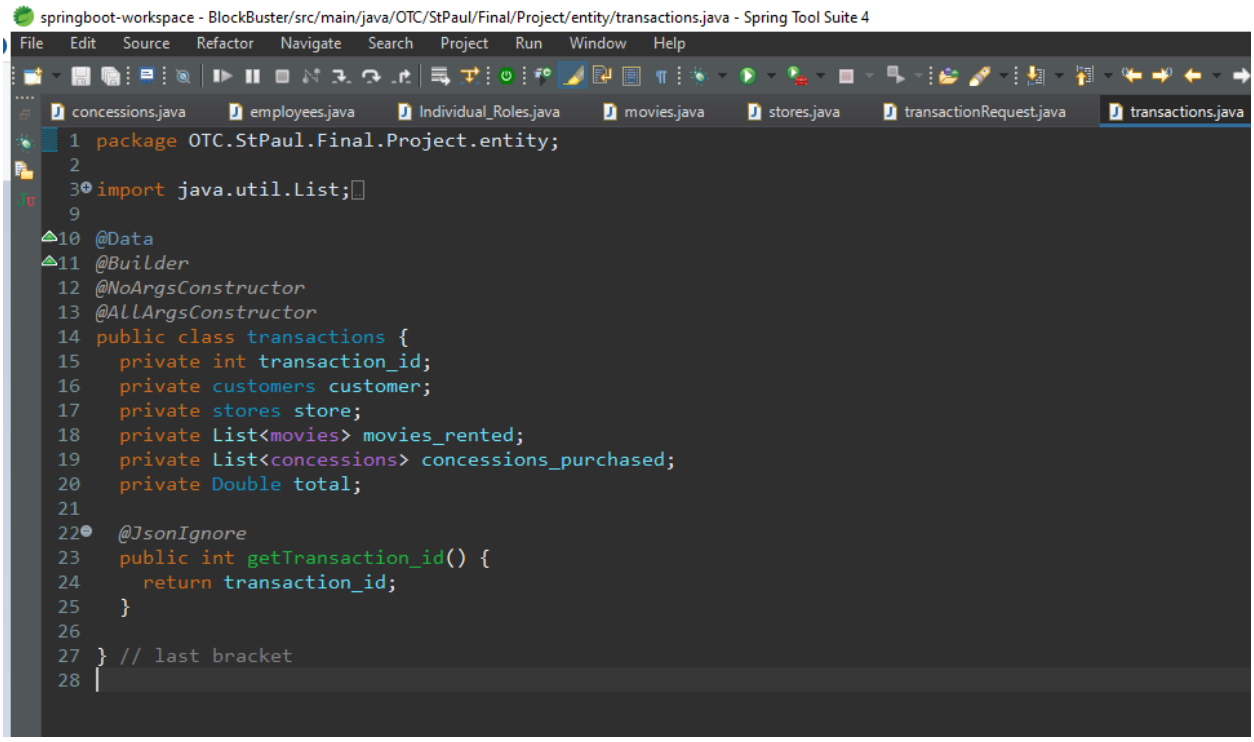
Movies-

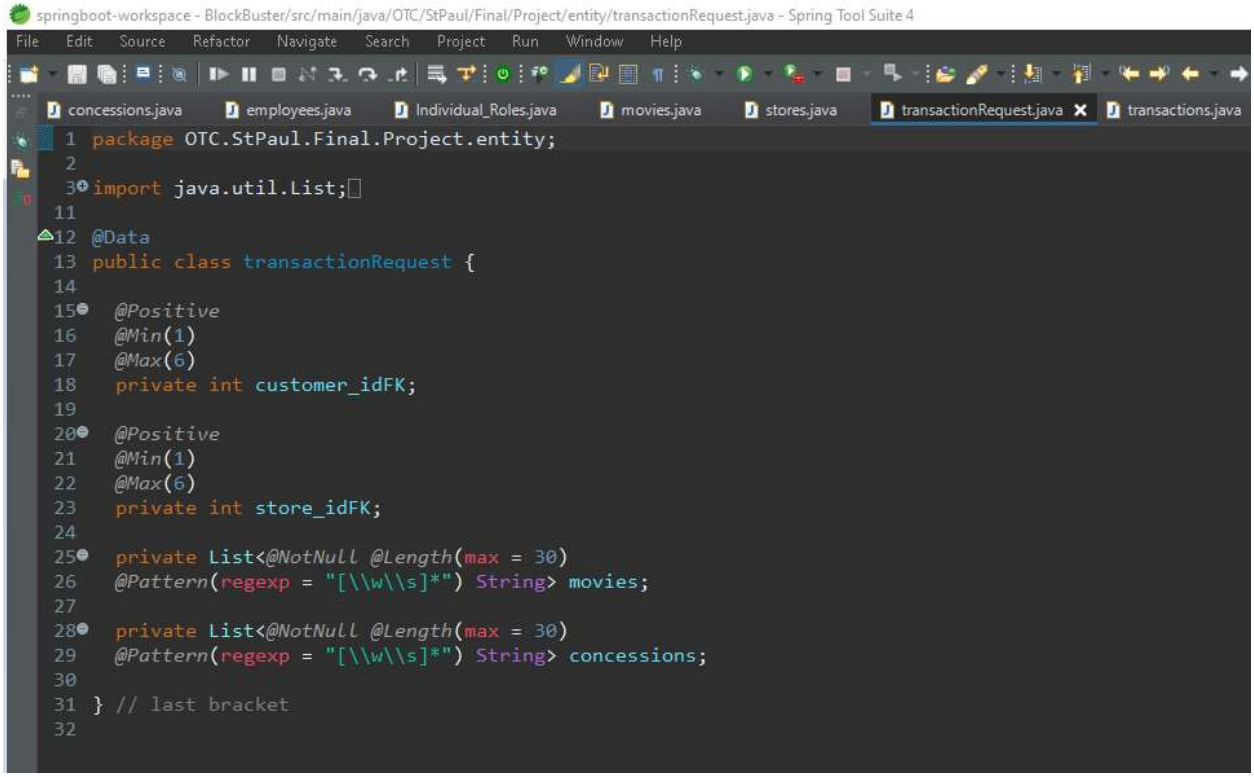


Concessions-



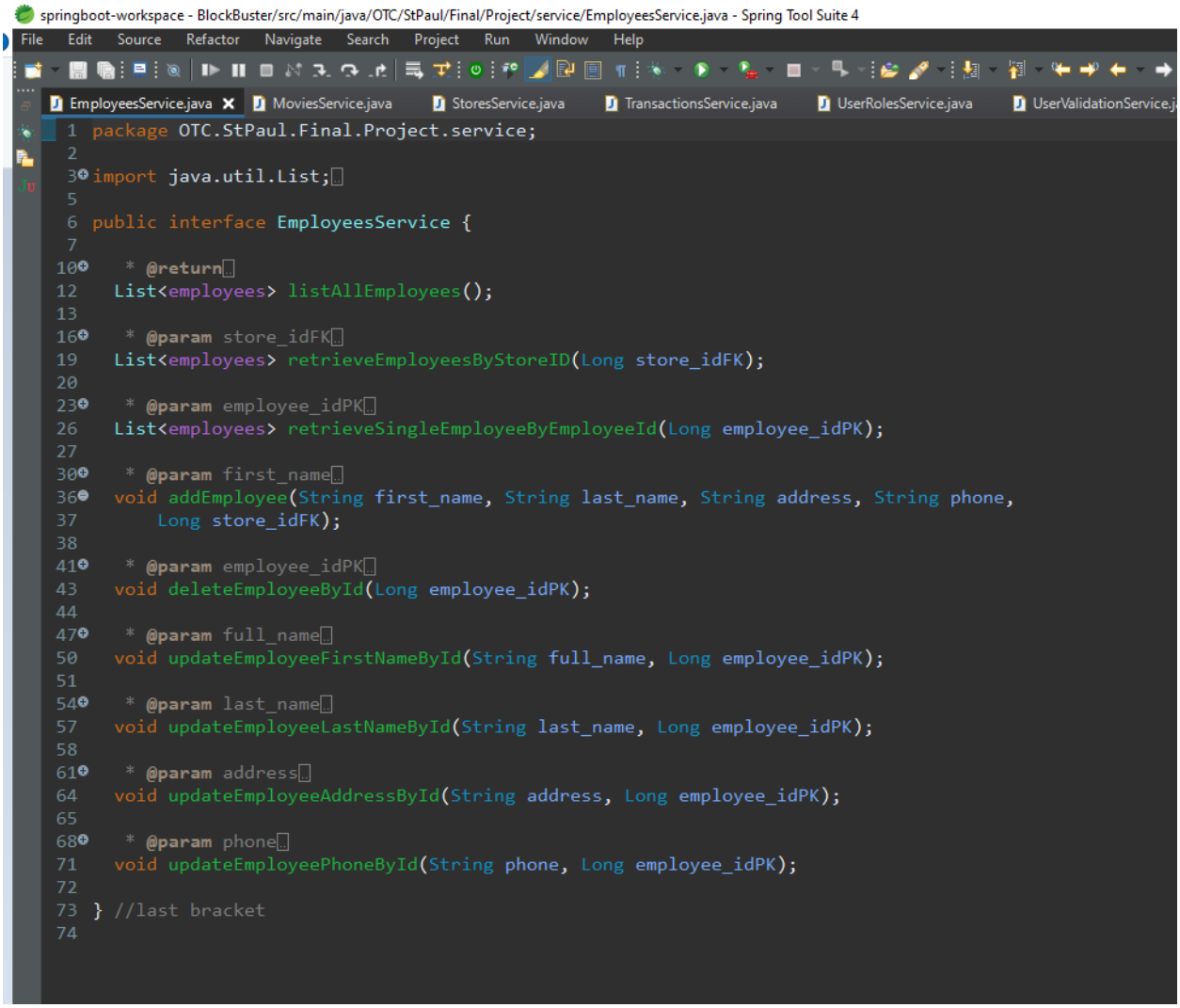
Transactions-

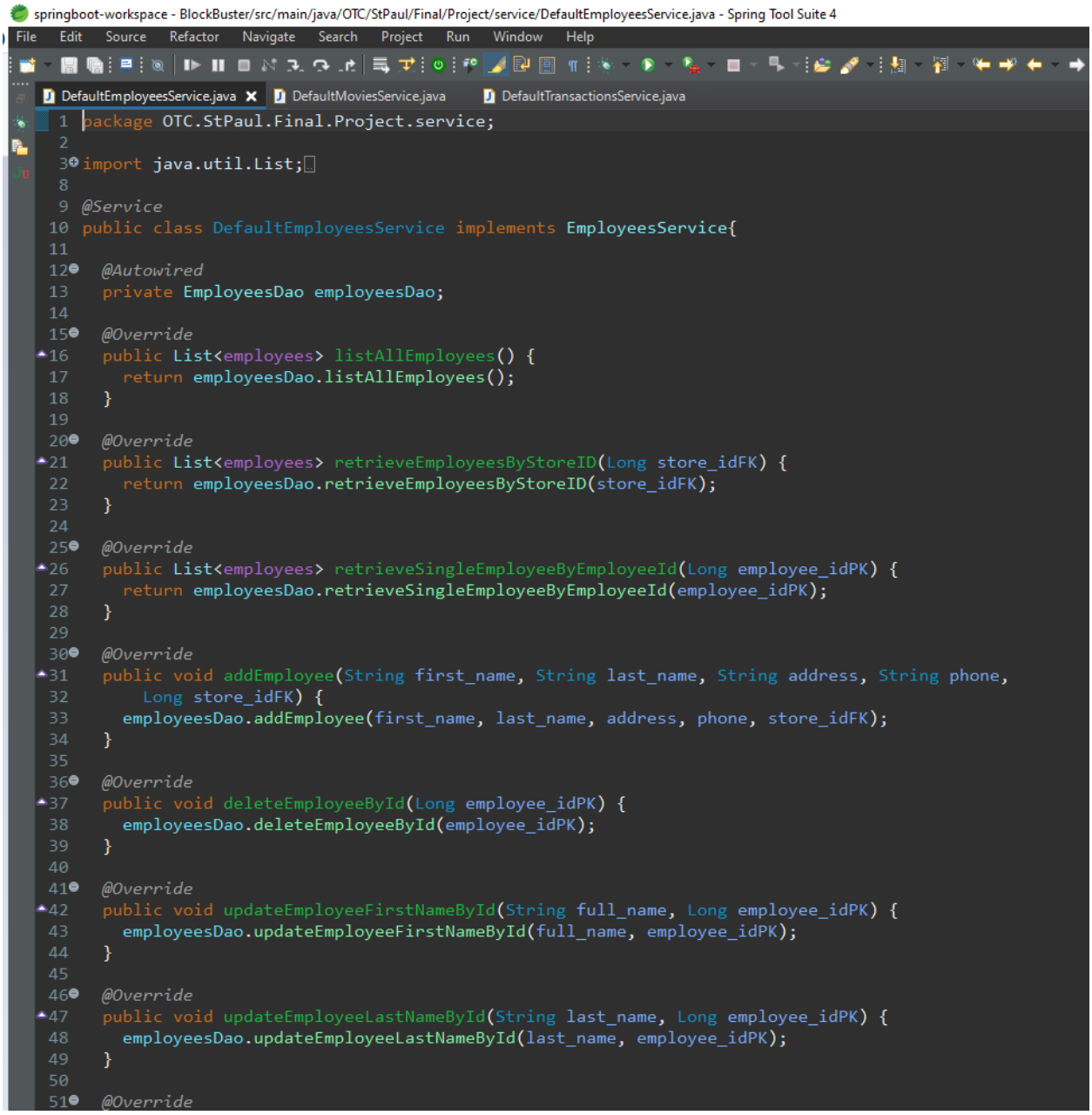




Service

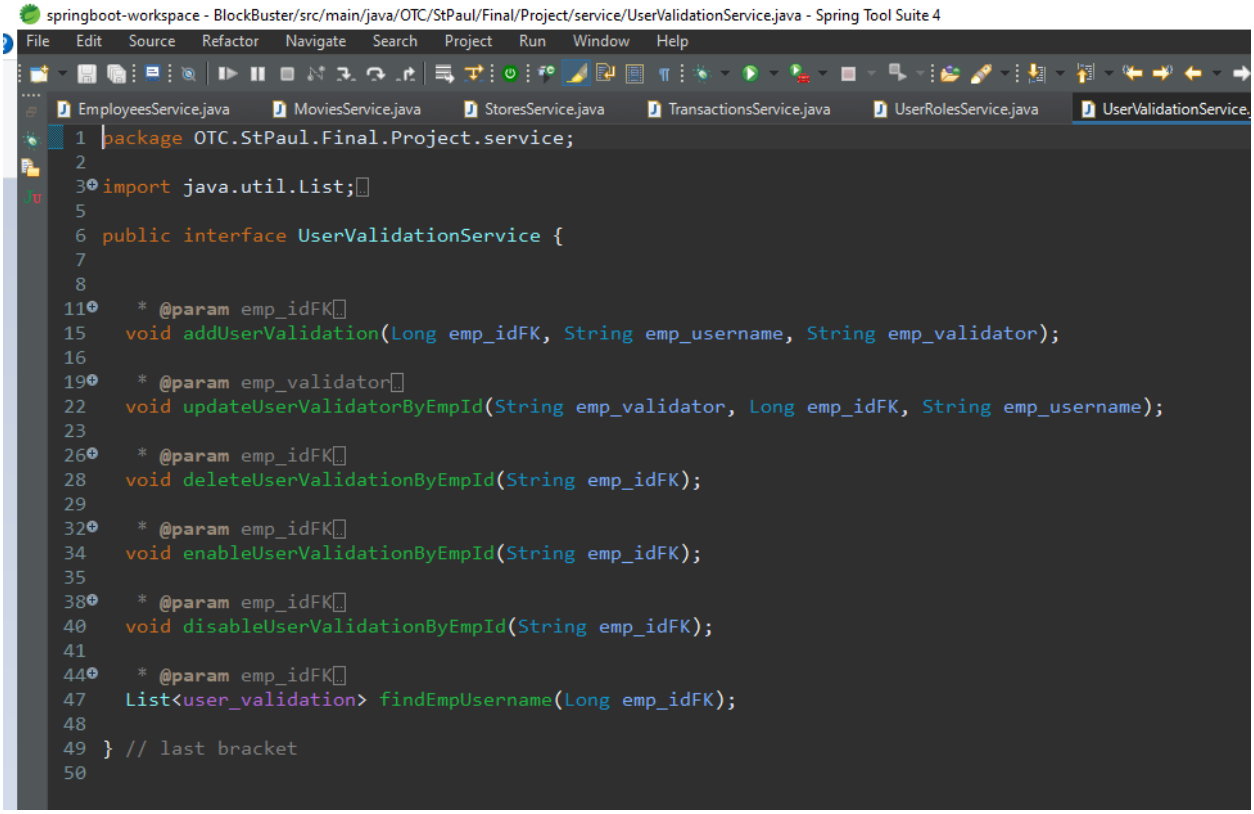
Employees-

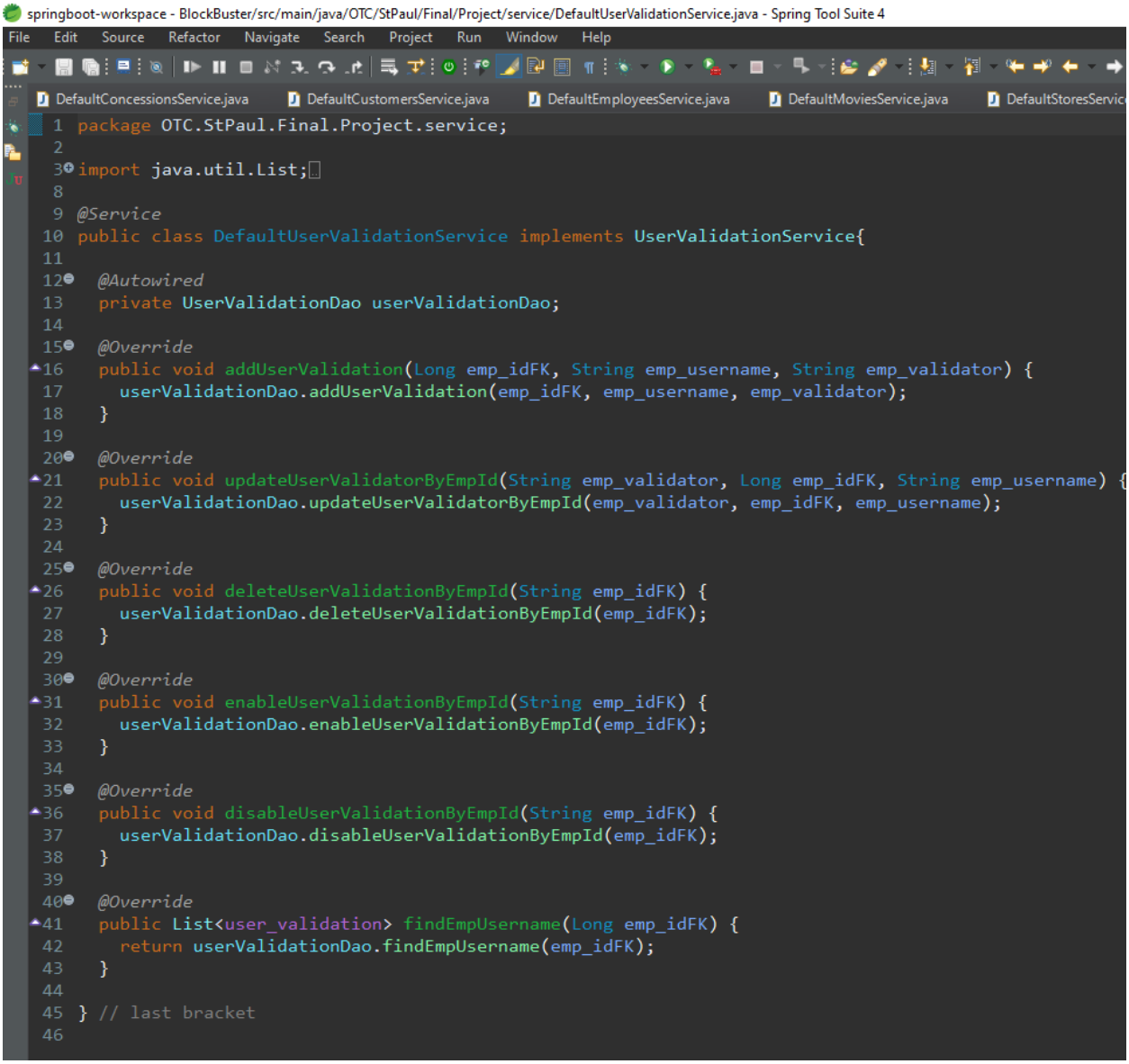




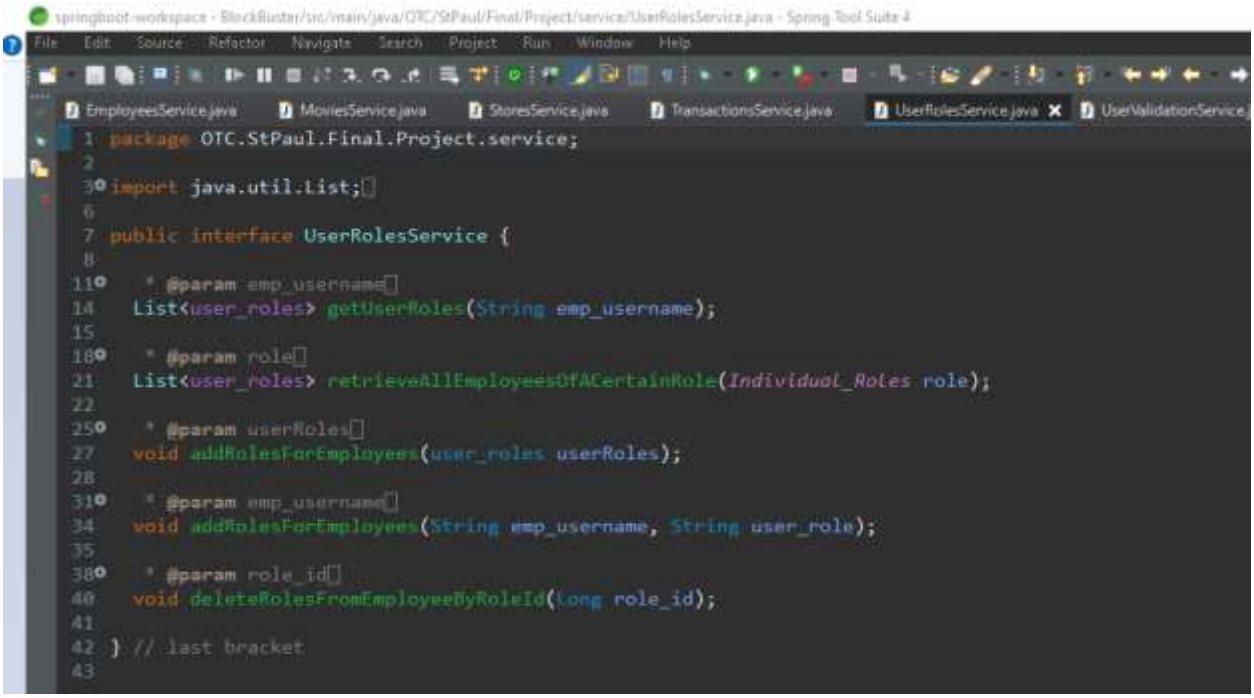

```
52   public void updateEmployeeAddressById(String address, Long employee_idPK) {
53     employeesDao.updateEmployeeAddressById(address, employee_idPK);
54   }
55
56   @Override
57   public void updateEmployeePhoneById(String phone, Long employee_idPK) {
58     employeesDao.updateEmployeePhoneById(phone, employee_idPK);
59   }
60
61 } // last bracket
62
```

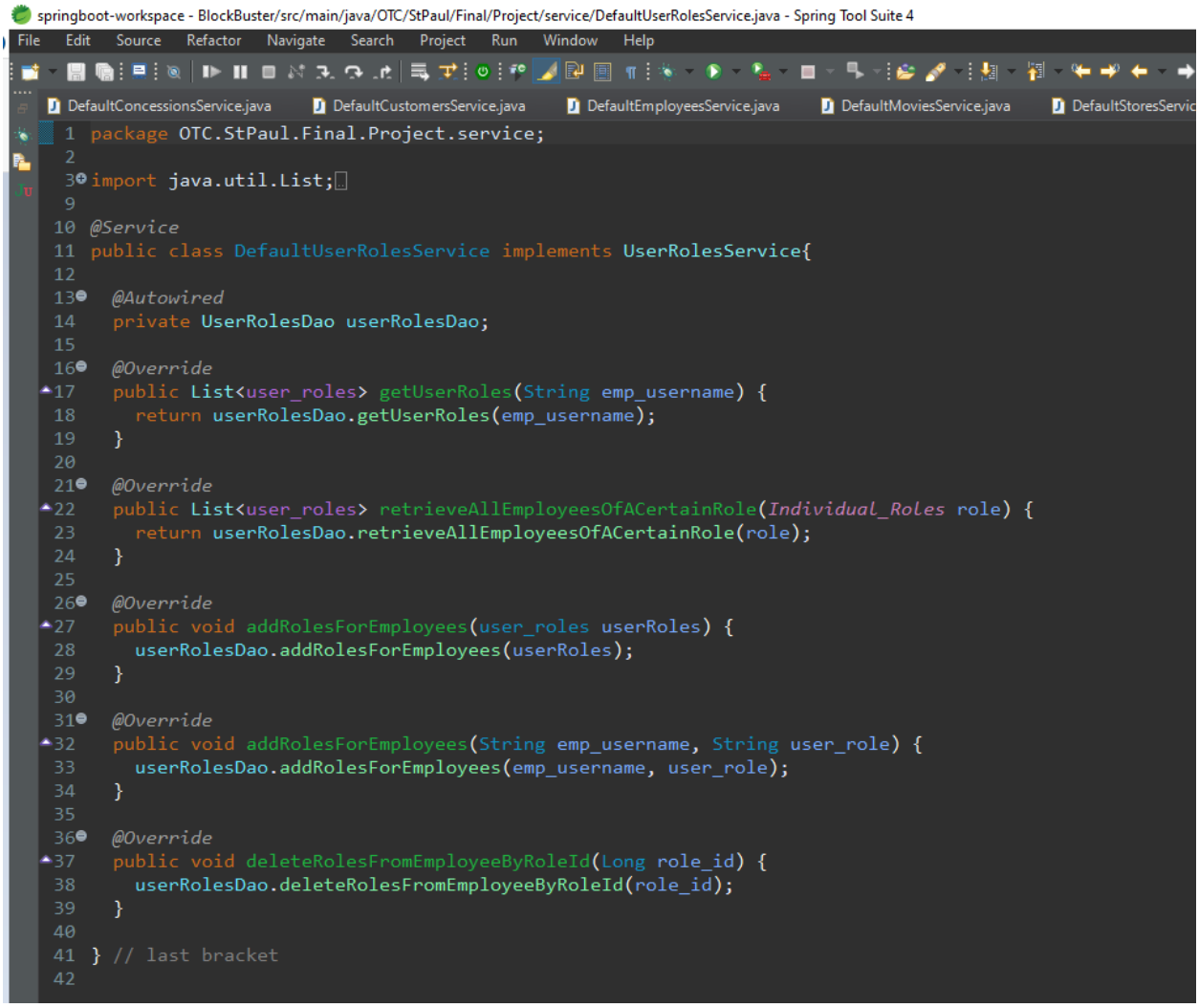
User Validation-



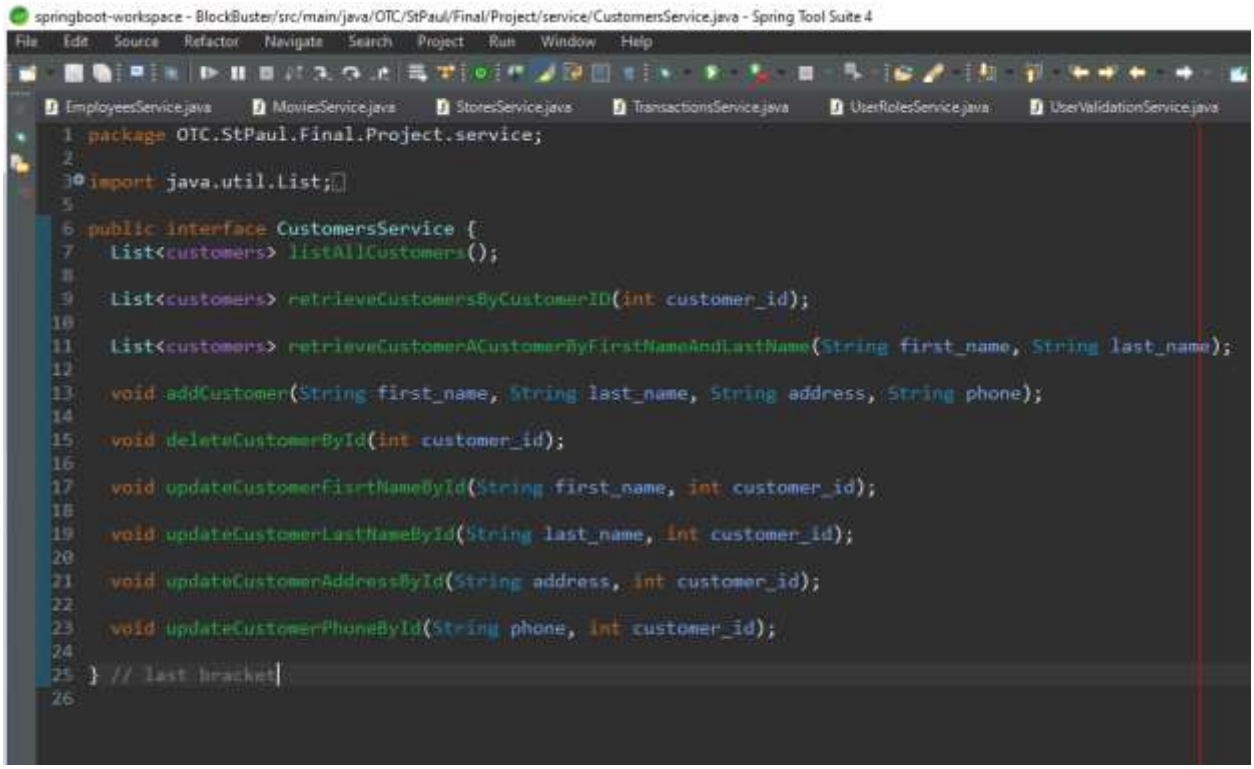

springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/service/DefaultUserValidationService.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
DefaultConcessionsService.java
DefaultCustomersService.java
DefaultEmployeesService.java
DefaultMoviesService.java
DefaultStoresServic
1 package OTC.StPaul.Final.Project.service;
2
3 import java.util.List;
8
9 @Service
10 public class DefaultUserValidationService implements UserValidationService{
11
12 @Autowired
13 private UserValidationDao userValidationDao;
14
15 @Override
16 public void addUserValidation(Long emp_idFK, String emp_username, String emp_validator) {
17 userValidationDao.addUserValidation(emp_idFK, emp_username, emp_validator);
18 }
19
20 @Override
21 public void updateUserValidatorByEmpId(String emp_validator, Long emp_idFK, String emp_username) {
22 userValidationDao.updateUserValidatorByEmpId(emp_validator, emp_idFK, emp_username);
23 }
24
25 @Override
26 public void deleteUserValidationByEmpId(String emp_idFK) {
27 userValidationDao.deleteUserValidationByEmpId(emp_idFK);
28 }
29
30 @Override
31 public void enableUserValidationByEmpId(String emp_idFK) {
32 userValidationDao.enableUserValidationByEmpId(emp_idFK);
33 }
34
35 @Override
36 public void disableUserValidationByEmpId(String emp_idFK) {
37 userValidationDao.disableUserValidationByEmpId(emp_idFK);
38 }
39
40 @Override
41 public List<user_validation> findEmpUsername(Long emp_idFK) {
42 return userValidationDao.findEmpUsername(emp_idFK);
43 }
44
45 } // last bracket
46

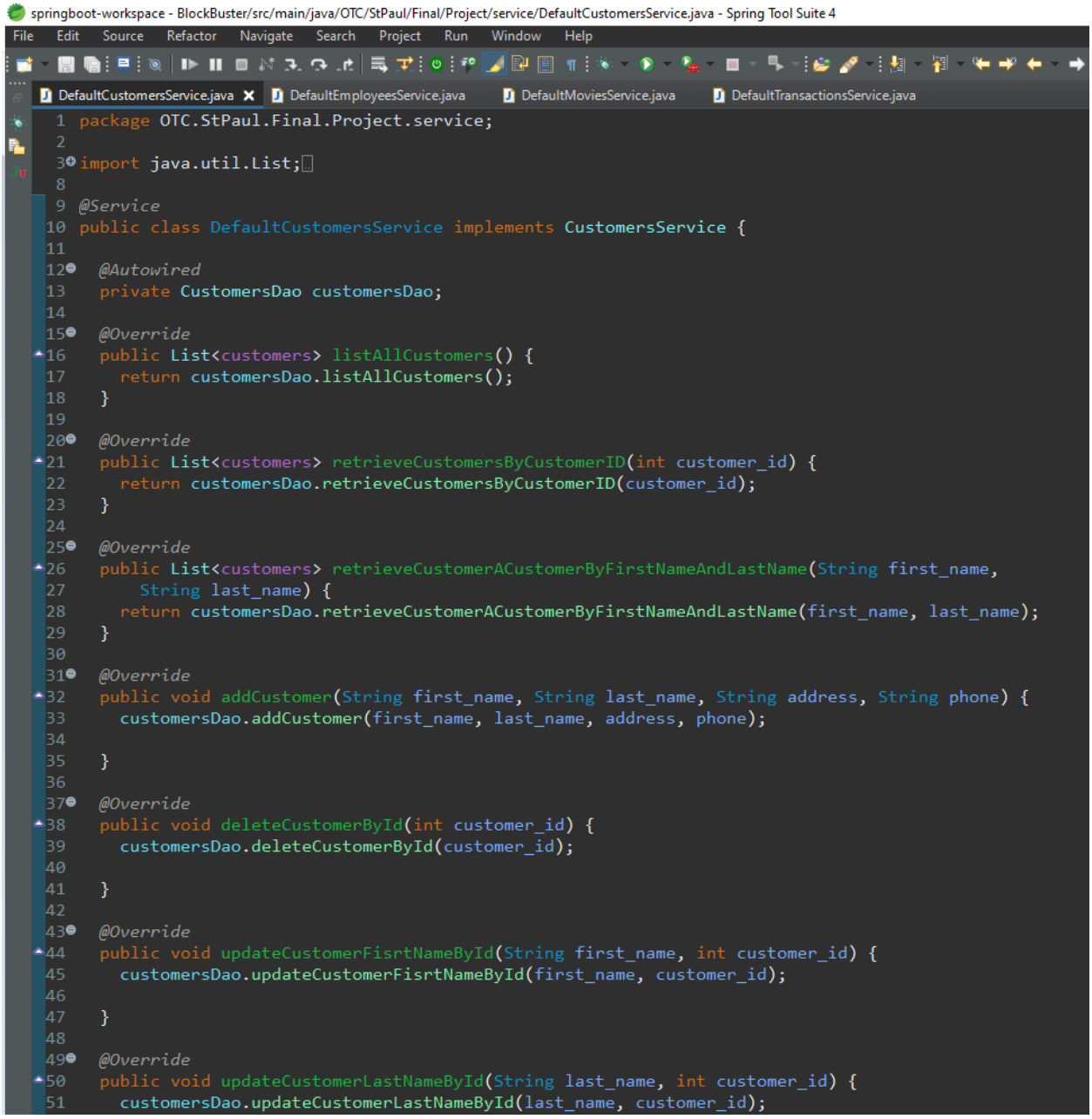
User Roles –


springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/service/UserRolesService.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
EmployeesService.java
MoviesService.java
StoresService.java
TransactionsService.java
UserRolesService.java
UserValidationService.j
1 package OTC.StPaul.Final.Project.service;
2
3 import java.util.List;
6
7 public interface UserRolesService {
8
11 * @param emp_username
14 List<user_roles> getUserRoles(String emp_username);
15
18 * @param role
21 List<user_roles> retrieveAllEmployeesOfACertainRole(Individual_Roles role);
22
25 * @param userRoles
27 void addRolesForEmployees(user_roles userRoles);
28
31 * @param emp_username
34 void addRolesForEmployees(String emp_username, String user_role);
35
38 * @param role_id
40 void deleteRolesFromEmployeeByRoleId(Long role_id);
41
42 } // last bracket
43


springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/service/DefaultUserRolesService.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
DefaultConcessionsService.java
DefaultCustomersService.java
DefaultEmployeesService.java
DefaultMoviesService.java
DefaultStoresServic
1 package OTC.StPaul.Final.Project.service;
2
3 import java.util.List;
9
10 @Service
11 public class DefaultUserRolesService implements UserRolesService{
12
13 @Autowired
14 private UserRolesDao userRolesDao;
15
16 @Override
17 public List<user_roles> getUserRoles(String emp_username) {
18 return userRolesDao.getUserRoles(emp_username);
19 }
20
21 @Override
22 public List<user_roles> retrieveAllEmployeesOfACertainRole(Individual_Roles role) {
23 return userRolesDao.retrieveAllEmployeesOfACertainRole(role);
24 }
25
26 @Override
27 public void addRolesForEmployees(user_roles userRoles) {
28 userRolesDao.addRolesForEmployees(userRoles);
29 }
30
31 @Override
32 public void addRolesForEmployees(String emp_username, String user_role) {
33 userRolesDao.addRolesForEmployees(emp_username, user_role);
34 }
35
36 @Override
37 public void deleteRolesFromEmployeeByRoleId(Long role_id) {
38 userRolesDao.deleteRolesFromEmployeeByRoleId(role_id);
39 }
40
41 } // last bracket
42

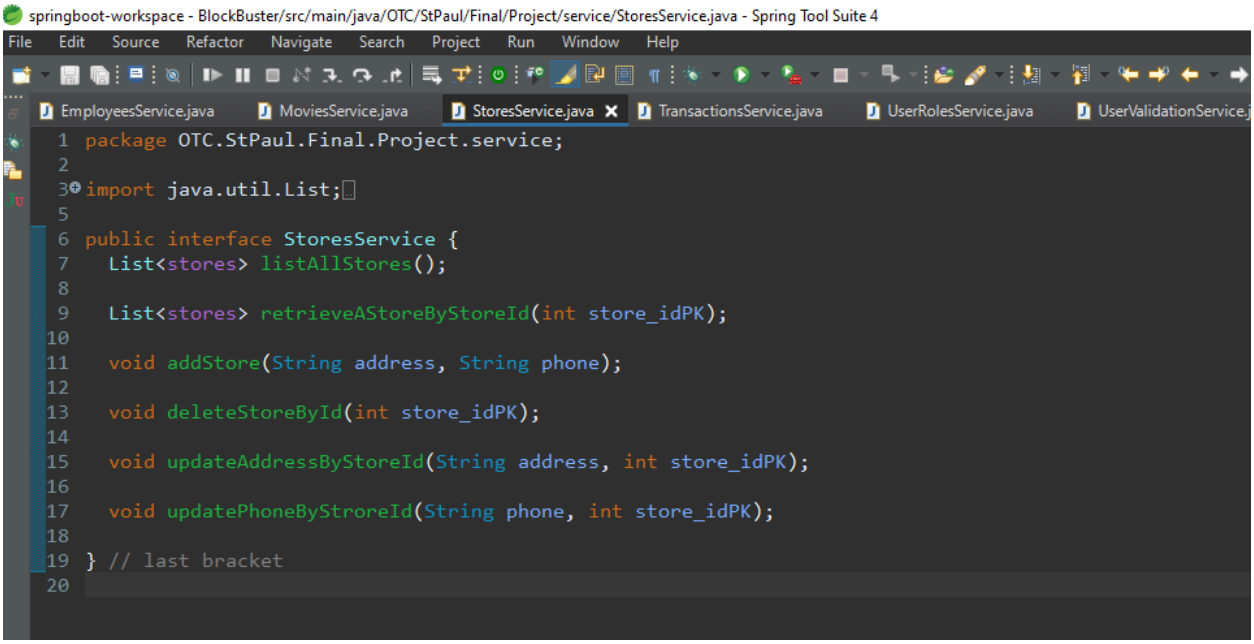
Customers –

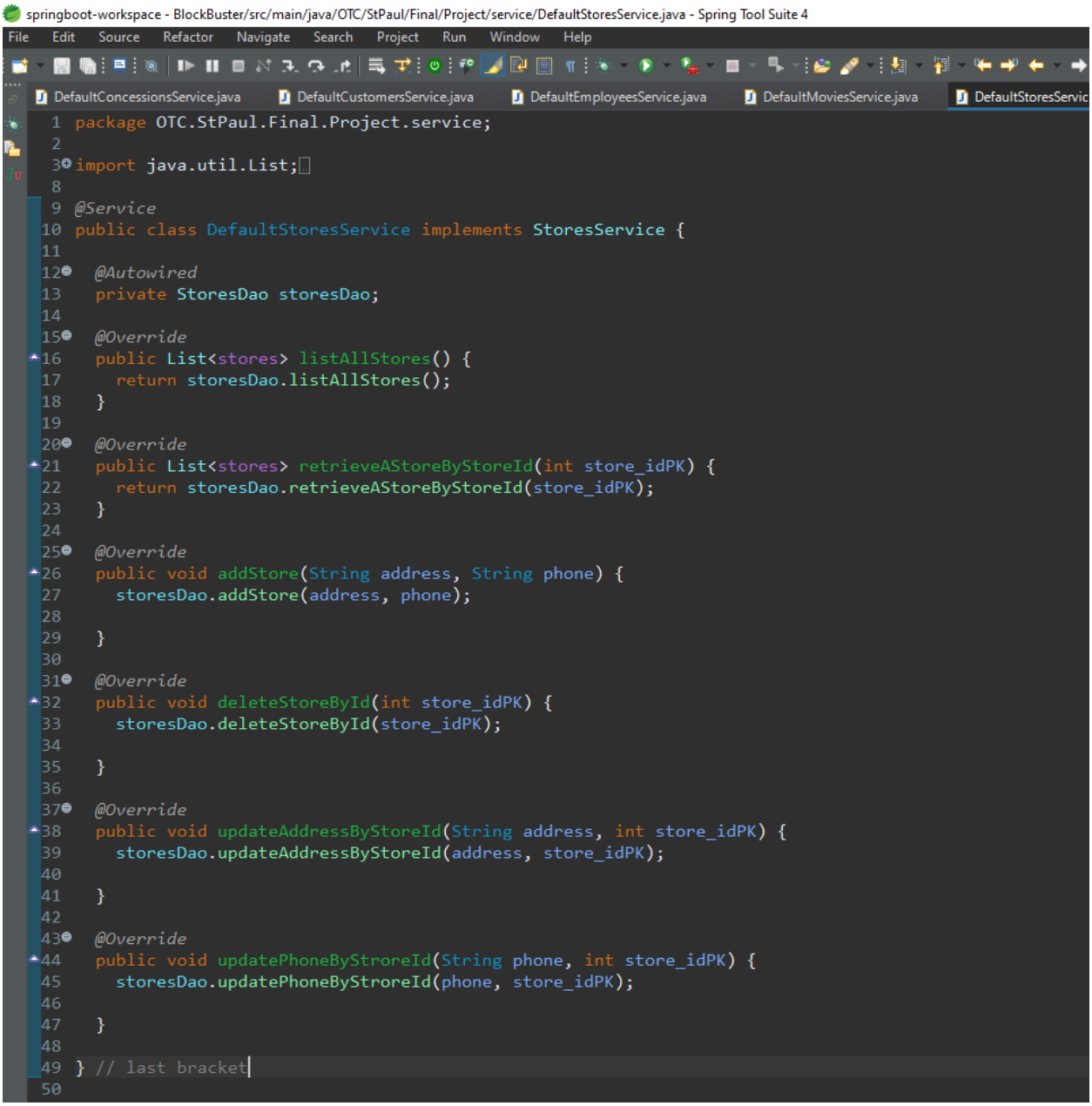

springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/service/CustomersService.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
EmployeesService.java MoviesService.java StoresService.java TransactionsService.java UserRolesService.java UserValidationService.java
1 package OTC.StPaul.Final.Project.service;
2
3 import java.util.List;
5
6 public interface CustomersService {
7 List<customers> listAllCustomers();
8
9 List<customers> retrieveCustomersByCustomerID(int customer_id);
10
11 List<customers> retrieveCustomerACustomerByFirstNameAndLastName(String first_name, String last_name);
12
13 void addCustomer(String first_name, String last_name, String address, String phone);
14
15 void deleteCustomerById(int customer_id);
16
17 void updateCustomerFisrtNameById(String first_name, int customer_id);
18
19 void updateCustomerLastNameById(String last_name, int customer_id);
20
21 void updateCustomerAddressById(String address, int customer_id);
22
23 void updateCustomerPhoneById(String phone, int customer_id);
24
25 } // last bracket
26


springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/service/DefaultCustomersService.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
DefaultCustomersService.java
DefaultEmployeesService.java
DefaultMoviesService.java
DefaultTransactionsService.java
1 package OTC.StPaul.Final.Project.service;
2
3 import java.util.List;
8
9 @Service
10 public class DefaultCustomersService implements CustomersService {
11
12 @Autowired
13 private CustomersDao customersDao;
14
15 @Override
16 public List<customers> listAllCustomers() {
17 return customersDao.listAllCustomers();
18 }
19
20 @Override
21 public List<customers> retrieveCustomersByCustomerID(int customer_id) {
22 return customersDao.retrieveCustomersByCustomerID(customer_id);
23 }
24
25 @Override
26 public List<customers> retrieveCustomerACustomerByFirstNameAndLastName(String first_name,
27 String last_name) {
28 return customersDao.retrieveCustomerACustomerByFirstNameAndLastName(first_name, last_name);
29 }
30
31 @Override
32 public void addCustomer(String first_name, String last_name, String address, String phone) {
33 customersDao.addCustomer(first_name, last_name, address, phone);
34
35 }
36
37 @Override
38 public void deleteCustomerById(int customer_id) {
39 customersDao.deleteCustomerById(customer_id);
40
41 }
42
43 @Override
44 public void updateCustomerFisrtNameById(String first_name, int customer_id) {
45 customersDao.updateCustomerFisrtNameById(first_name, customer_id);
46
47 }
48
49 @Override
50 public void updateCustomerLastNameById(String last_name, int customer_id) {
51 customersDao.updateCustomerLastNameById(last_name, customer_id);

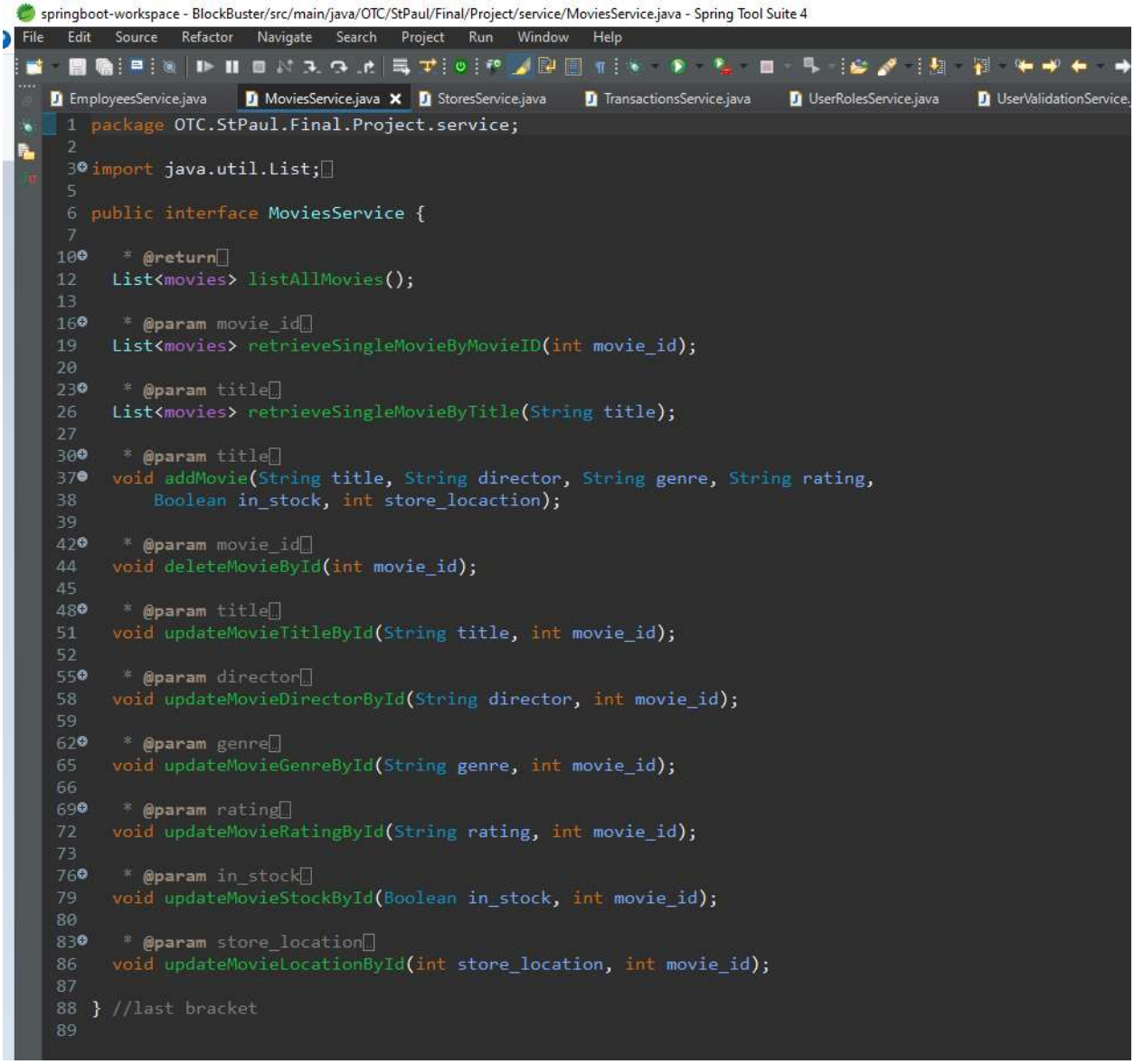
```
52
53    }
54
55    @Override
56    public void updateCustomerAddressById(String address, int customer_id) {
57      customersDao.updateCustomerAddressById(address, customer_id);
58
59    }
60
61    @Override
62    public void updateCustomerPhoneById(String phone, int customer_id) {
63      customersDao.updateCustomerPhoneById(phone, customer_id);
64
65    }
66
67  } // last bracket
68
```

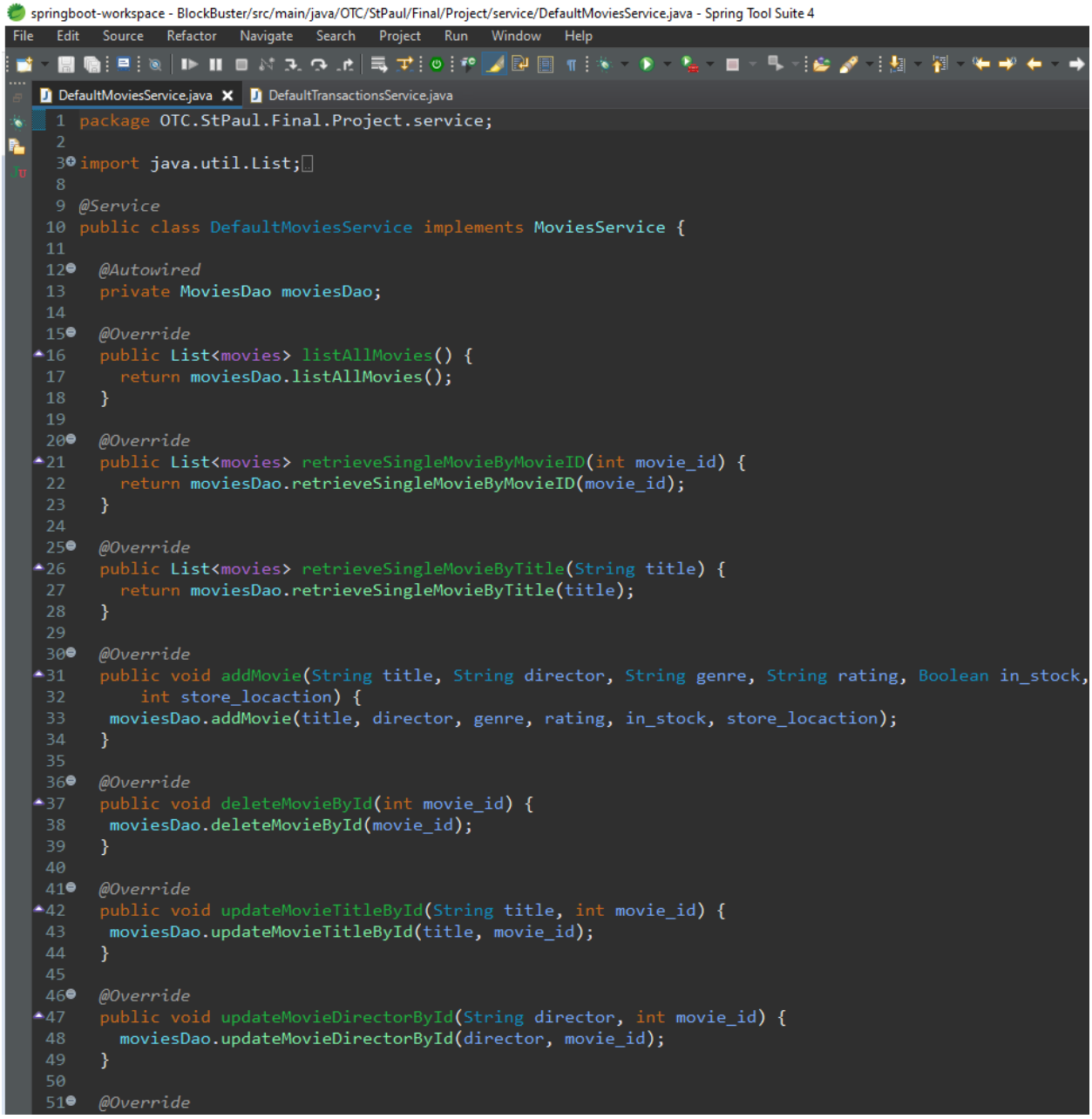
Stores-


springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/service/StoresService.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
EmployeesService.java
MoviesService.java
StoresService.java
TransactionsService.java
UserRolesService.java
UserValidationService.j
1 package OTC.StPaul.Final.Project.service;
2
3 import java.util.List;
5
6 public interface StoresService {
7 List<stores> listAllStores();
8
9 List<stores> retrieveAStoreByStoreId(int store_idPK);
10
11 void addStore(String address, String phone);
12
13 void deleteStoreById(int store_idPK);
14
15 void updateAddressByStoreId(String address, int store_idPK);
16
17 void updatePhoneByStroreId(String phone, int store_idPK);
18
19 } // last bracket
20


springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/service/DefaultStoresService.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
DefaultConcessionsService.java
DefaultCustomersService.java
DefaultEmployeesService.java
DefaultMoviesService.java
DefaultStoresServic
1 package OTC.StPaul.Final.Project.service;
2
3 import java.util.List;
8
9 @Service
10 public class DefaultStoresService implements StoresService {
11
12 @Autowired
13 private StoresDao storesDao;
14
15 @Override
16 public List<stores> listAllStores() {
17 return storesDao.listAllStores();
18 }
19
20 @Override
21 public List<stores> retrieveAStoreByStoreId(int store_idPK) {
22 return storesDao.retrieveAStoreByStoreId(store_idPK);
23 }
24
25 @Override
26 public void addStore(String address, String phone) {
27 storesDao.addStore(address, phone);
28
29 }
30
31 @Override
32 public void deleteStoreById(int store_idPK) {
33 storesDao.deleteStoreById(store_idPK);
34
35 }
36
37 @Override
38 public void updateAddressByStoreId(String address, int store_idPK) {
39 storesDao.updateAddressByStoreId(address, store_idPK);
40
41 }
42
43 @Override
44 public void updatePhoneByStroreId(String phone, int store_idPK) {
45 storesDao.updatePhoneByStroreId(phone, store_idPK);
46
47 }
48
49 } // last bracket
50

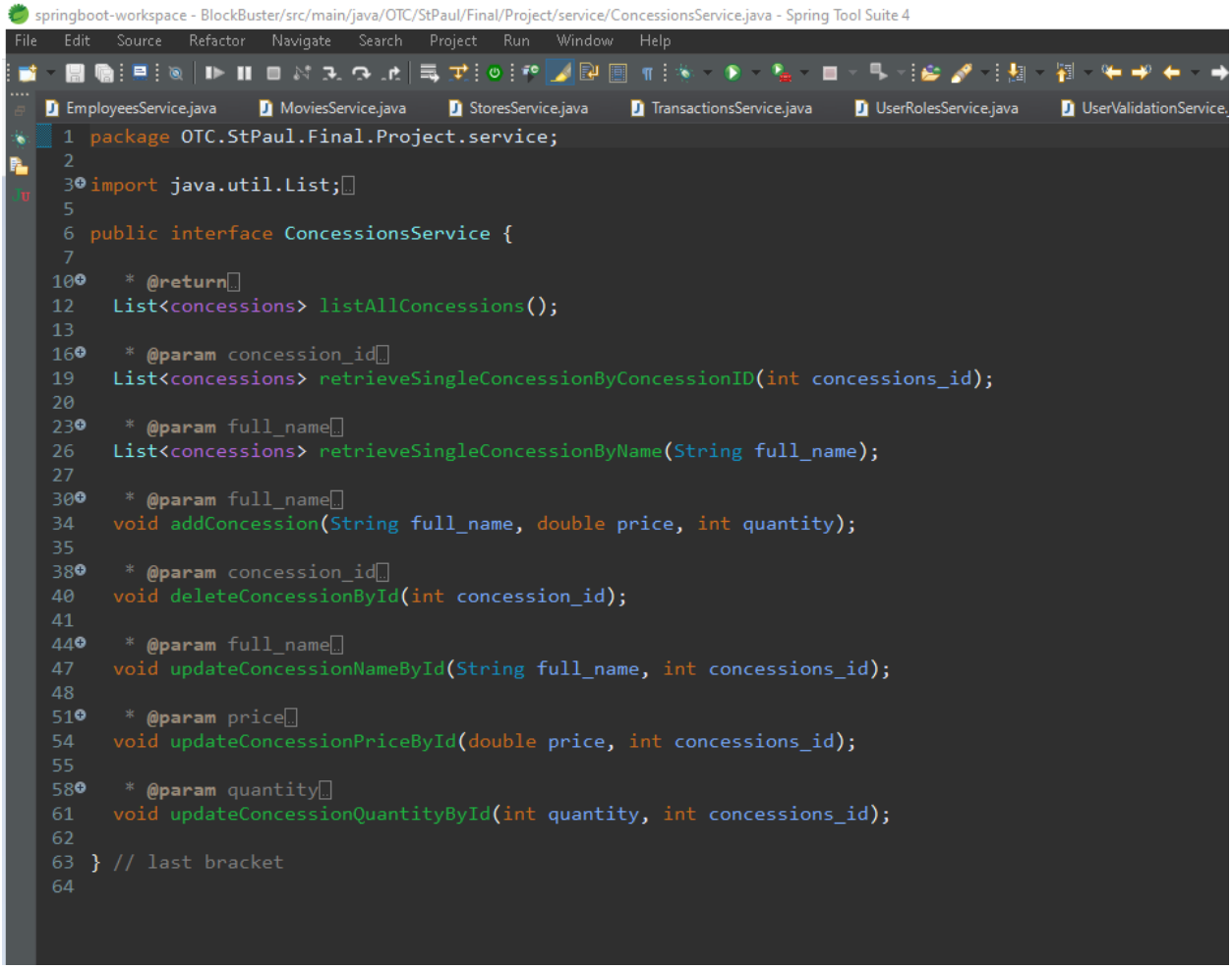
Movies-

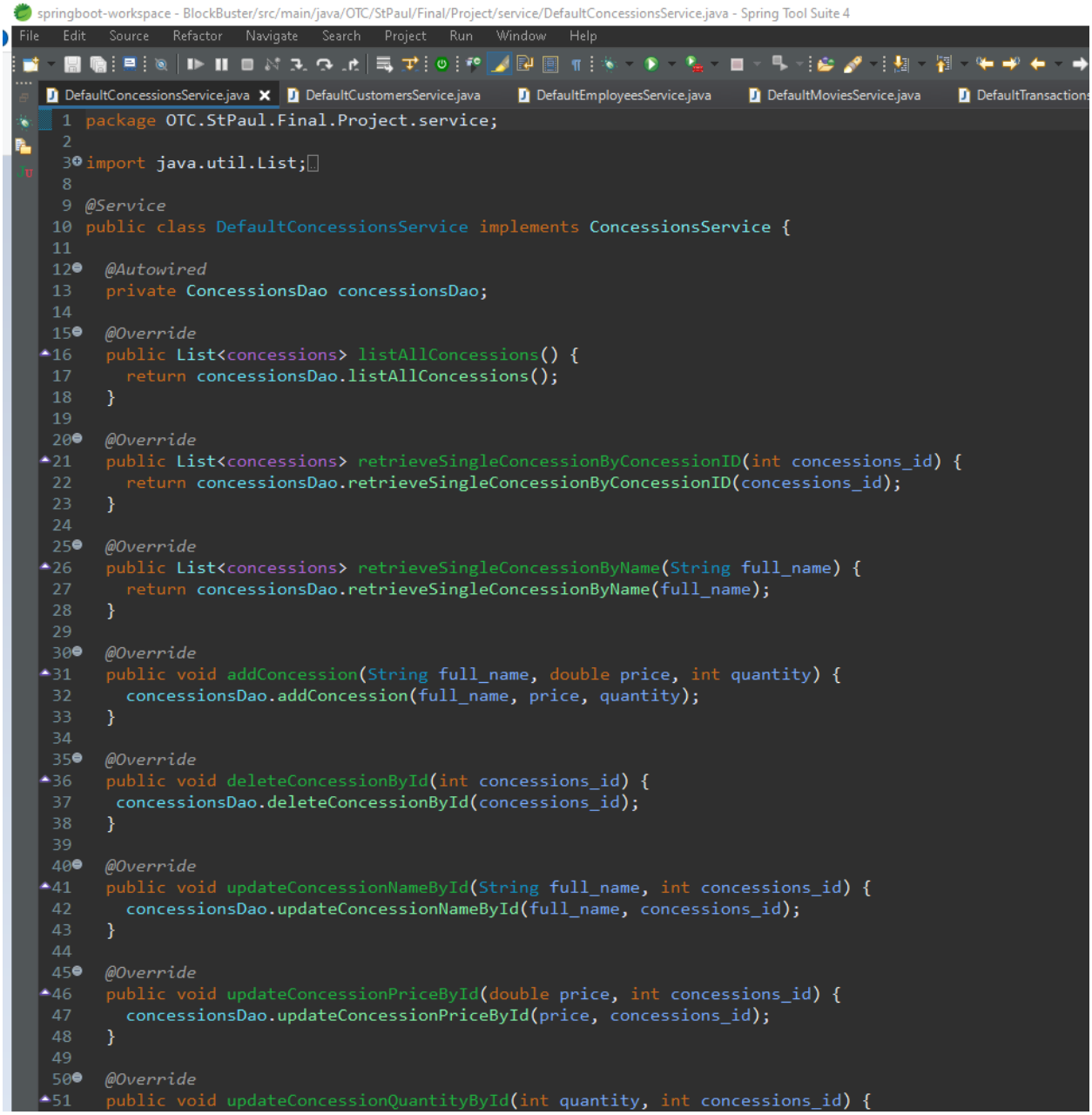

springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/service/MoviesService.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
EmployeesService.java
MoviesService.java
StoresService.java
TransactionsService.java
UserRolesService.java
UserValidationService.
1 package OTC.StPaul.Final.Project.service;
2
3 import java.util.List;
5
6 public interface MoviesService {
7
10 * @return
12 List<movies> listAllMovies();
13
16 * @param movie_id
19 List<movies> retrieveSingleMovieByMovieID(int movie_id);
20
23 * @param title
26 List<movies> retrieveSingleMovieByTitle(String title);
27
30 * @param title
37 void addMovie(String title, String director, String genre, String rating,
38 Boolean in_stock, int store_locaction);
39
42 * @param movie_id
44 void deleteMovieById(int movie_id);
45
48 * @param title
51 void updateMovieTitleById(String title, int movie_id);
52
55 * @param director
58 void updateMovieDirectorById(String director, int movie_id);
59
62 * @param genre
65 void updateMovieGenreById(String genre, int movie_id);
66
69 * @param rating
72 void updateMovieRatingById(String rating, int movie_id);
73
76 * @param in_stock
79 void updateMovieStockById(Boolean in_stock, int movie_id);
80
83 * @param store_location
86 void updateMovieLocationById(int store_location, int movie_id);
87
88 } //last bracket
89


springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/service/DefaultMoviesService.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
DefaultMoviesService.java
DefaultTransactionsService.java
1 package OTC.StPaul.Final.Project.service;
2
3 import java.util.List;
8
9 @Service
10 public class DefaultMoviesService implements MoviesService {
11
12 @Autowired
13 private MoviesDao moviesDao;
14
15 @Override
16 public List<movies> listAllMovies() {
17 return moviesDao.listAllMovies();
18 }
19
20 @Override
21 public List<movies> retrieveSingleMovieByMovieID(int movie_id) {
22 return moviesDao.retrieveSingleMovieByMovieID(movie_id);
23 }
24
25 @Override
26 public List<movies> retrieveSingleMovieByTitle(String title) {
27 return moviesDao.retrieveSingleMovieByTitle(title);
28 }
29
30 @Override
31 public void addMovie(String title, String director, String genre, String rating, Boolean in_stock,
32 int store_locaction) {
33 moviesDao.addMovie(title, director, genre, rating, in_stock, store_locaction);
34 }
35
36 @Override
37 public void deleteMovieById(int movie_id) {
38 moviesDao.deleteMovieById(movie_id);
39 }
40
41 @Override
42 public void updateMovieTitleById(String title, int movie_id) {
43 moviesDao.updateMovieTitleById(title, movie_id);
44 }
45
46 @Override
47 public void updateMovieDirectorById(String director, int movie_id) {
48 moviesDao.updateMovieDirectorById(director, movie_id);
49 }
50
51 @Override

```
52    public void updateMovieGenreById(String genre, int movie_id) {
53      moviesDao.updateMovieGenreById(genre, movie_id);
54    }
55
56    @Override
57    public void updateMovieRatingById(String rating, int movie_id) {
58      moviesDao.updateMovieRatingById(rating, movie_id);
59    }
60
61    @Override
62    public void updateMovieStockById(Boolean in_stock, int movie_id) {
63      moviesDao.updateMovieStockById(in_stock, movie_id);
64    }
65
66    @Override
67    public void updateMovieLocationById(int store_location, int movie_id) {
68      moviesDao.updateMovieLocationById(store_location, movie_id);
69    }
70
71  } //last bracket
72
```

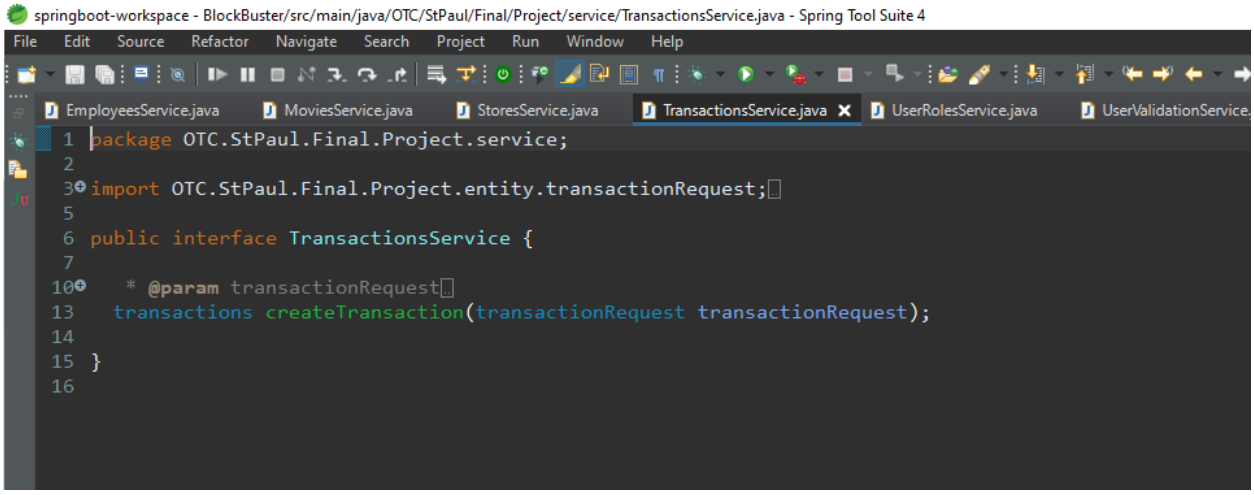
Concessions-

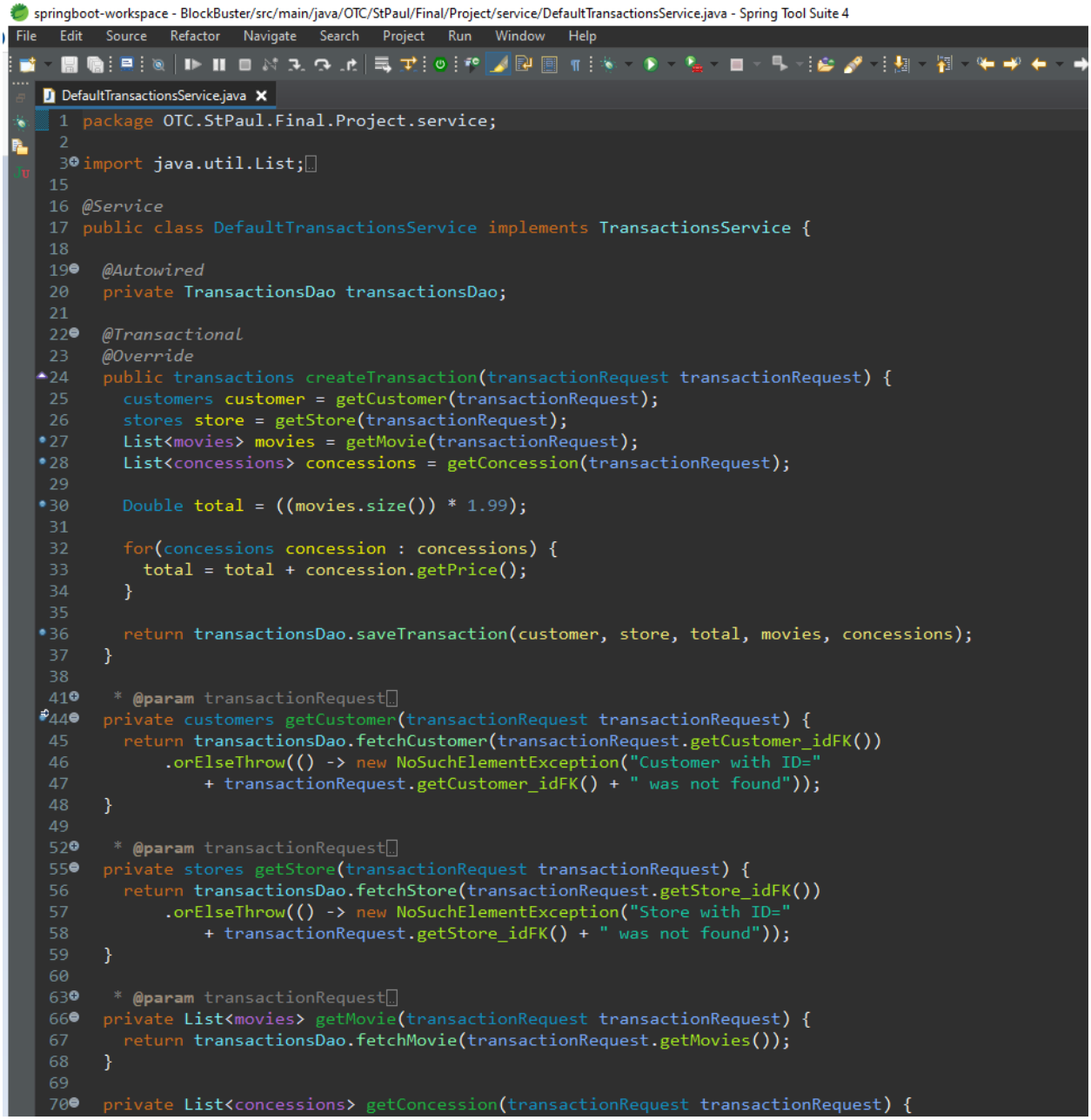


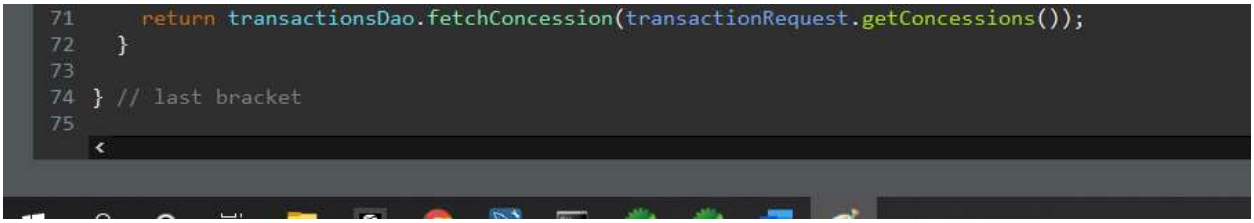



```
51   public void updateConcessionQuantityById(int quantity, int concessions_id) {
52     concessionsDao.updateConcessionQuantityById(quantity, concessions_id);
53   }
54
55 } // last bracket
56
```

Transactions-

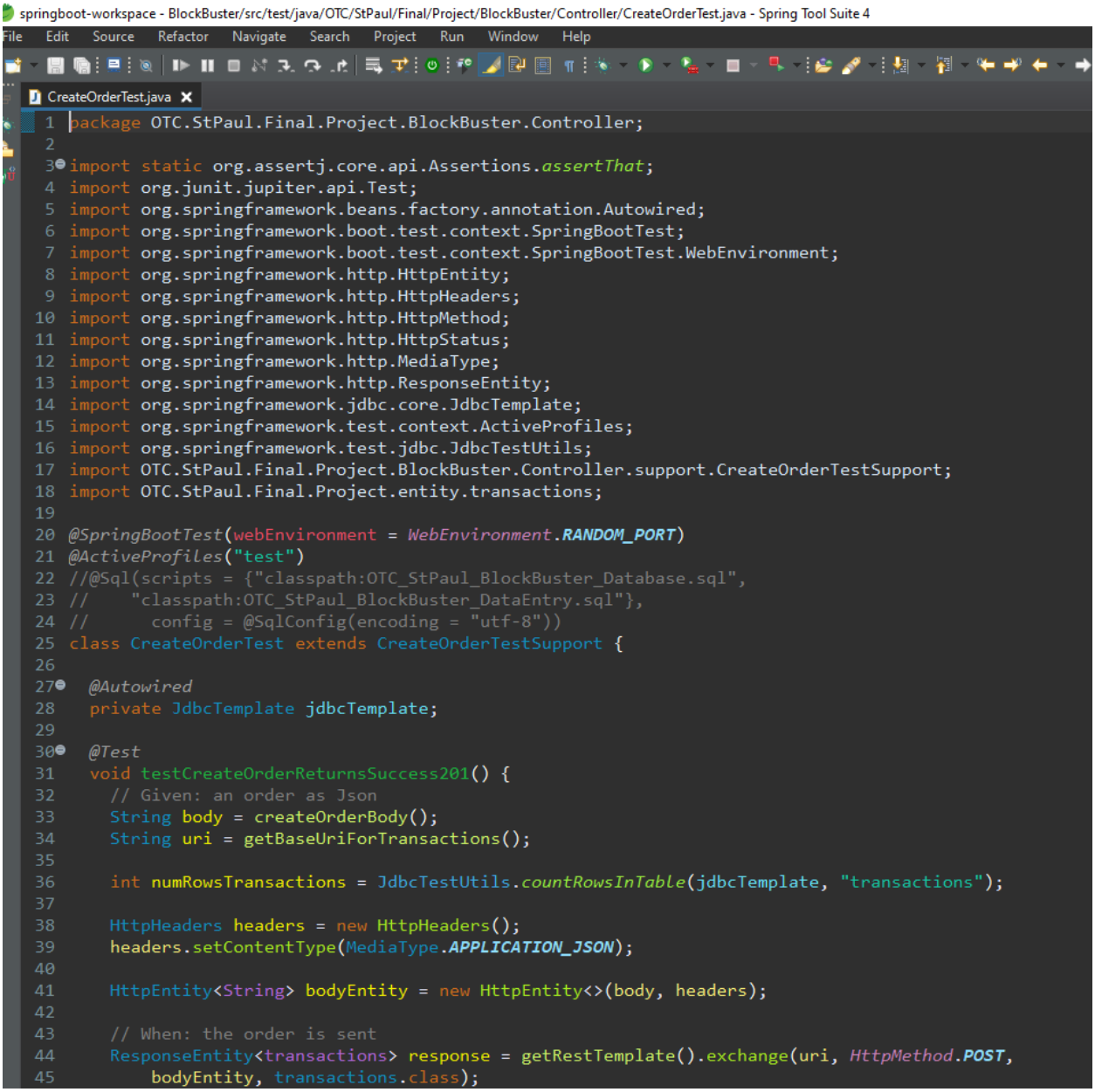


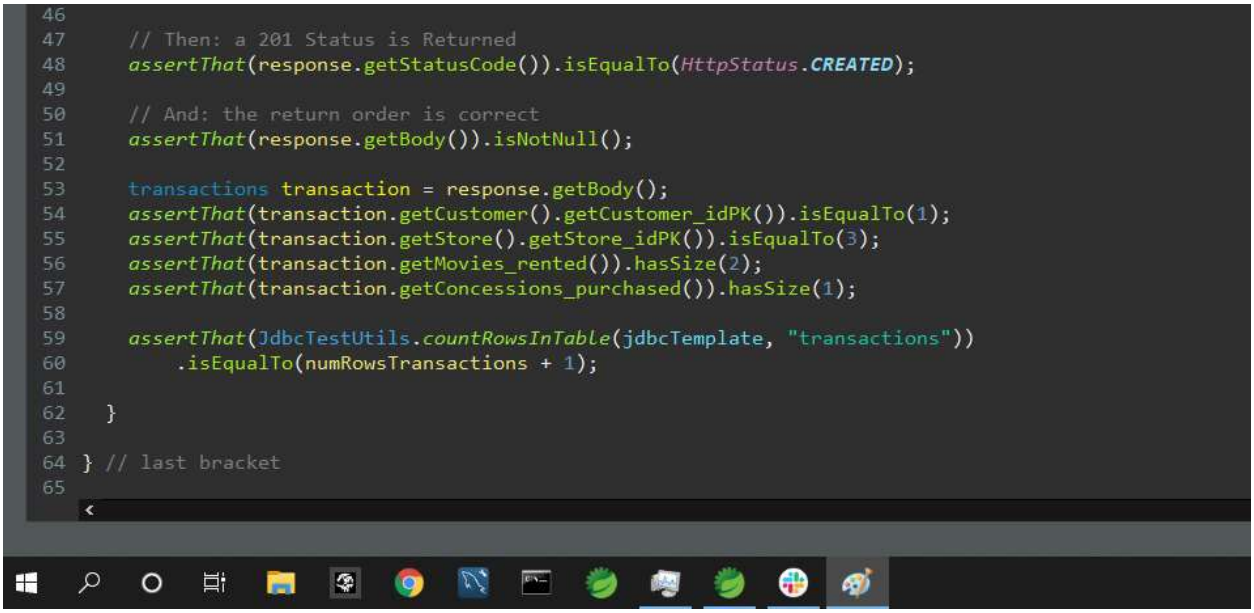

springboot-workspace - BlockBuster/src/main/java/OTC/StPaul/Final/Project/service/DefaultTransactionsService.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
DefaultTransactionsService.java
1 package OTC.StPaul.Final.Project.service;
2
3 import java.util.List;
15
16 @Service
17 public class DefaultTransactionsService implements TransactionsService {
18
19 @Autowired
20 private TransactionsDao transactionsDao;
21
22 @Transactional
23 @Override
24 public transactions createTransaction(transactionRequest transactionRequest) {
25 customers customer = getCustomer(transactionRequest);
26 stores store = getStore(transactionRequest);
27 List<movies> movies = getMovie(transactionRequest);
28 List<concessions> concessions = getConcession(transactionRequest);
29
30 Double total = ((movies.size()) * 1.99);
31
32 for(concessions concession : concessions) {
33 total = total + concession.getPrice();
34 }
35
36 return transactionsDao.saveTransaction(customer, store, total, movies, concessions);
37 }
38
41 * @param transactionRequest
44 private customers getCustomer(transactionRequest transactionRequest) {
45 return transactionsDao.fetchCustomer(transactionRequest.getCustomer_idFK())
46 .orElseThrow(() -> new NoSuchElementException("Customer with ID="
47 + transactionRequest.getCustomer_idFK() + " was not found"));
48 }
49
52 * @param transactionRequest
55 private stores getStore(transactionRequest transactionRequest) {
56 return transactionsDao.fetchStore(transactionRequest.getStore_idFK())
57 .orElseThrow(() -> new NoSuchElementException("Store with ID="
58 + transactionRequest.getStore_idFK() + " was not found"));
59 }
60
63 * @param transactionRequest
66 private List<movies> getMovie(transactionRequest transactionRequest) {
67 return transactionsDao.fetchMovie(transactionRequest.getMovies());
68 }
69
70 private List<concessions> getConcession(transactionRequest transactionRequest) {


71 return transactionsDao.fetchConcession(transactionRequest.getConcessions());
72 }
73
74 } // last bracket
75

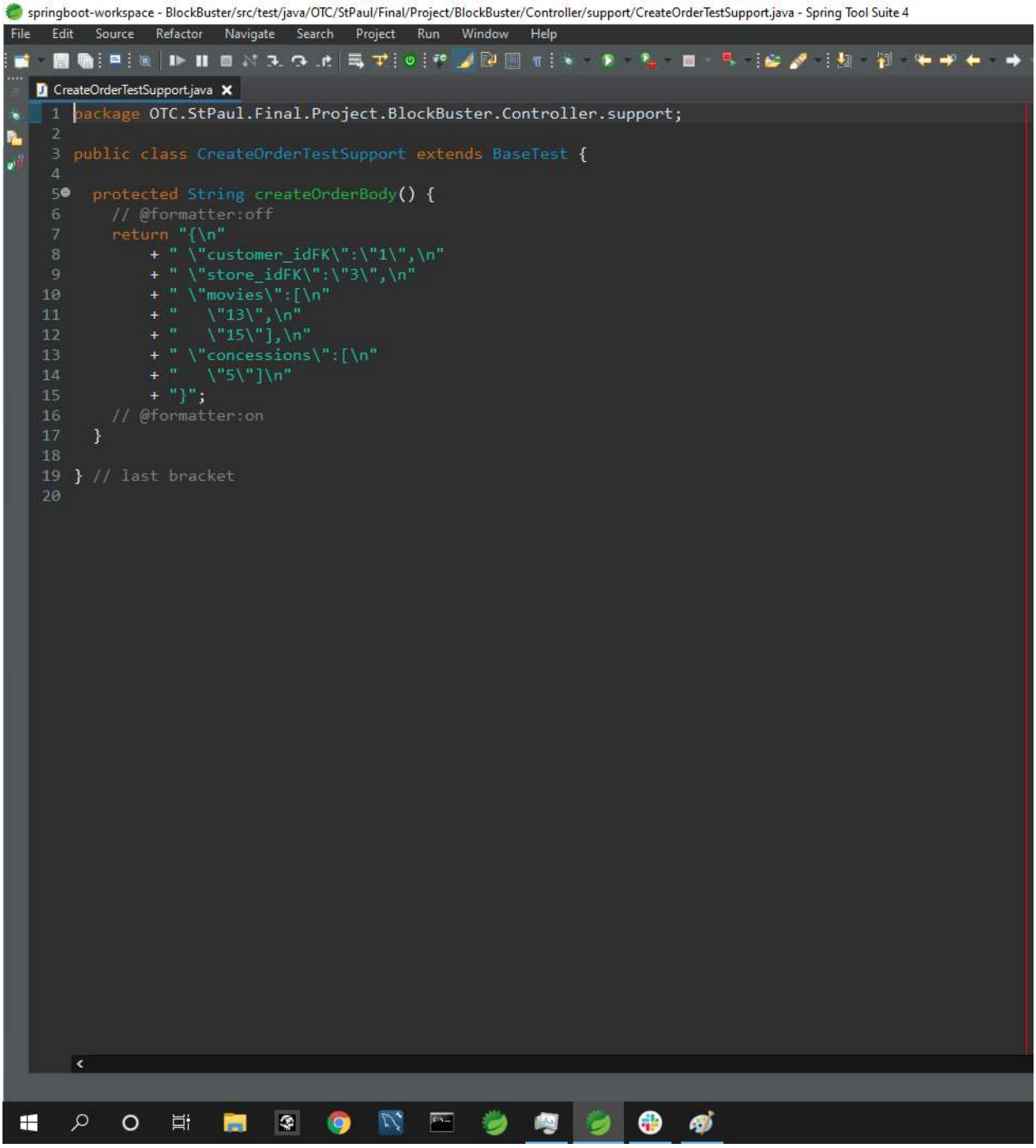
Test:

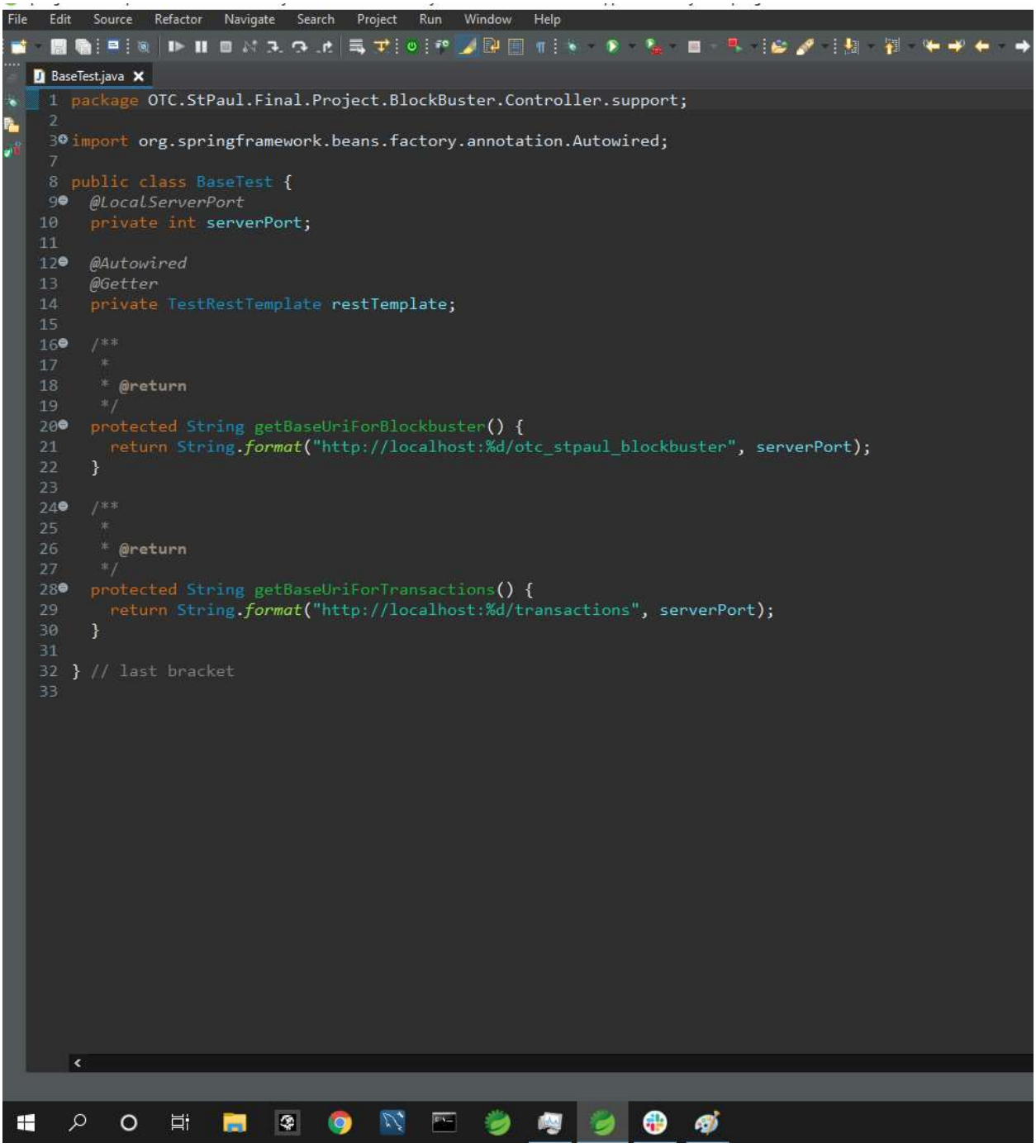
Controller-




// Then: a 201 Status is Returned
assertThat(response.getStatusCode()).isEqualTo(HttpStatus.CREATED);
// And: the return order is correct
assertThat(response.getBody()).isNotNull();
transactions transaction = response.getBody();
assertThat(transaction.getCustomer().getCustomer_idPK()).isEqualTo(1);
assertThat(transaction.getStore().getStore_idPK()).isEqualTo(3);
assertThat(transaction.getMovies_rented()).hasSize(2);
assertThat(transaction.getConcessions_purchased()).hasSize(1);
assertThat(JdbcTestUtils.countRowsInTable(jdbcTemplate, "transactions"))
.isEqualTo(numRowsTransactions + 1);
}
} // last bracket

Support-

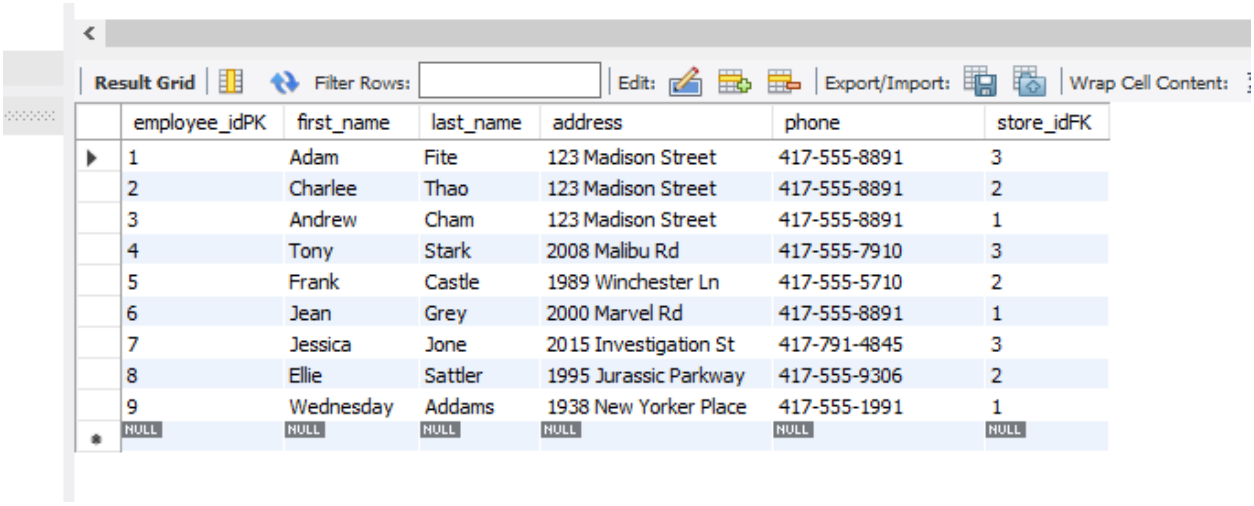

springboot-workspace - BlockBuster/src/test/java/OTC/StPaul/Final/Project/BlockBuster/Controller/support/CreateOrderTestSupport.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
CreateOrderTestSupport.java
1 package OTC.StPaul.Final.Project.BlockBuster.Controller.support;
2
3 public class CreateOrderTestSupport extends BaseTest {
4
5 protected String createOrderBody() {
6 // @formatter:off
7 return "{\n"
8 + " \"customer_idFK\":\"1\",\n"
9 + " \"store_idFK\":\"3\",\n"
10 + " \"movies\":[\n"
11 + " \"13\",\n"
12 + " \"15\"],\n"
13 + " \"concessions\":[\n"
14 + " \"5\"]\n"
15 + "}";
16 // @formatter:on
17 }
18
19 } // last bracket
20


File Edit Source Refactor Navigate Search Project Run Window Help
BaseTest.java
1 package OTC.StPaul.Final.Project.BlockBuster.Controller.support;
2
3 import org.springframework.beans.factory.annotation.Autowired;
7
8 public class BaseTest {
9 @LocalServerPort
10 private int serverPort;
11
12 @Autowired
13 @Getter
14 private TestRestTemplate restTemplate;
15
16 /**
17 *
18 * @return
19 */
20 protected String getBaseUriForBlockbuster() {
21 return String.format("http://localhost:%d/otc_stpaul_blockbuster", serverPort);
22 }
23
24 /**
25 *
26 * @return
27 */
28 protected String getBaseUriForTransactions() {
29 return String.format("http://localhost:%d/transactions", serverPort);
30 }
31
32 } // last bracket
33

Swagger Running API:

Employees:

Starting Employees Table -



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employee_idPK	first_name	last_name	address	phone	store_idFK
1	Adam	Fite	123 Madison Street	417-555-8891	3
2	Charlee	Thao	123 Madison Street	417-555-8891	2
3	Andrew	Cham	123 Madison Street	417-555-8891	1
4	Tony	Stark	2008 Malibu Rd	417-555-7910	3
5	Frank	Castle	1989 Winchester Ln	417-555-5710	2
6	Jean	Grey	2000 Marvel Rd	417-555-8891	1
7	Jessica	Jone	2015 Investigation St	417-791-4845	3
8	Ellie	Sattler	1995 Jurassic Parkway	417-555-9306	2
9	Wednesday	Addams	1938 New Yorker Place	417-555-1991	1
NULL	NULL	NULL	NULL	NULL	NULL

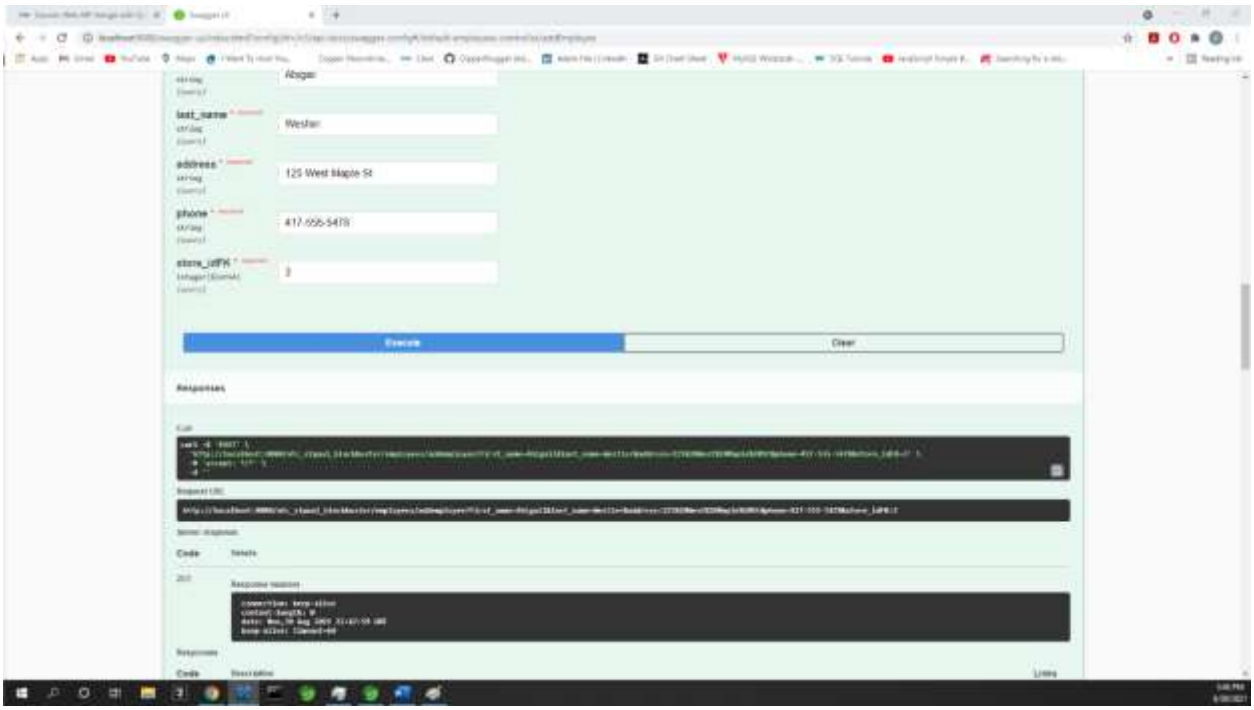
Variables:

Creating Use: Abigail Westler 125 West Maple St 417-555-5478 store 3

Update Sample: Abby Franklin 8525 North South Rd 417-555-6369

Create Employee –

-Swagger-

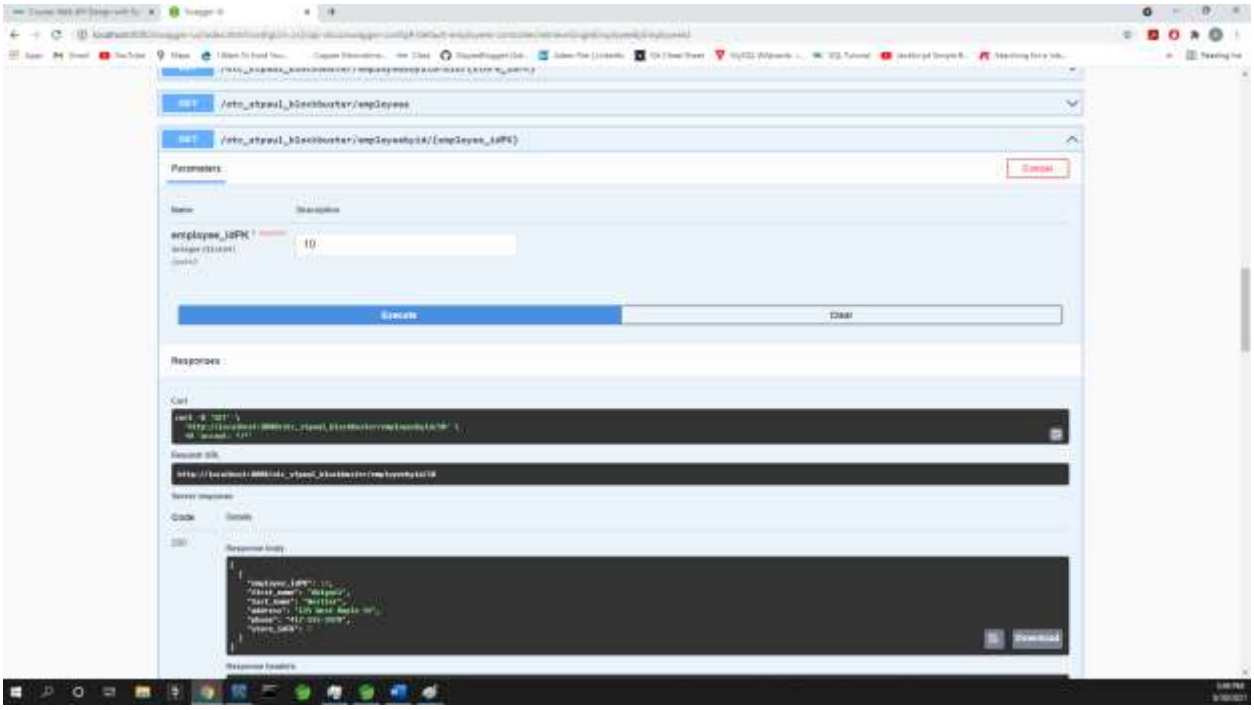


-Database-

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell C

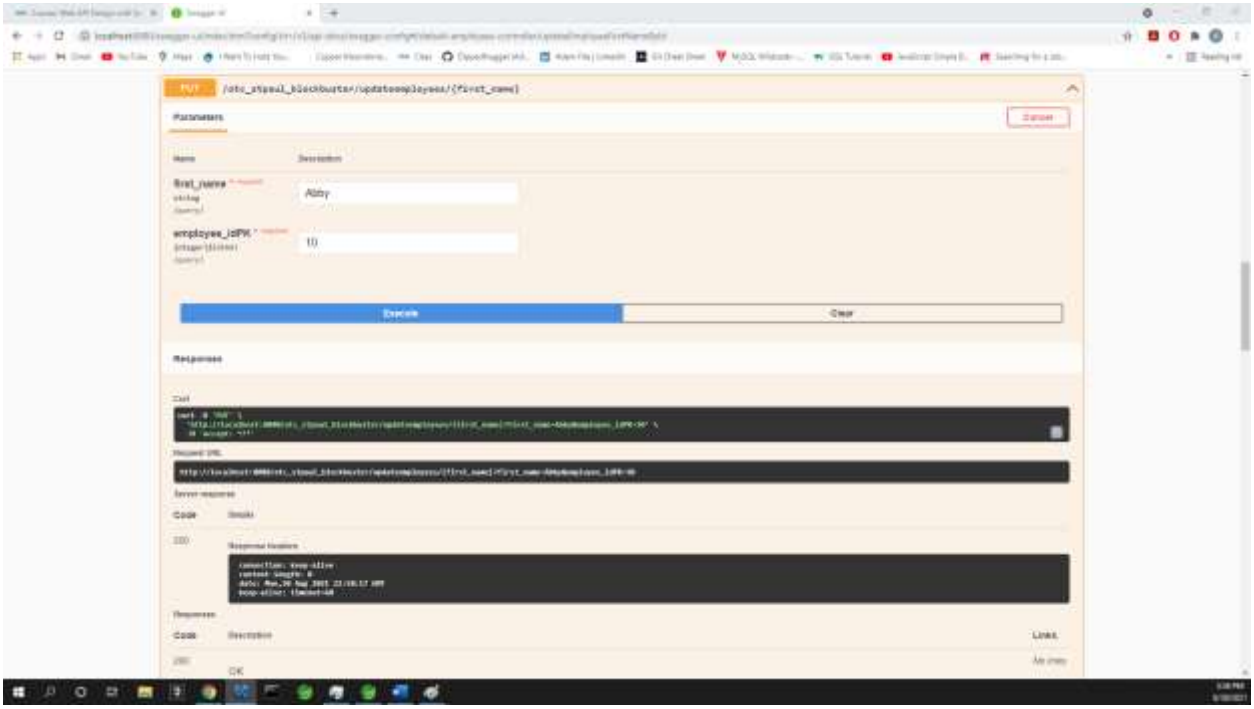
employee_idPK	first_name	last_name	address	phone	store_idFK
1	Adam	Fite	123 Madison Street	417-555-8891	3
2	Charlee	Thao	123 Madison Street	417-555-8891	2
3	Andrew	Cham	123 Madison Street	417-555-8891	1
4	Tony	Stark	2008 Malibu Rd	417-555-7910	3
5	Frank	Castle	1989 Winchester Ln	417-555-5710	2
6	Jean	Grey	2000 Marvel Rd	417-555-8891	1
7	Jessica	Jone	2015 Investigation St	417-791-4845	3
8	Ellie	Sattler	1995 Jurassic Parkway	417-555-9306	2
9	Wednesday	Addams	1938 New Yorker Place	417-555-1991	1
10	Abigail	Westler	125 West Maple St	417-555-5478	3
NULL	NULL	NULL	NULL	NULL	NULL

Get Employee –

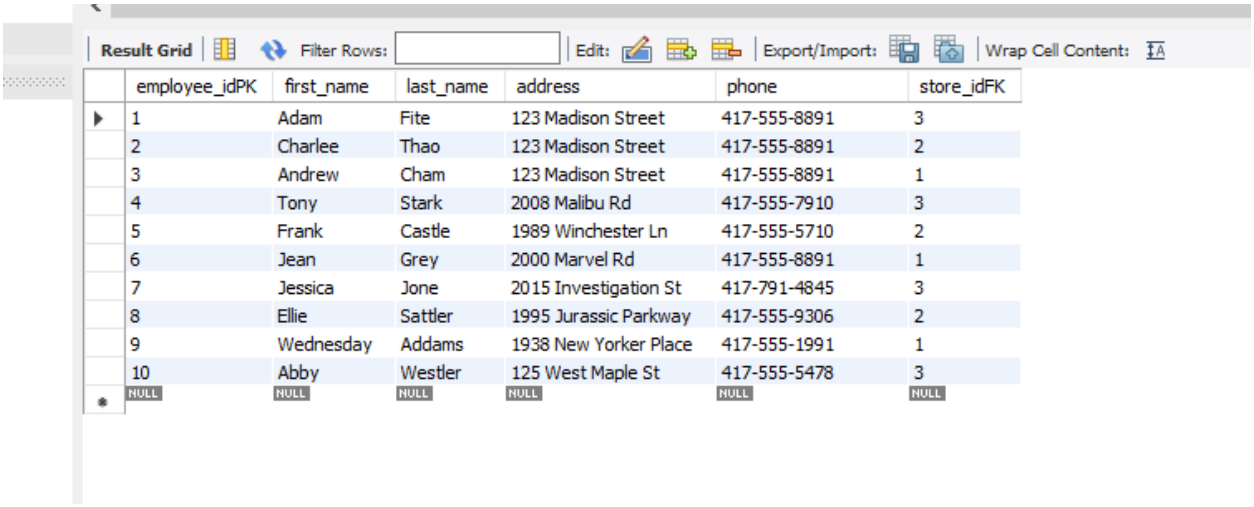


Update First Name –

-Swagger-



-Database-

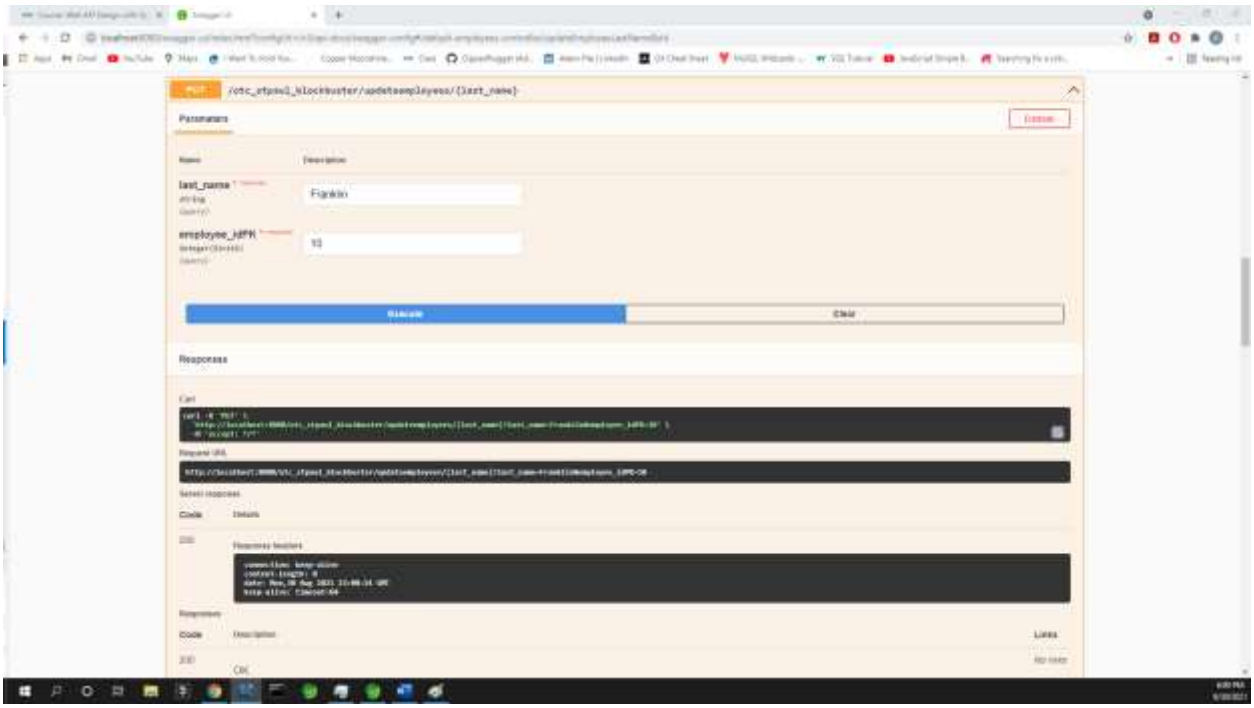


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employee_idPK	first_name	last_name	address	phone	store_idFK
1	Adam	Fite	123 Madison Street	417-555-8891	3
2	Charlee	Thao	123 Madison Street	417-555-8891	2
3	Andrew	Cham	123 Madison Street	417-555-8891	1
4	Tony	Stark	2008 Malibu Rd	417-555-7910	3
5	Frank	Castle	1989 Winchester Ln	417-555-5710	2
6	Jean	Grey	2000 Marvel Rd	417-555-8891	1
7	Jessica	Jone	2015 Investigation St	417-791-4845	3
8	Ellie	Sattler	1995 Jurassic Parkway	417-555-9306	2
9	Wednesday	Addams	1938 New Yorker Place	417-555-1991	1
10	Abby	Westler	125 West Maple St	417-555-5478	3
NULL	NULL	NULL	NULL	NULL	NULL

Update Last Name –

-Swagger-



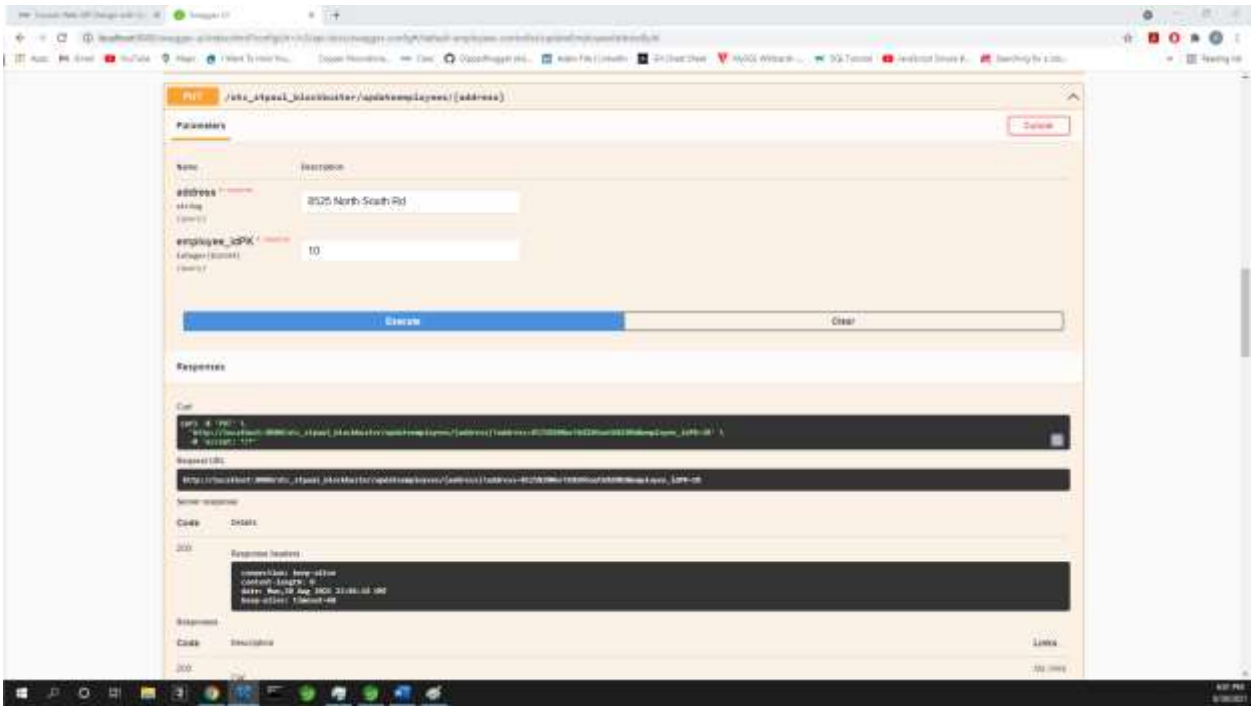
-Database-

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employee_idPK	first_name	last_name	address	phone	store_idFK
1	Adam	Fite	123 Madison Street	417-555-8891	3
2	Charlee	Thao	123 Madison Street	417-555-8891	2
3	Andrew	Cham	123 Madison Street	417-555-8891	1
4	Tony	Stark	2008 Malibu Rd	417-555-7910	3
5	Frank	Castle	1989 Winchester Ln	417-555-5710	2
6	Jean	Grey	2000 Marvel Rd	417-555-8891	1
7	Jessica	Jone	2015 Investigation St	417-791-4845	3
8	Ellie	Sattler	1995 Jurassic Parkway	417-555-9306	2
9	Wednesday	Addams	1938 New Yorker Place	417-555-1991	1
10	Abby	Franklin	125 West Maple St	417-555-5478	3
NULL	NULL	NULL	NULL	NULL	NULL

Update Address –

-Swagger-



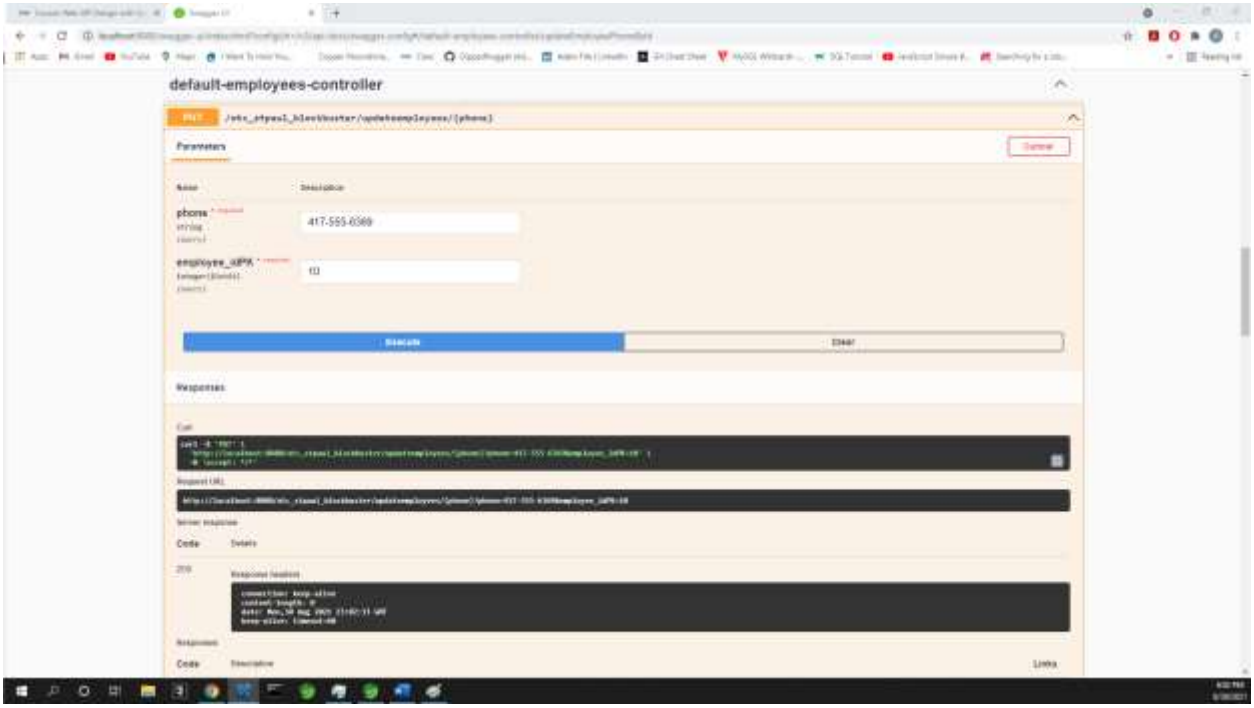
-Database-

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

employee_idPK	first_name	last_name	address	phone	store_idFK
1	Adam	Fite	123 Madison Street	417-555-8891	3
2	Charlee	Thao	123 Madison Street	417-555-8891	2
3	Andrew	Cham	123 Madison Street	417-555-8891	1
4	Tony	Stark	2008 Malibu Rd	417-555-7910	3
5	Frank	Castle	1989 Winchester Ln	417-555-5710	2
6	Jean	Grey	2000 Marvel Rd	417-555-8891	1
7	Jessica	Jone	2015 Investigation St	417-791-4845	3
8	Ellie	Sattler	1995 Jurassic Parkway	417-555-9306	2
9	Wednesday	Addams	1938 New Yorker Place	417-555-1991	1
10	Abby	Franklin	8525 North South Rd	417-555-5478	3
NULL	NULL	NULL	NULL	NULL	NULL

Update Phone Number –

-Swagger-



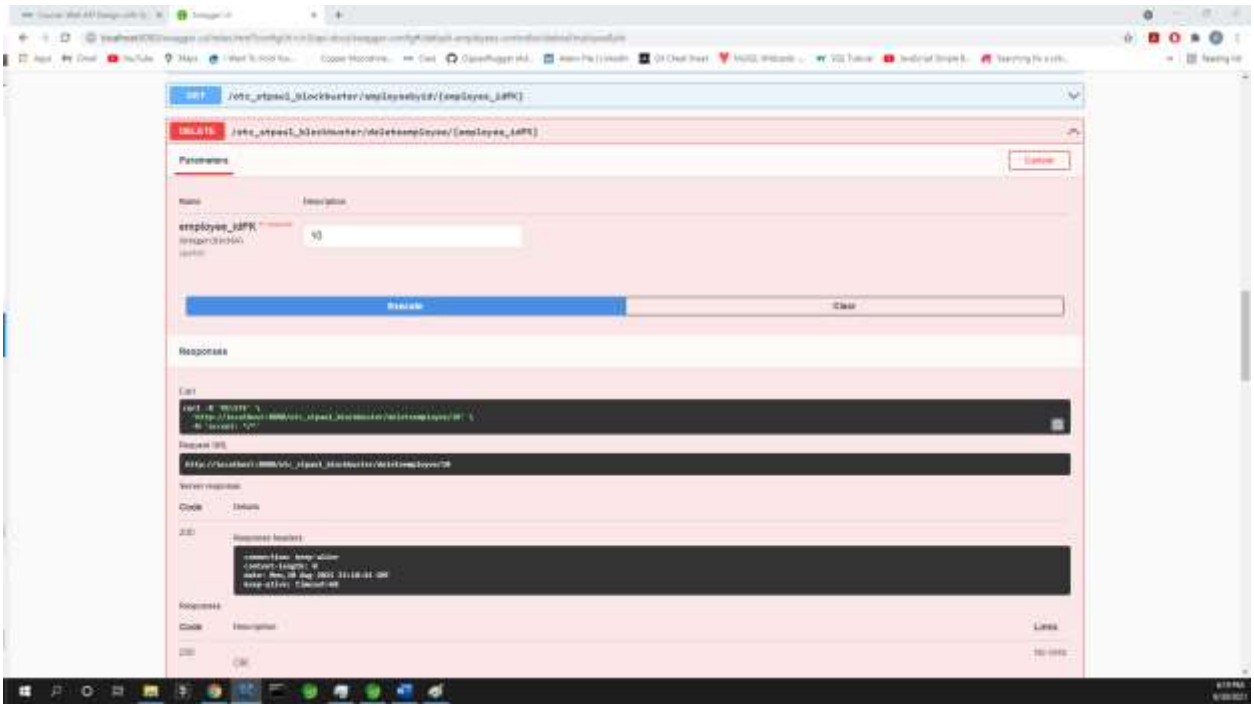
-Database-

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

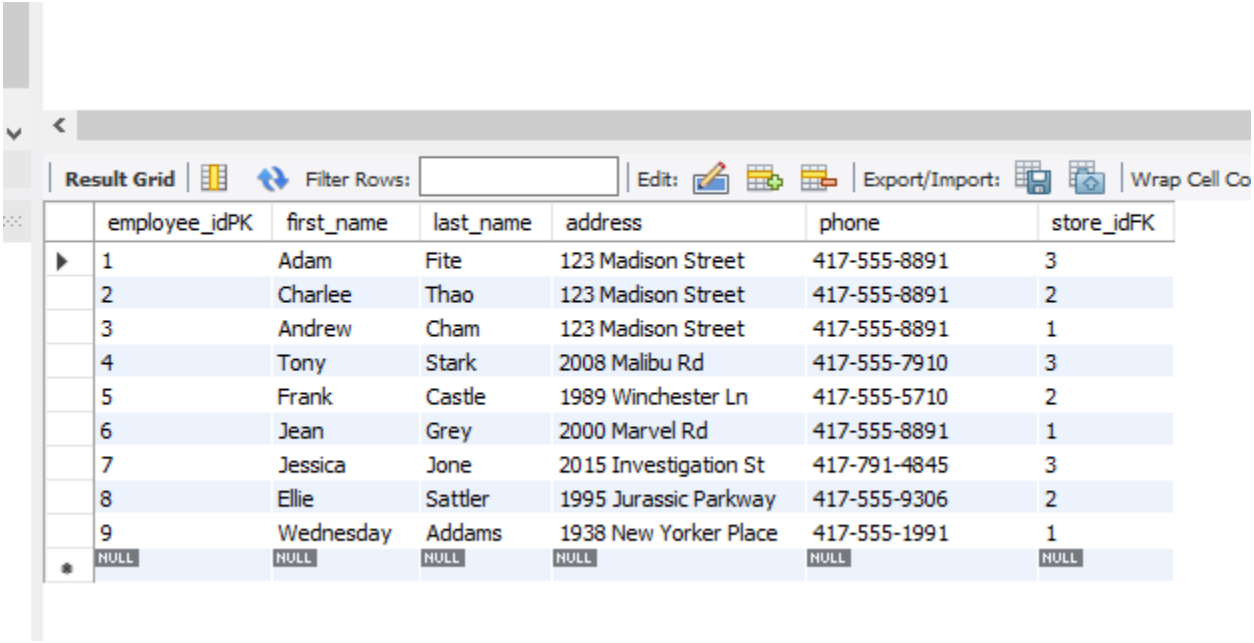
employee_idPK	first_name	last_name	address	phone	store_idFK
1	Adam	Fite	123 Madison Street	417-555-8891	3
2	Charlee	Thao	123 Madison Street	417-555-8891	2
3	Andrew	Cham	123 Madison Street	417-555-8891	1
4	Tony	Stark	2008 Malibu Rd	417-555-7910	3
5	Frank	Castle	1989 Winchester Ln	417-555-5710	2
6	Jean	Grey	2000 Marvel Rd	417-555-8891	1
7	Jessica	Jone	2015 Investigation St	417-791-4845	3
8	Ellie	Sattler	1995 Jurassic Parkway	417-555-9306	2
9	Wednesday	Addams	1938 New Yorker Place	417-555-1991	1
10	Abby	Franklin	8525 North South Rd	417-555-6369	3
NULL	NULL	NULL	NULL	NULL	NULL

Delete Employee –

-Swagger-



-Database-

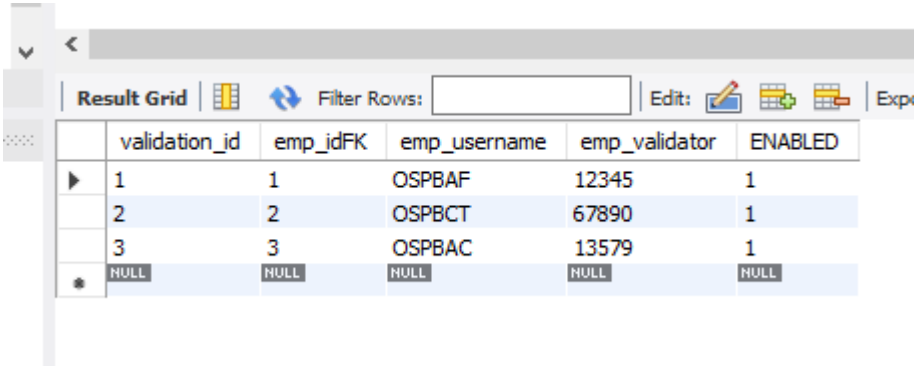


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Co

employee_idPK	first_name	last_name	address	phone	store_idFK
1	Adam	Fite	123 Madison Street	417-555-8891	3
2	Charlee	Thao	123 Madison Street	417-555-8891	2
3	Andrew	Cham	123 Madison Street	417-555-8891	1
4	Tony	Stark	2008 Malibu Rd	417-555-7910	3
5	Frank	Castle	1989 Winchester Ln	417-555-5710	2
6	Jean	Grey	2000 Marvel Rd	417-555-8891	1
7	Jessica	Jone	2015 Investigation St	417-791-4845	3
8	Ellie	Sattler	1995 Jurassic Parkway	417-555-9306	2
9	Wednesday	Addams	1938 New Yorker Place	417-555-1991	1
NULL	NULL	NULL	NULL	NULL	NULL

User Validation:

Starting User Validation –



Result Grid | Filter Rows: | Edit:

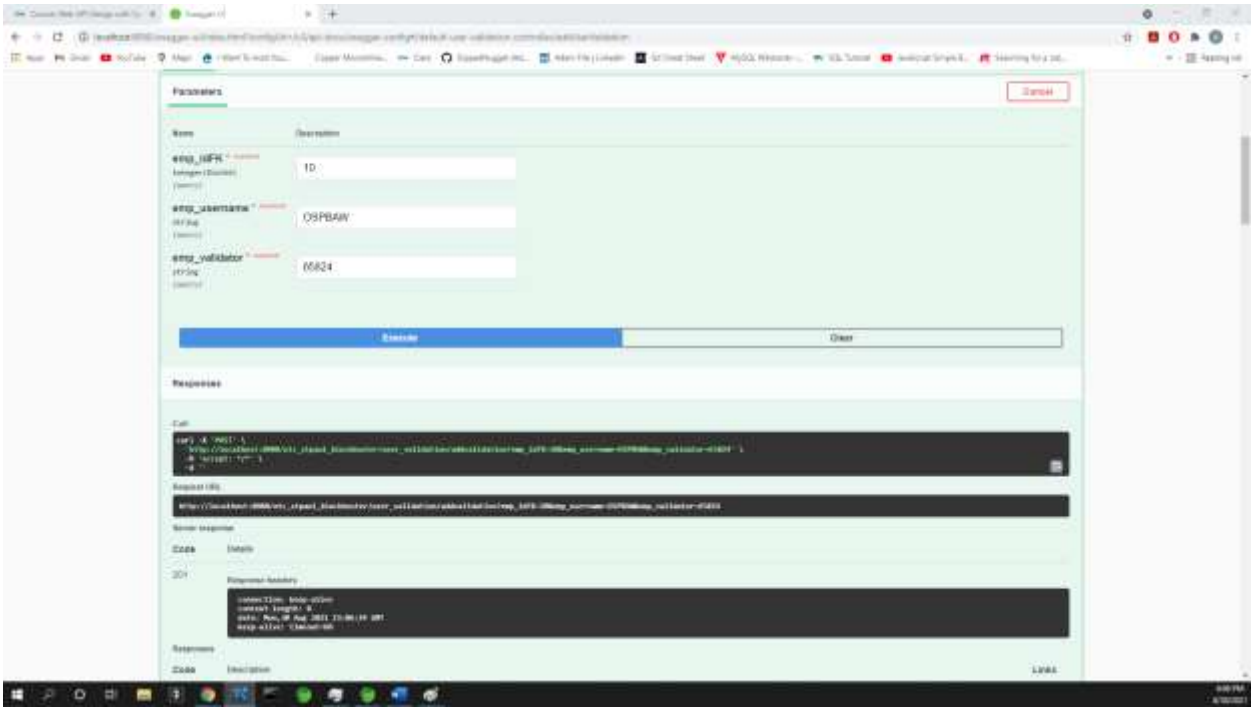
validation_id	emp_idFK	emp_username	emp_validator	ENABLED
1	1	OSPBAF	12345	1
2	2	OSPBCT	67890	1
3	3	OSPBAC	13579	1
NULL	NULL	NULL	NULL	NULL

Variables: emp# 10 username OSPBAW password 65824

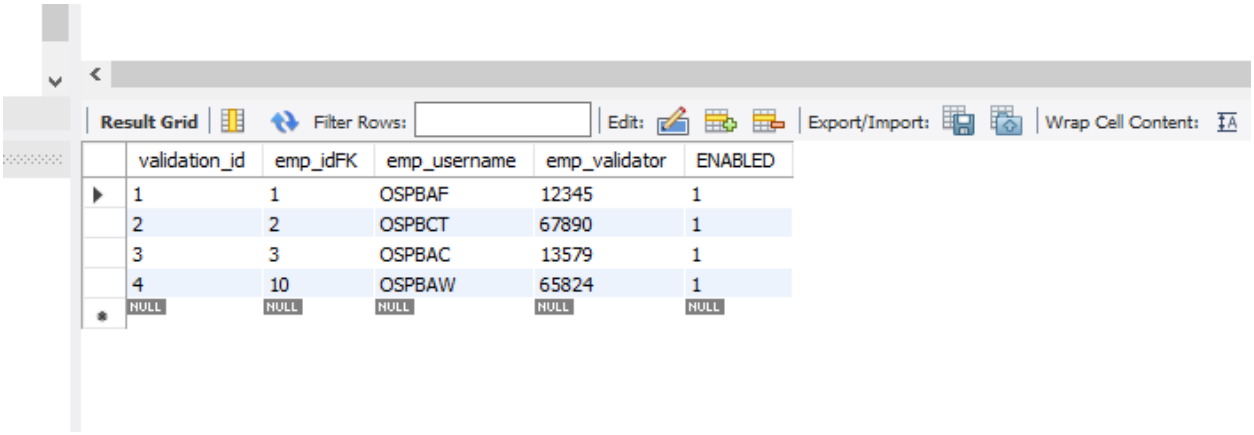
Update Variable: password

Create Validation –

-Swagger-



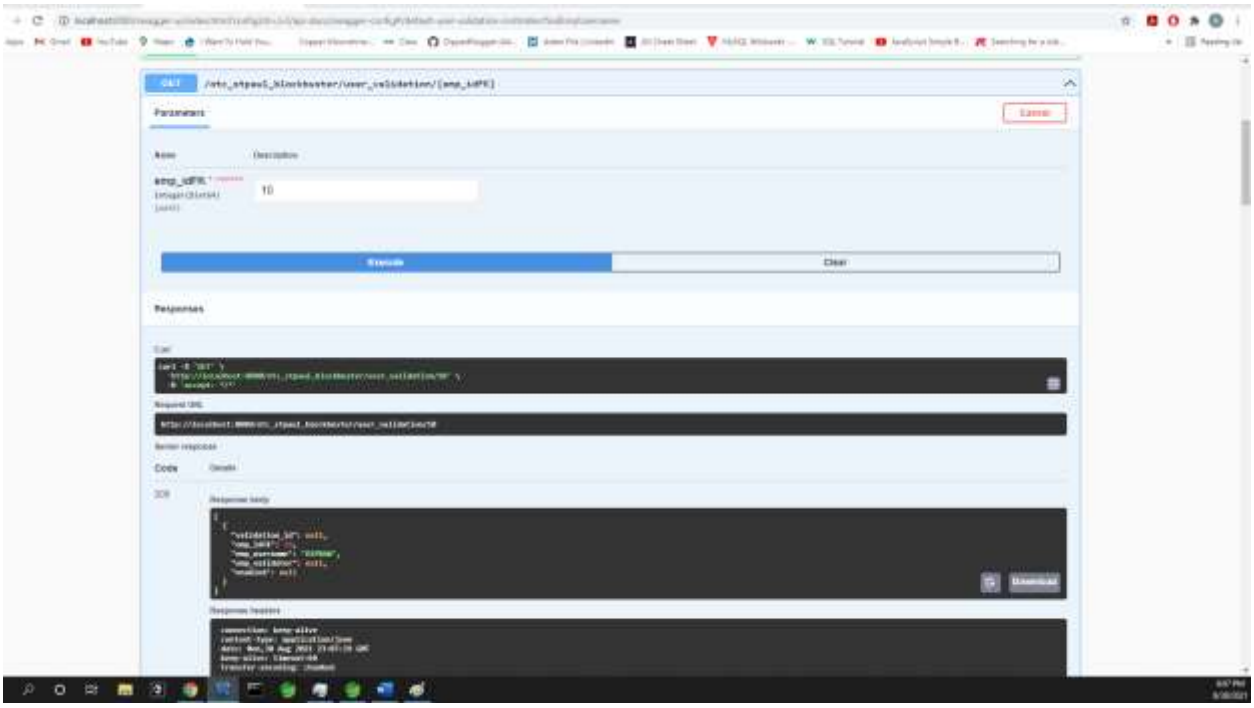
-Database-



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

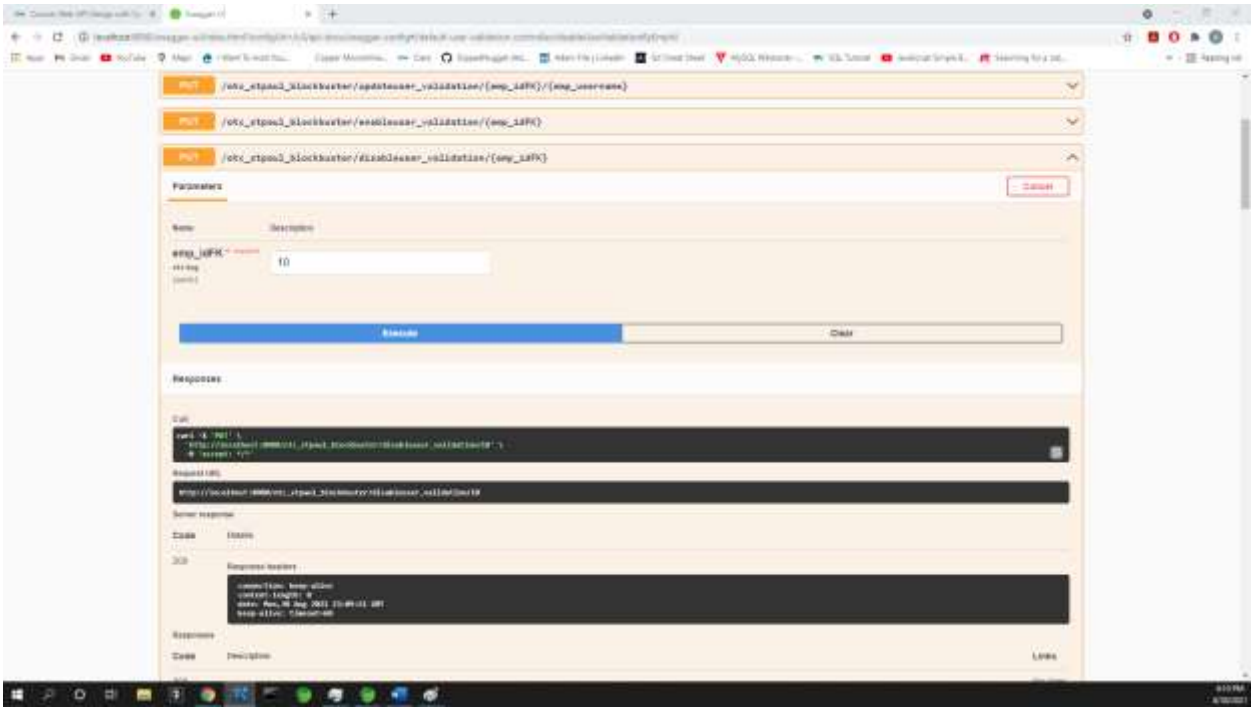
validation_id	emp_idFK	emp_username	emp_validator	ENABLED
1	1	OSPBAF	12345	1
2	2	OSPBCT	67890	1
3	3	OSPBAC	13579	1
4	10	OSPBAW	65824	1
NULL	NULL	NULL	NULL	NULL

Get Username –

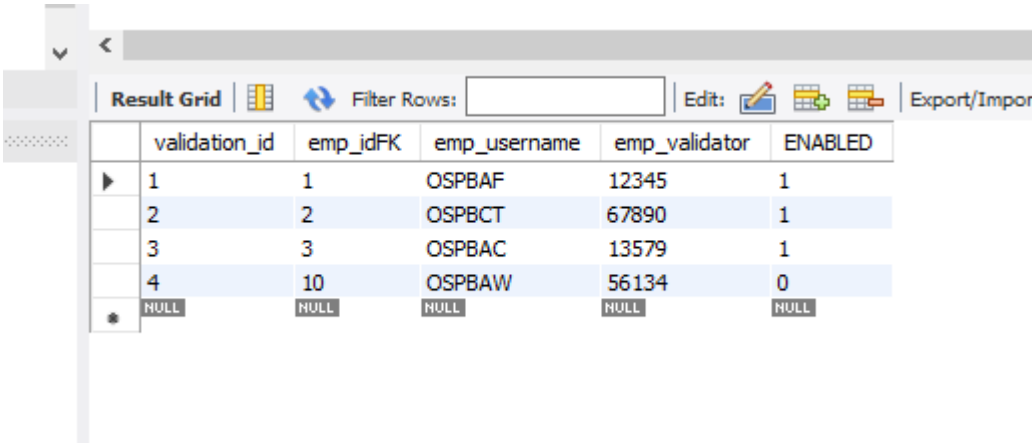


Disable Validation-

-Swagger-



-Database-

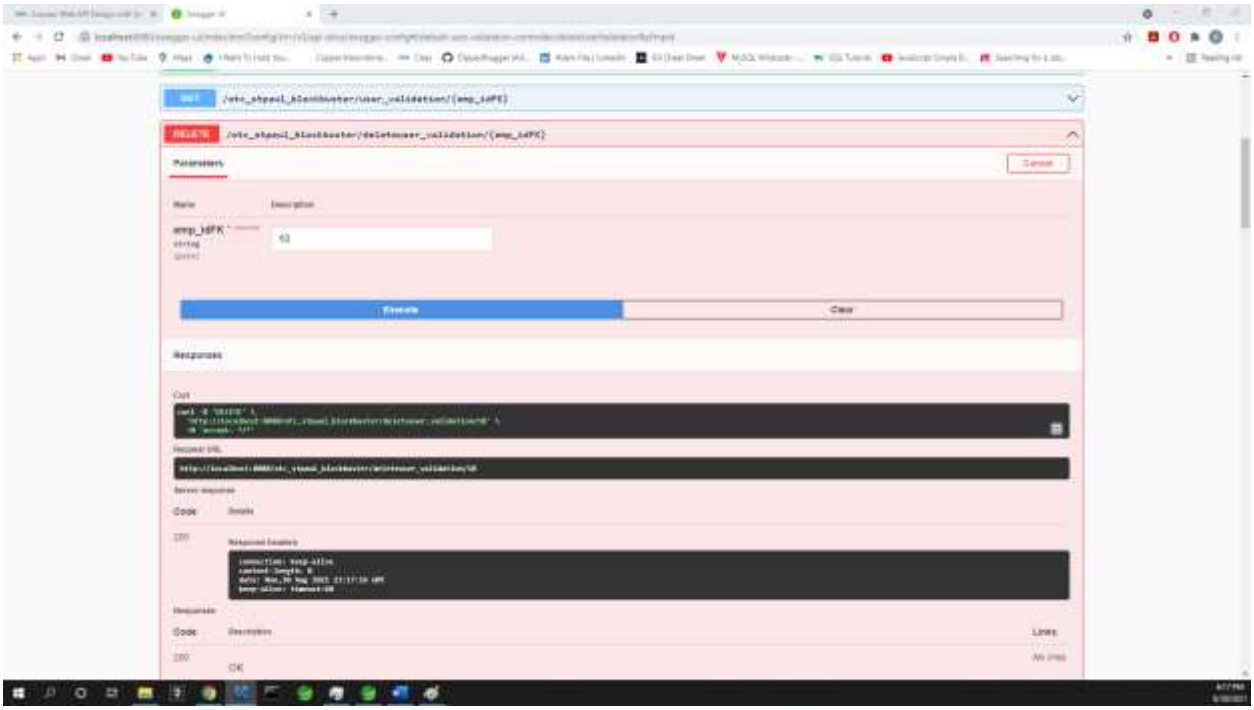


Result Grid | Filter Rows: | Edit: | Export/Impor

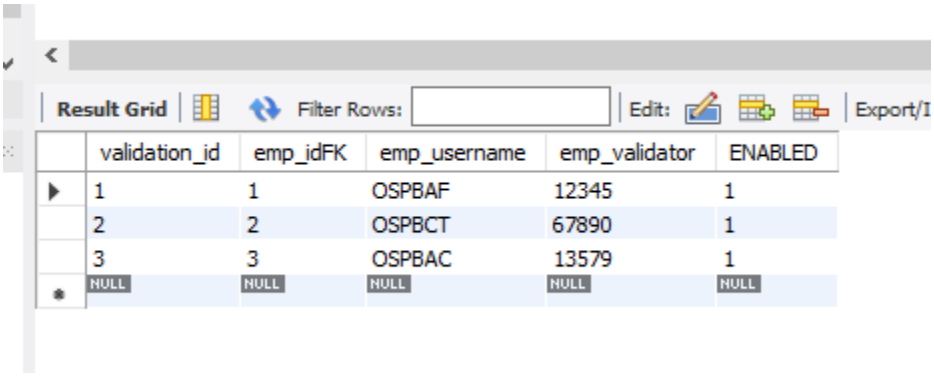
validation_id	emp_idFK	emp_username	emp_validator	ENABLED
1	1	OSPBAF	12345	1
2	2	OSPBCT	67890	1
3	3	OSPBAC	13579	1
4	10	OSPBAW	56134	0
NULL	NULL	NULL	NULL	NULL

Delete Validation –

-Swagger-



-Database-

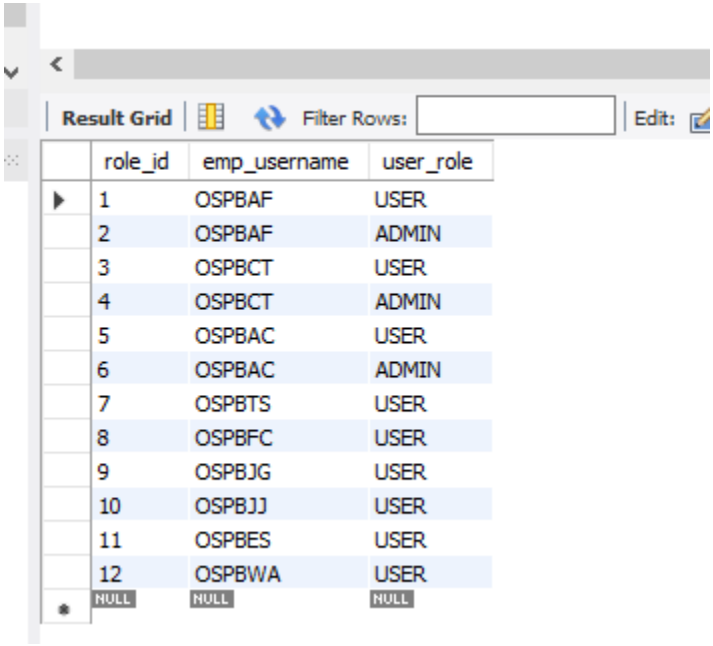


Result Grid | Filter Rows: | Edit: | Export/I

validation_id	emp_idFK	emp_username	emp_validator	ENABLED
1	1	OSPBAF	12345	1
2	2	OSPBCT	67890	1
3	3	OSPBAC	13579	1
NULL	NULL	NULL	NULL	NULL

User Roles:

Starting Database –

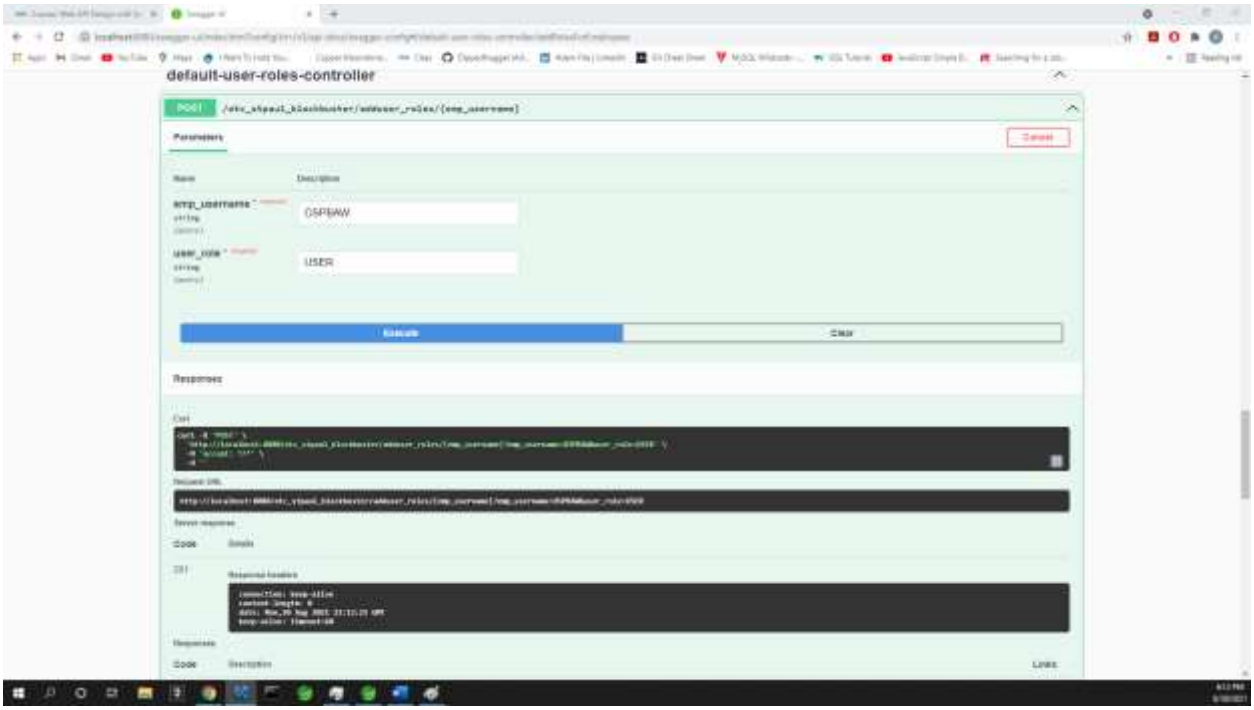


Result Grid | Filter Rows: | Edit:

role_id	emp_username	user_role
1	OSPBAF	USER
2	OSPBAF	ADMIN
3	OSPBCT	USER
4	OSPBCT	ADMIN
5	OSPBAC	USER
6	OSPBAC	ADMIN
7	OSPBTS	USER
8	OSPBFC	USER
9	OSPBJG	USER
10	OSPBJJ	USER
11	OSPBES	USER
12	OSPBWA	USER
NULL	NULL	NULL

Giving a role –

-Swagger-

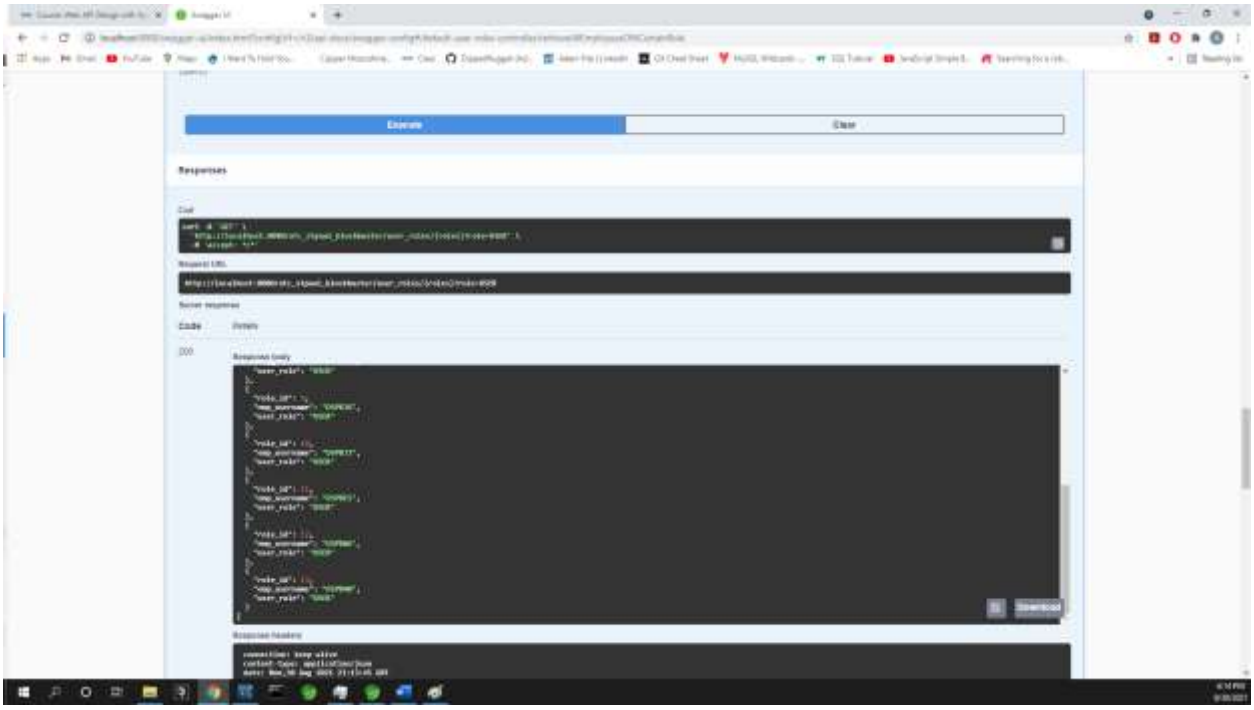


-Database-

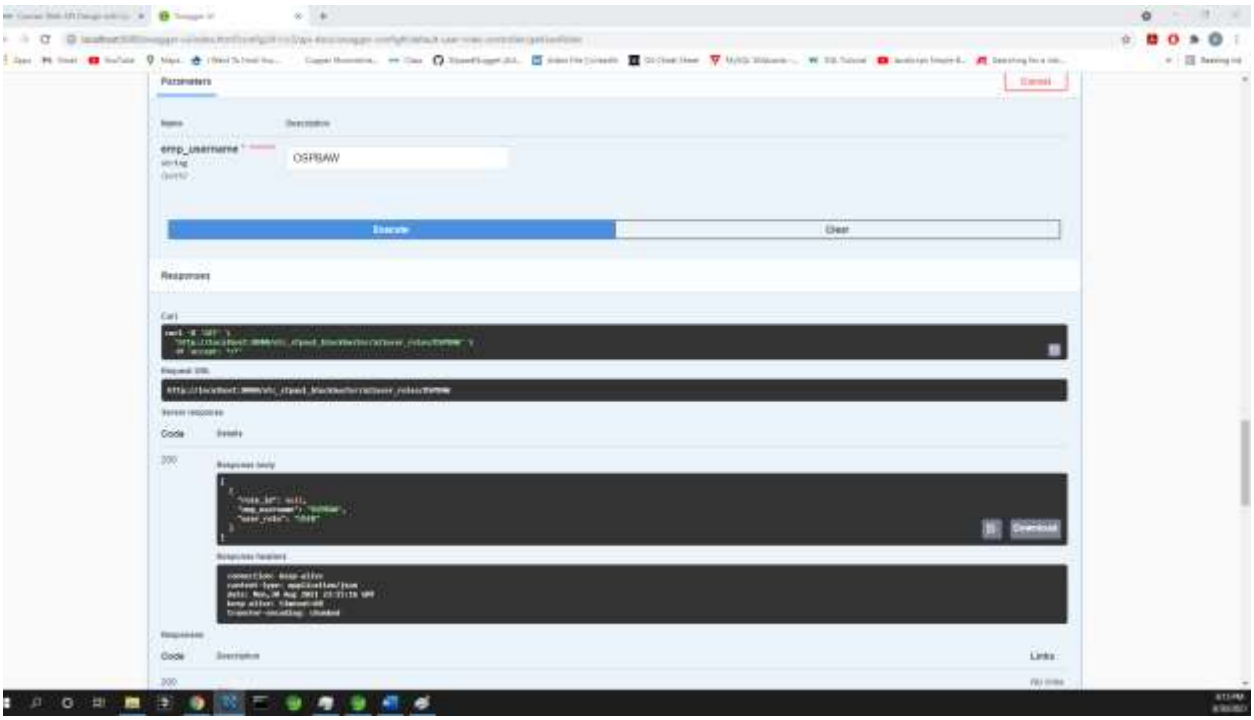
Result Grid | Filter Rows: | Edit:

role_id	emp_username	user_role
1	OSPBAF	USER
2	OSPBAF	ADMIN
3	OSPBCT	USER
4	OSPBCT	ADMIN
5	OSPBAC	USER
6	OSPBAC	ADMIN
7	OSPBTS	USER
8	OSPBFC	USER
9	OSPBJG	USER
10	OSPBJJ	USER
11	OSPBES	USER
12	OSPBWA	USER
13	OSPBAW	USER
NULL	NULL	NULL

Getting all employees of a certain role –

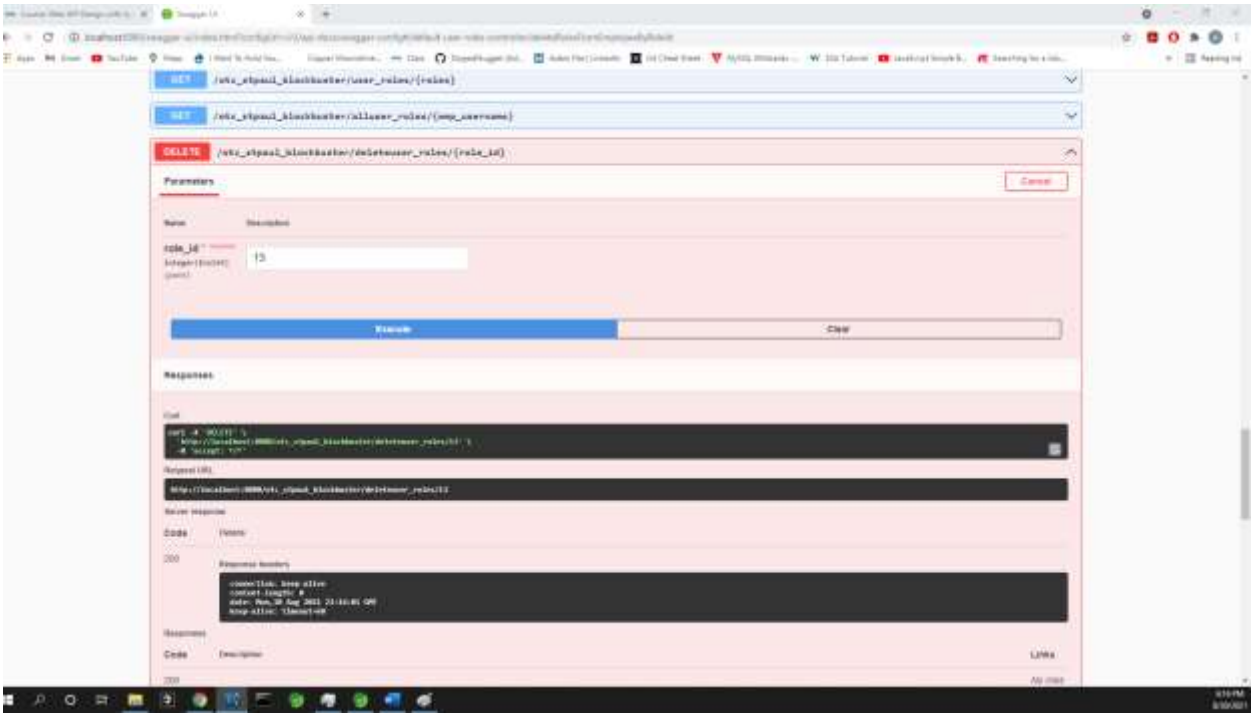


Getting an employees roles –



Deleting a role –

-Swagger-



-Database-

Result Grid | Filter Rows: | Edit:

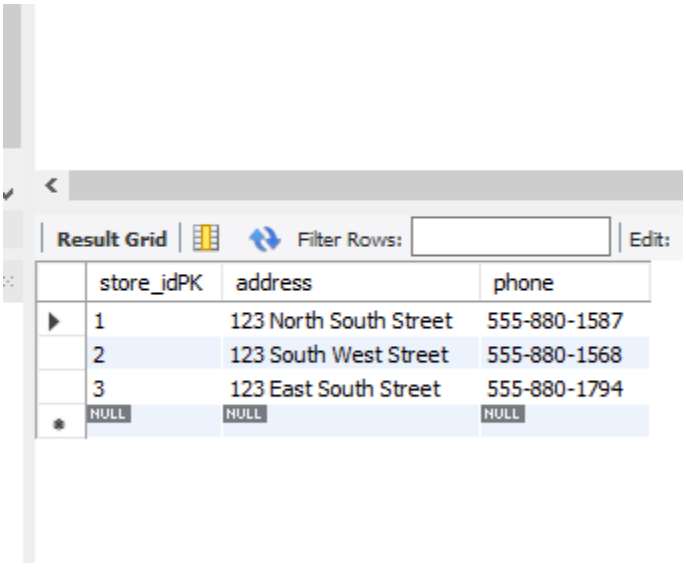
role_id	emp_username	user_role
1	OSPBAF	USER
2	OSPBAF	ADMIN
3	OSPBCT	USER
4	OSPBCT	ADMIN
5	OSPBAC	USER
6	OSPBAC	ADMIN
7	OSPBTS	USER
8	OSPBFC	USER
9	OSPBJG	USER
10	OSPBJJ	USER
11	OSPBES	USER
12	OSPBWA	USER
NULL	NULL	NULL

Stores:

Variables : 698 West East St 417-999-3646

Update variables : 705 W East Street 417-999-4578

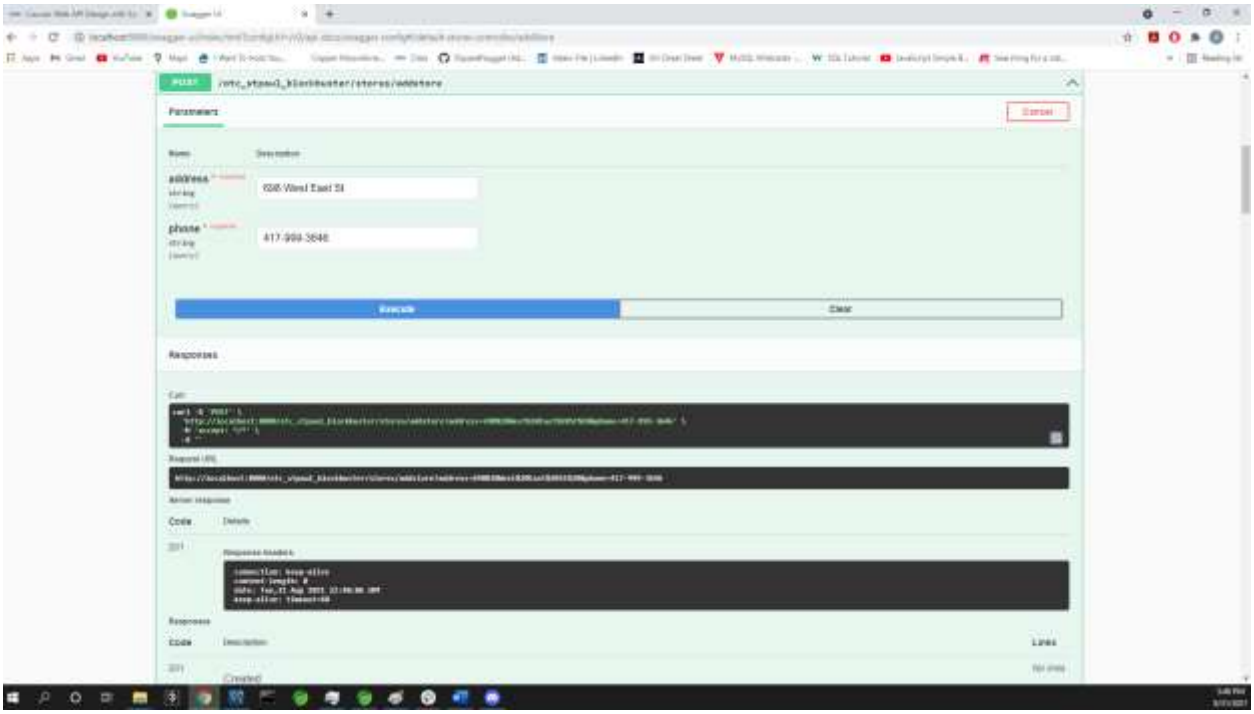
Starting Database –



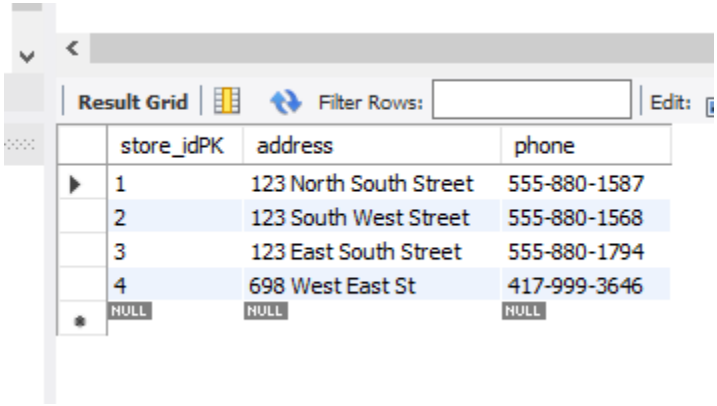
store_idPK	address	phone
1	123 North South Street	555-880-1587
2	123 South West Street	555-880-1568
3	123 East South Street	555-880-1794
NULL	NULL	NULL

Creating a Store-

-Swagger-

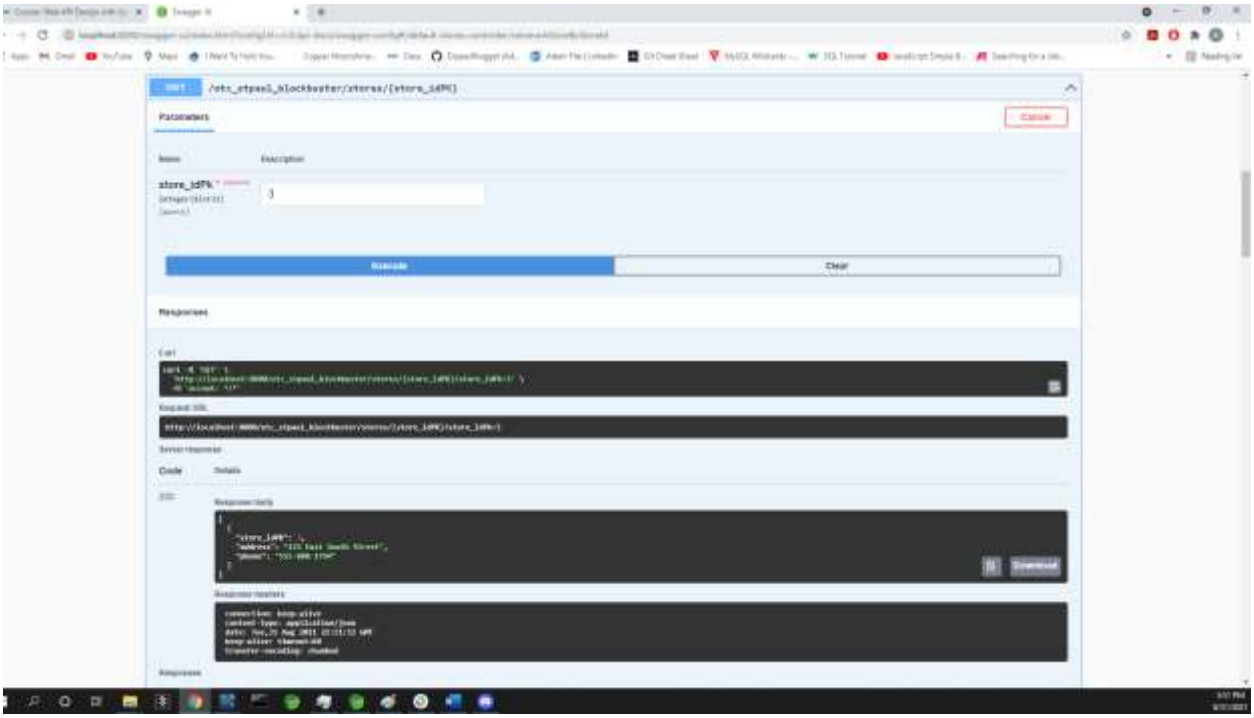


-Database-



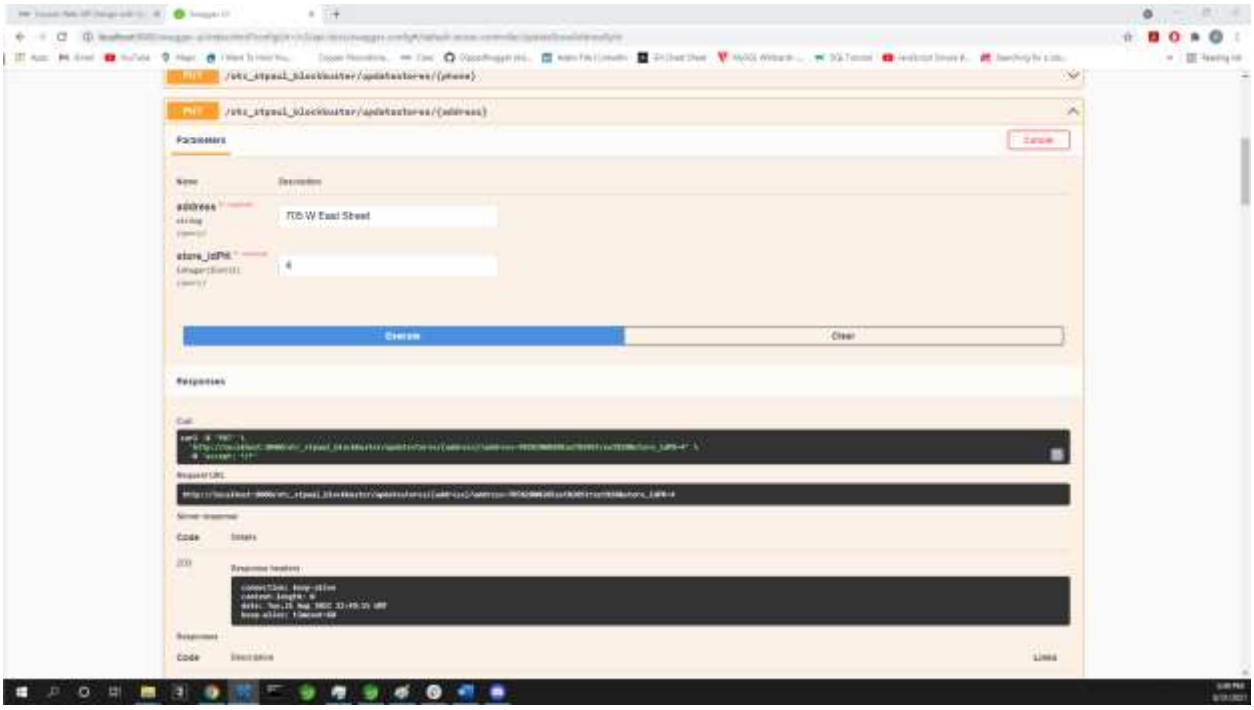
store_idPK	address	phone
1	123 North South Street	555-880-1587
2	123 South West Street	555-880-1568
3	123 East South Street	555-880-1794
4	698 West East St	417-999-3646
NULL	NULL	NULL

Getting a store –

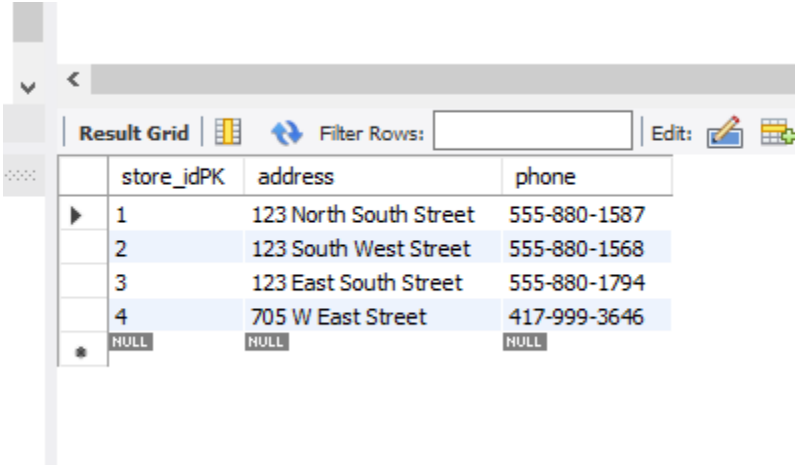


Update a store address-

-Swagger-



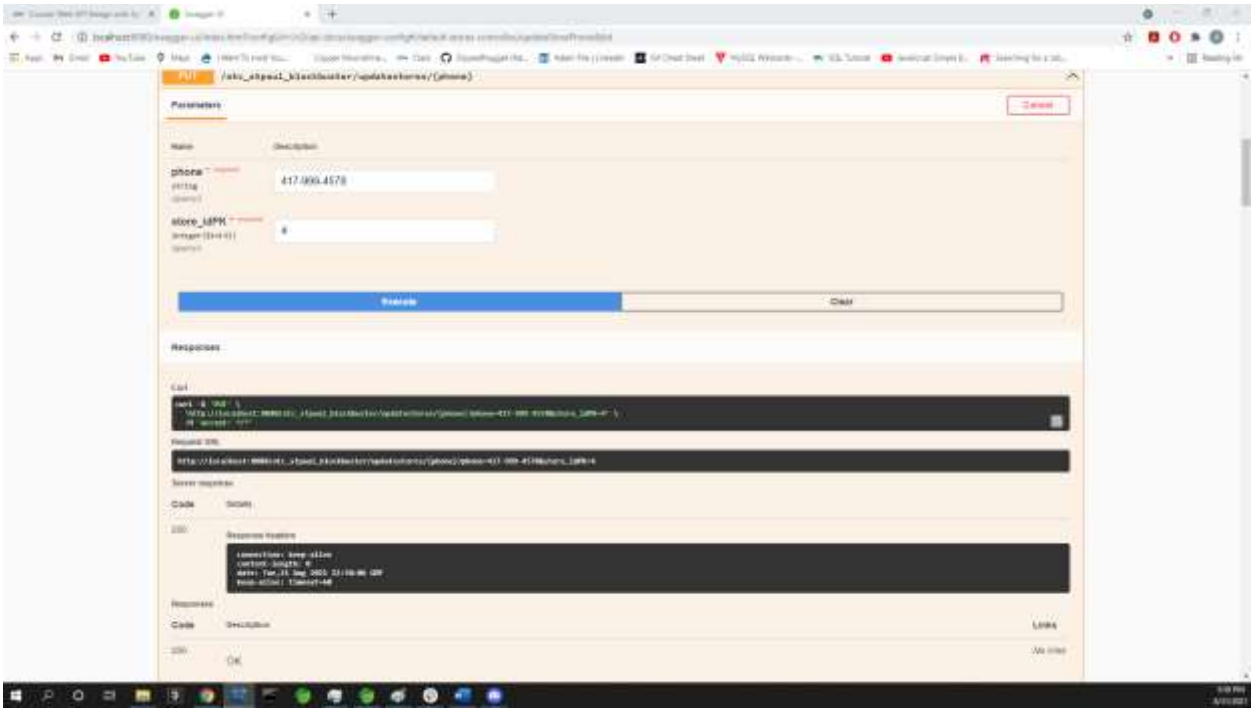
-Database-



store_idPK	address	phone
1	123 North South Street	555-880-1587
2	123 South West Street	555-880-1568
3	123 East South Street	555-880-1794
4	705 W East Street	417-999-3646
NULL	NULL	NULL

Update a store phone number-

-Swagger-



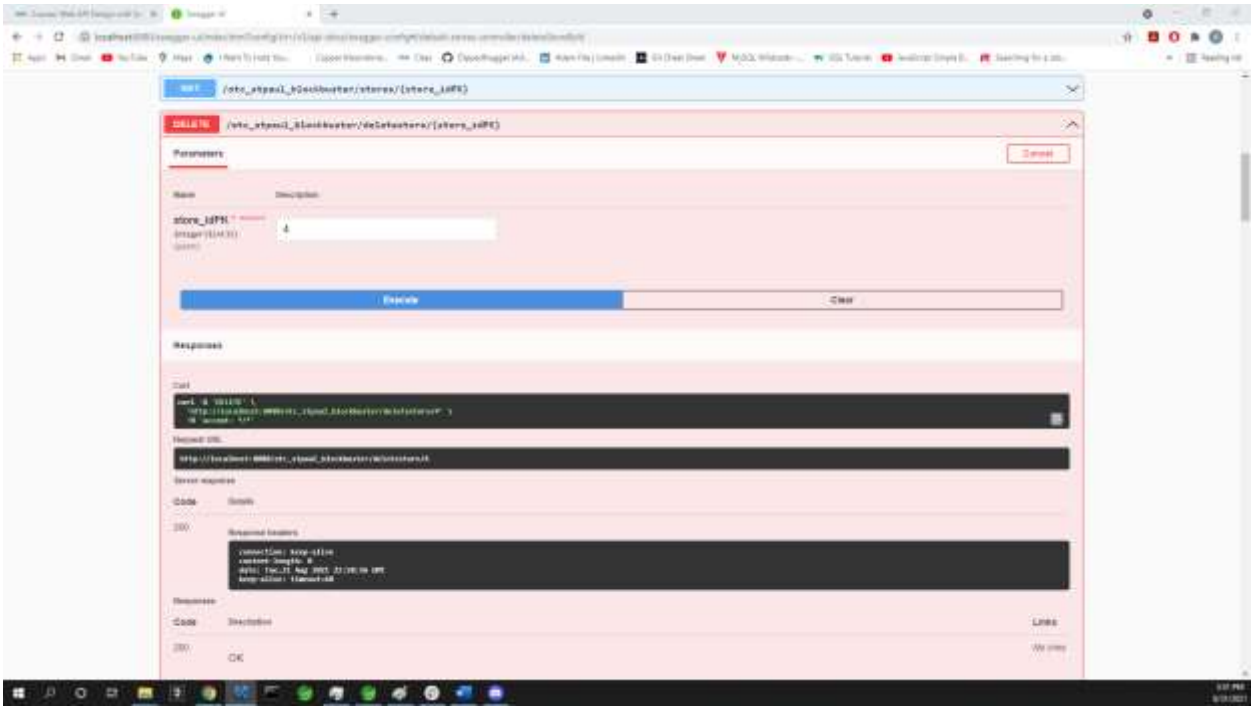
-Database-

Result Grid | Filter Rows: | Ed

store_idPK	address	phone
1	123 North South Street	555-880-1587
2	123 South West Street	555-880-1568
3	123 East South Street	555-880-1794
4	705 W East Street	417-999-4578
NULL	NULL	NULL

Delete a Store-

-Swagger-



-Database-

Result Grid | Filter Rows: | Edit:

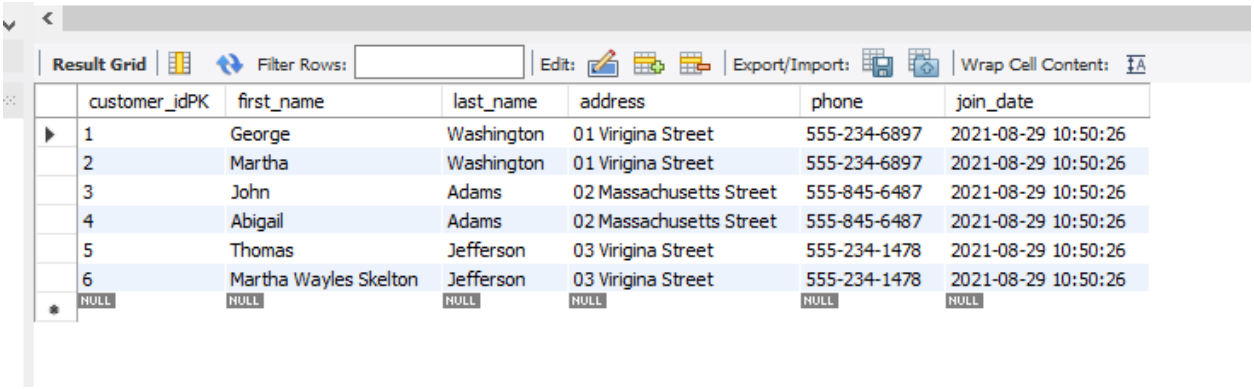
store_idPK	address	phone
1	123 North South Street	555-880-1587
2	123 South West Street	555-880-1568
3	123 East South Street	555-880-1794
NULL	NULL	NULL

Customers:

Variables: William Clinton 42 Arkansas Way 555-968-7945

Update variables: Bill Clamped 42 Brooklyn Way 555-753-2542

Starting Database –

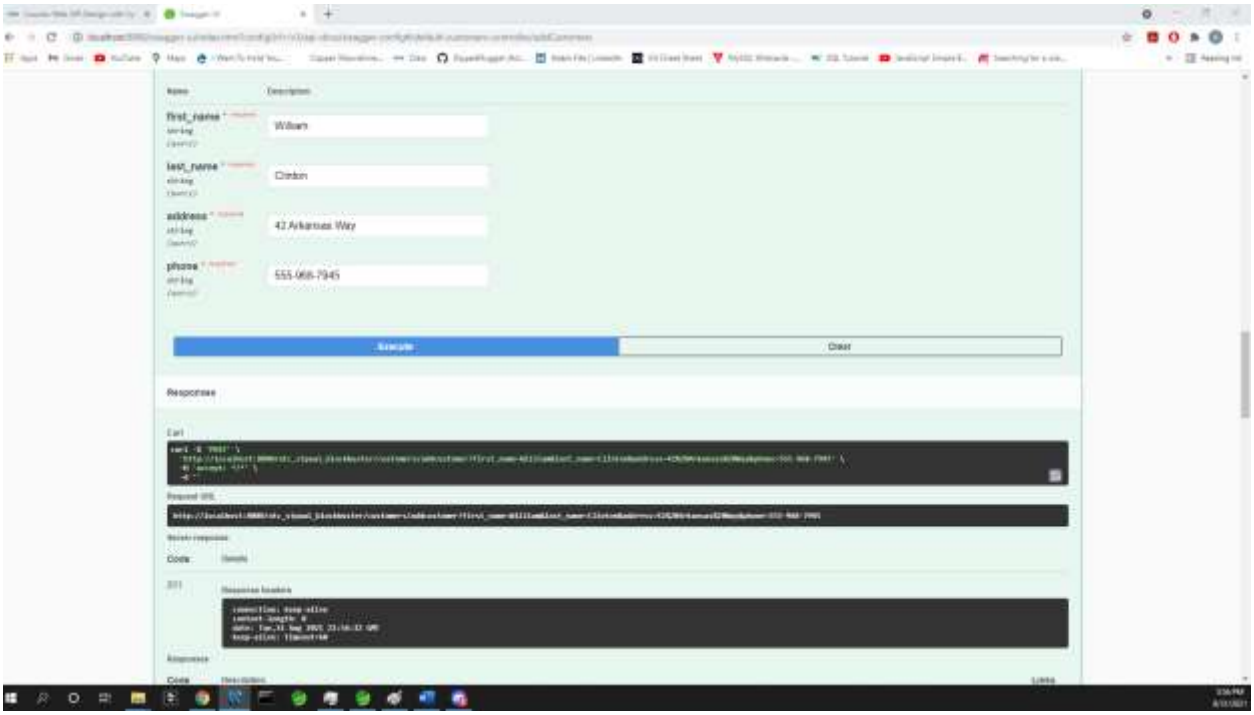


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

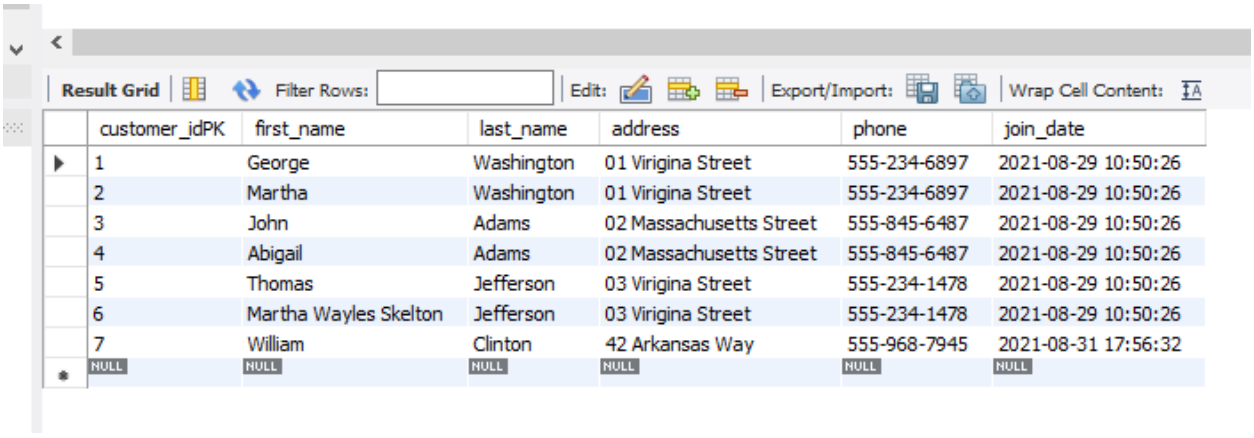
customer_idPK	first_name	last_name	address	phone	join_date
1	George	Washington	01 Virigina Street	555-234-6897	2021-08-29 10:50:26
2	Martha	Washington	01 Virigina Street	555-234-6897	2021-08-29 10:50:26
3	John	Adams	02 Massachusetts Street	555-845-6487	2021-08-29 10:50:26
4	Abigail	Adams	02 Massachusetts Street	555-845-6487	2021-08-29 10:50:26
5	Thomas	Jefferson	03 Virigina Street	555-234-1478	2021-08-29 10:50:26
6	Martha Wayles Skelton	Jefferson	03 Virigina Street	555-234-1478	2021-08-29 10:50:26
NULL	NULL	NULL	NULL	NULL	NULL

Create Customer-

-Swagger-



-Database-

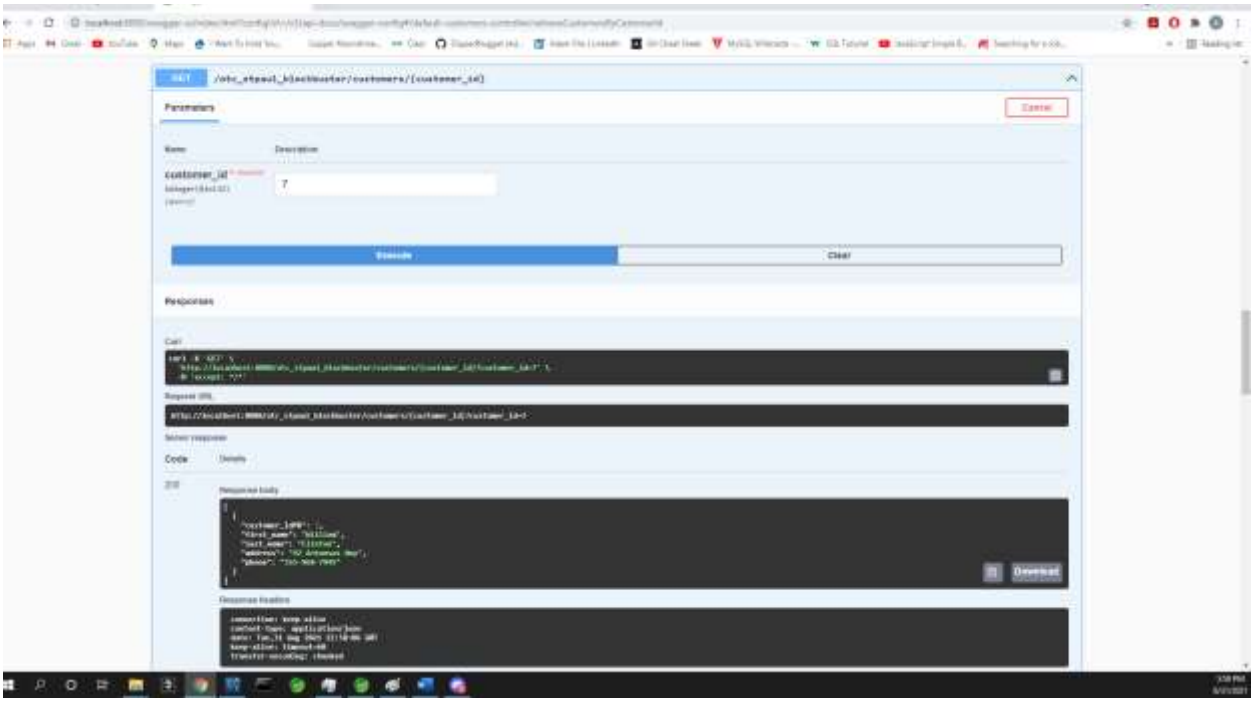


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_idPK	first_name	last_name	address	phone	join_date
1	George	Washington	01 Virigina Street	555-234-6897	2021-08-29 10:50:26
2	Martha	Washington	01 Virigina Street	555-234-6897	2021-08-29 10:50:26
3	John	Adams	02 Massachusetts Street	555-845-6487	2021-08-29 10:50:26
4	Abigail	Adams	02 Massachusetts Street	555-845-6487	2021-08-29 10:50:26
5	Thomas	Jefferson	03 Virigina Street	555-234-1478	2021-08-29 10:50:26
6	Martha Wayles Skelton	Jefferson	03 Virigina Street	555-234-1478	2021-08-29 10:50:26
7	William	Clinton	42 Arkansas Way	555-968-7945	2021-08-31 17:56:32
NULL	NULL	NULL	NULL	NULL	NULL

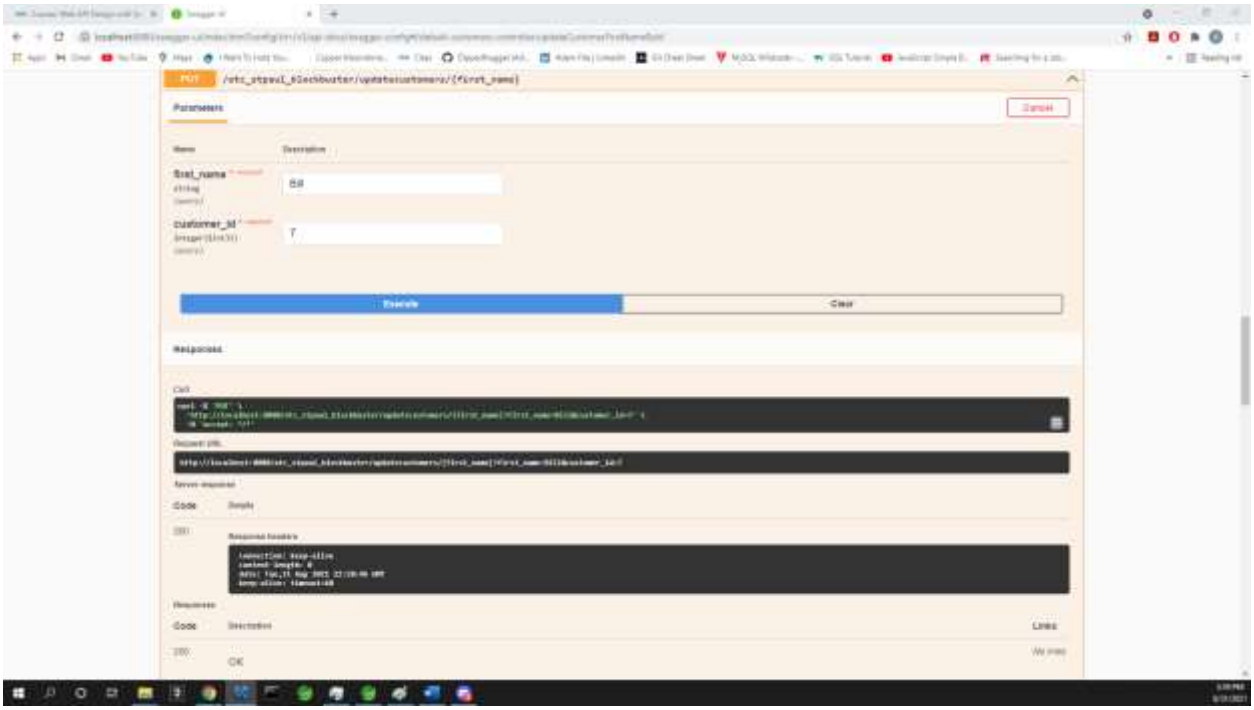
Get Customer-

-Swagger-



Update First Name-

-Swagger-



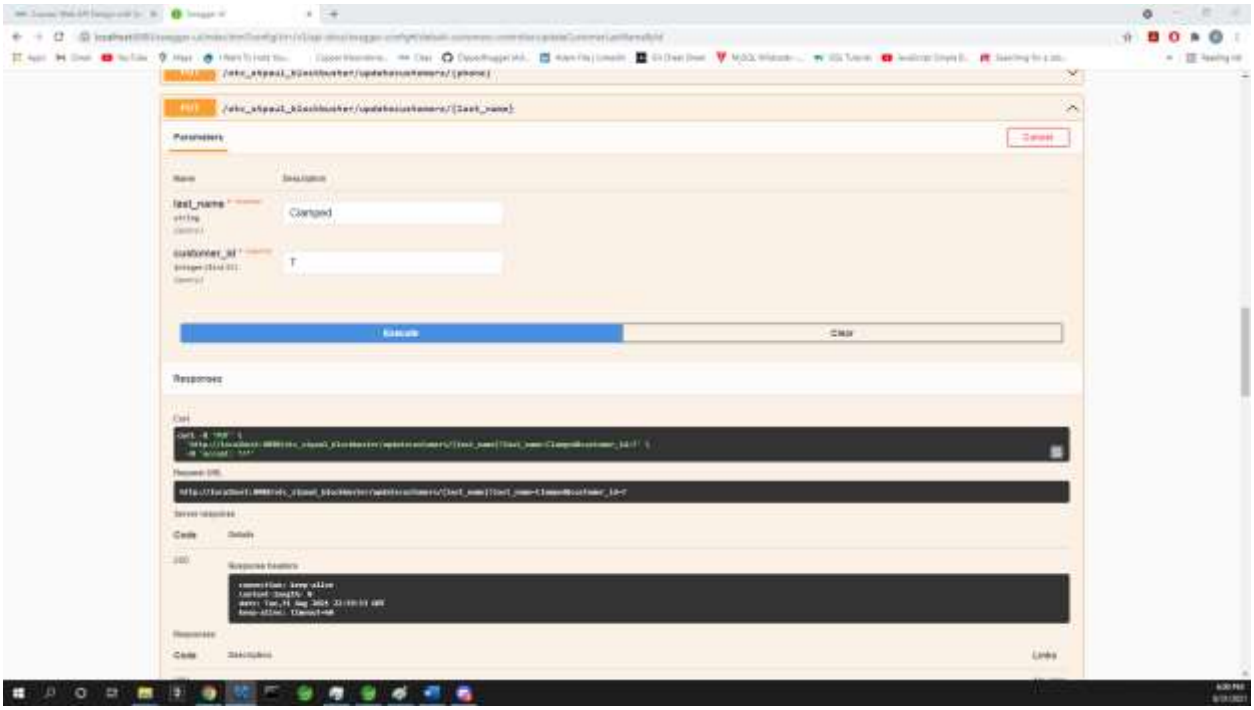
-Database-

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_idPK	first_name	last_name	address	phone	join_date
1	George	Washington	01 Virigina Street	555-234-6897	2021-08-29 10:50:26
2	Martha	Washington	01 Virigina Street	555-234-6897	2021-08-29 10:50:26
3	John	Adams	02 Massachusetts Street	555-845-6487	2021-08-29 10:50:26
4	Abigail	Adams	02 Massachusetts Street	555-845-6487	2021-08-29 10:50:26
5	Thomas	Jefferson	03 Virigina Street	555-234-1478	2021-08-29 10:50:26
6	Martha Wayles Skelton	Jefferson	03 Virigina Street	555-234-1478	2021-08-29 10:50:26
7	Bill	Clinton	42 Arkansas Way	555-968-7945	2021-08-31 17:56:32
NULL	NULL	NULL	NULL	NULL	NULL

Update Last Name –

-Swagger-



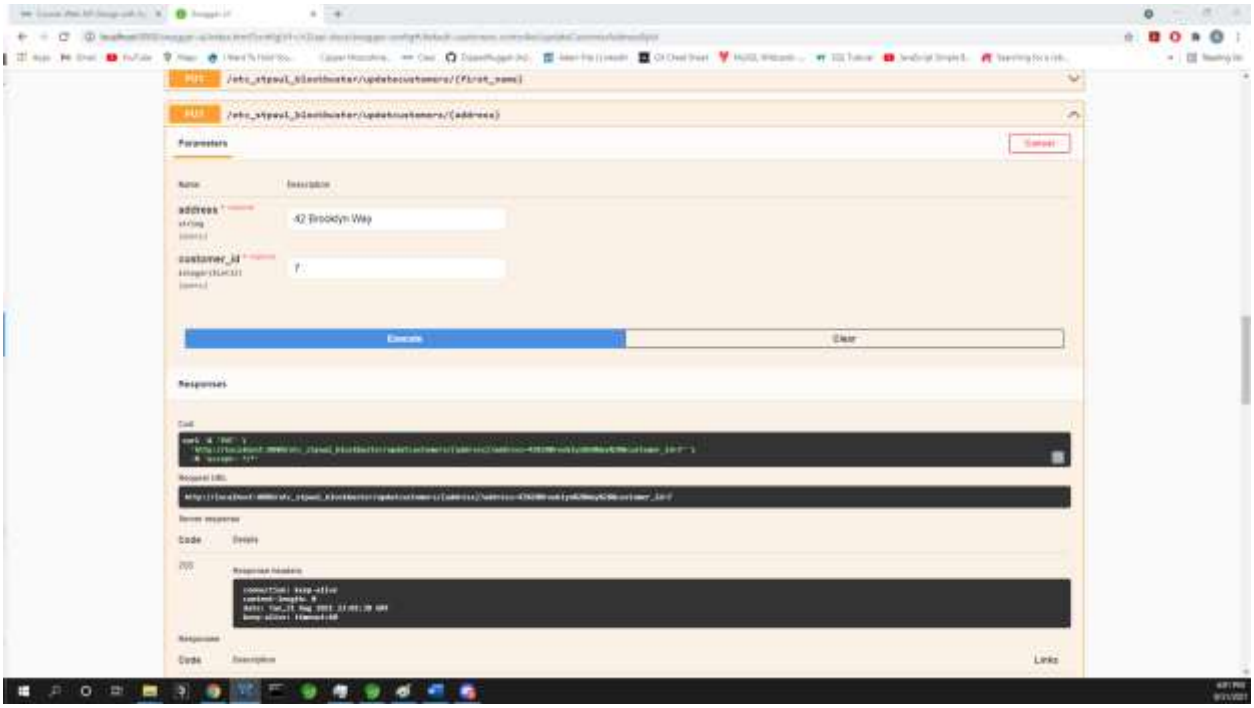
-Database-

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

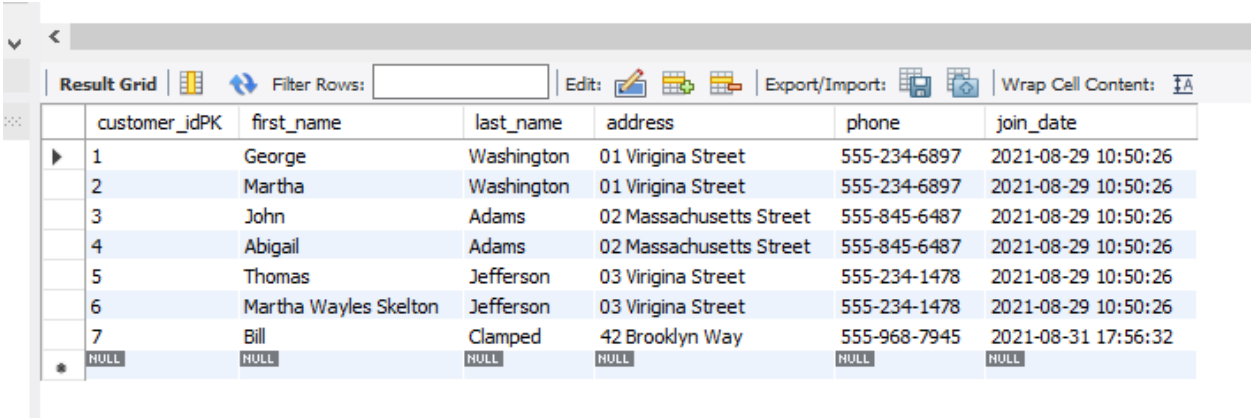
customer_idPK	first_name	last_name	address	phone	join_date
1	George	Washington	01 Virigina Street	555-234-6897	2021-08-29 10:50:26
2	Martha	Washington	01 Virigina Street	555-234-6897	2021-08-29 10:50:26
3	John	Adams	02 Massachusetts Street	555-845-6487	2021-08-29 10:50:26
4	Abigail	Adams	02 Massachusetts Street	555-845-6487	2021-08-29 10:50:26
5	Thomas	Jefferson	03 Virigina Street	555-234-1478	2021-08-29 10:50:26
6	Martha Wayles Skelton	Jefferson	03 Virigina Street	555-234-1478	2021-08-29 10:50:26
7	Bill	Clamped	42 Arkansas Way	555-968-7945	2021-08-31 17:56:32
NULL	NULL	NULL	NULL	NULL	NULL

Update Address –

-Swagger-



-Database-

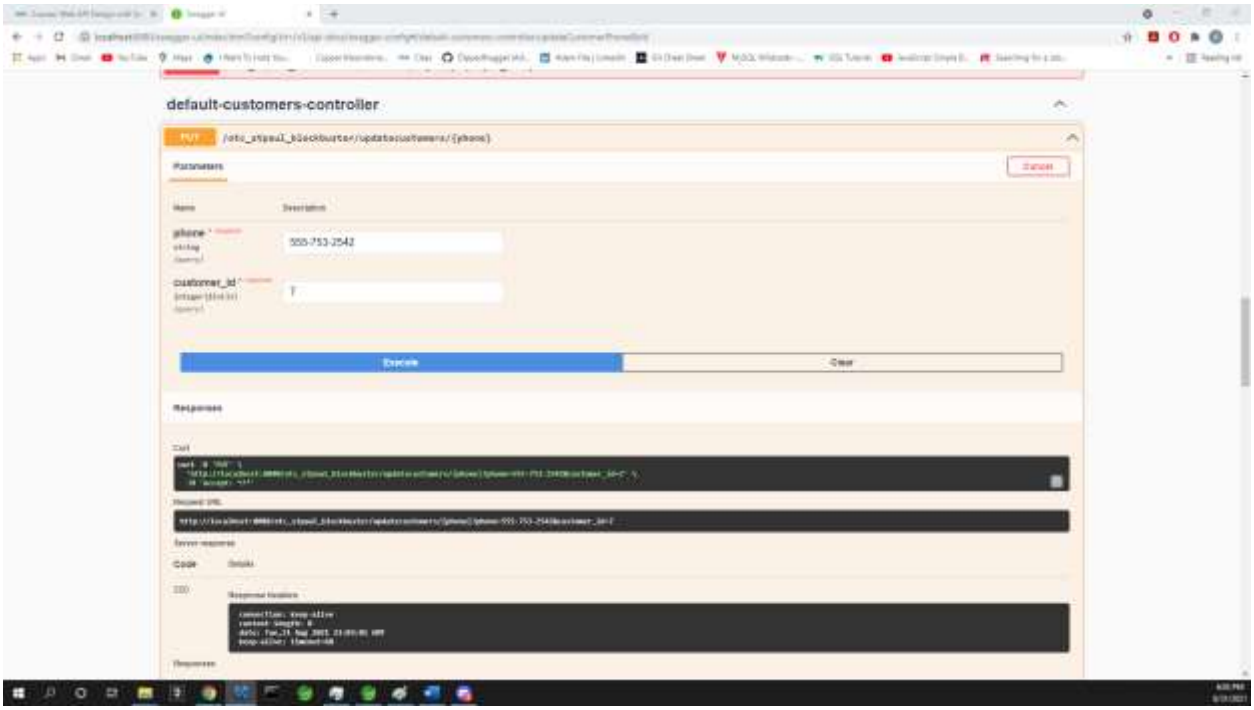


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

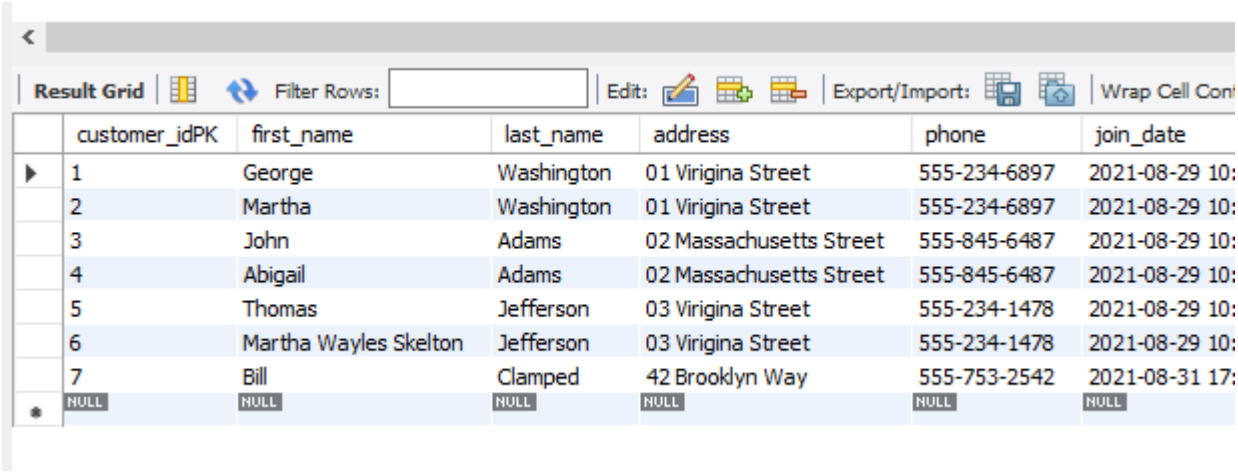
customer_idPK	first_name	last_name	address	phone	join_date
1	George	Washington	01 Virigina Street	555-234-6897	2021-08-29 10:50:26
2	Martha	Washington	01 Virigina Street	555-234-6897	2021-08-29 10:50:26
3	John	Adams	02 Massachusetts Street	555-845-6487	2021-08-29 10:50:26
4	Abigail	Adams	02 Massachusetts Street	555-845-6487	2021-08-29 10:50:26
5	Thomas	Jefferson	03 Virigina Street	555-234-1478	2021-08-29 10:50:26
6	Martha Wayles Skelton	Jefferson	03 Virigina Street	555-234-1478	2021-08-29 10:50:26
7	Bill	Clamped	42 Brooklyn Way	555-968-7945	2021-08-31 17:56:32
NULL	NULL	NULL	NULL	NULL	NULL

Update Phone Number-

-Swagger-



-Database-

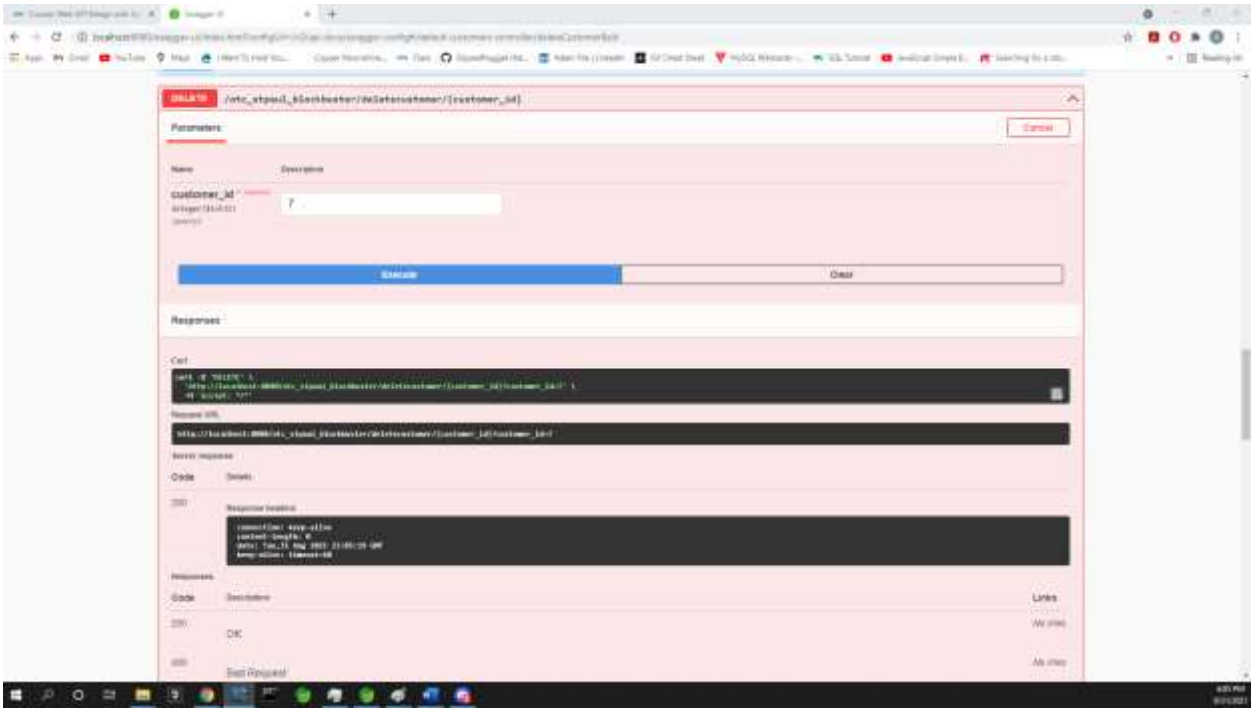


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Cont

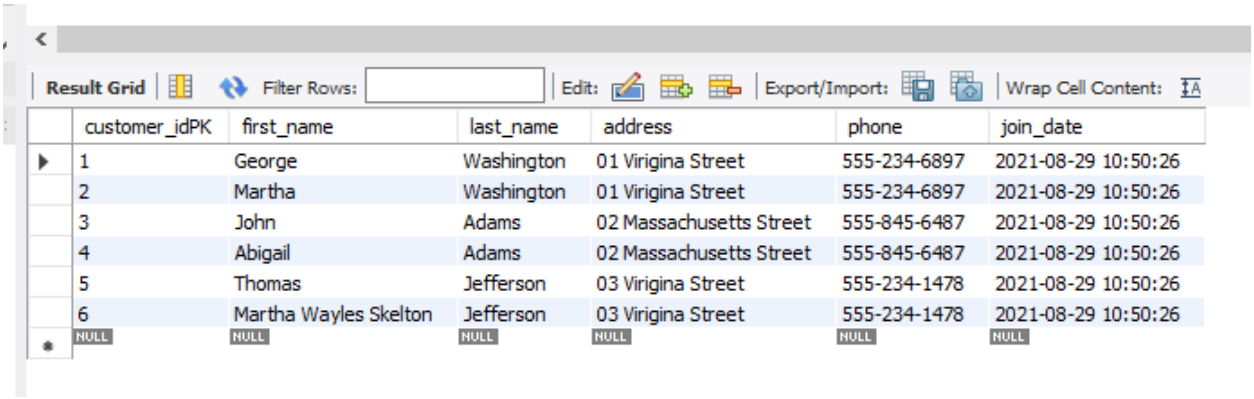
customer_idPK	first_name	last_name	address	phone	join_date
1	George	Washington	01 Virigina Street	555-234-6897	2021-08-29 10:
2	Martha	Washington	01 Virigina Street	555-234-6897	2021-08-29 10:
3	John	Adams	02 Massachusetts Street	555-845-6487	2021-08-29 10:
4	Abigail	Adams	02 Massachusetts Street	555-845-6487	2021-08-29 10:
5	Thomas	Jefferson	03 Virigina Street	555-234-1478	2021-08-29 10:
6	Martha Wayles Skelton	Jefferson	03 Virigina Street	555-234-1478	2021-08-29 10:
7	Bill	Clamped	42 Brooklyn Way	555-753-2542	2021-08-31 17:
NULL	NULL	NULL	NULL	NULL	NULL

Delete a Customer-

-Swagger-



-Database-



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

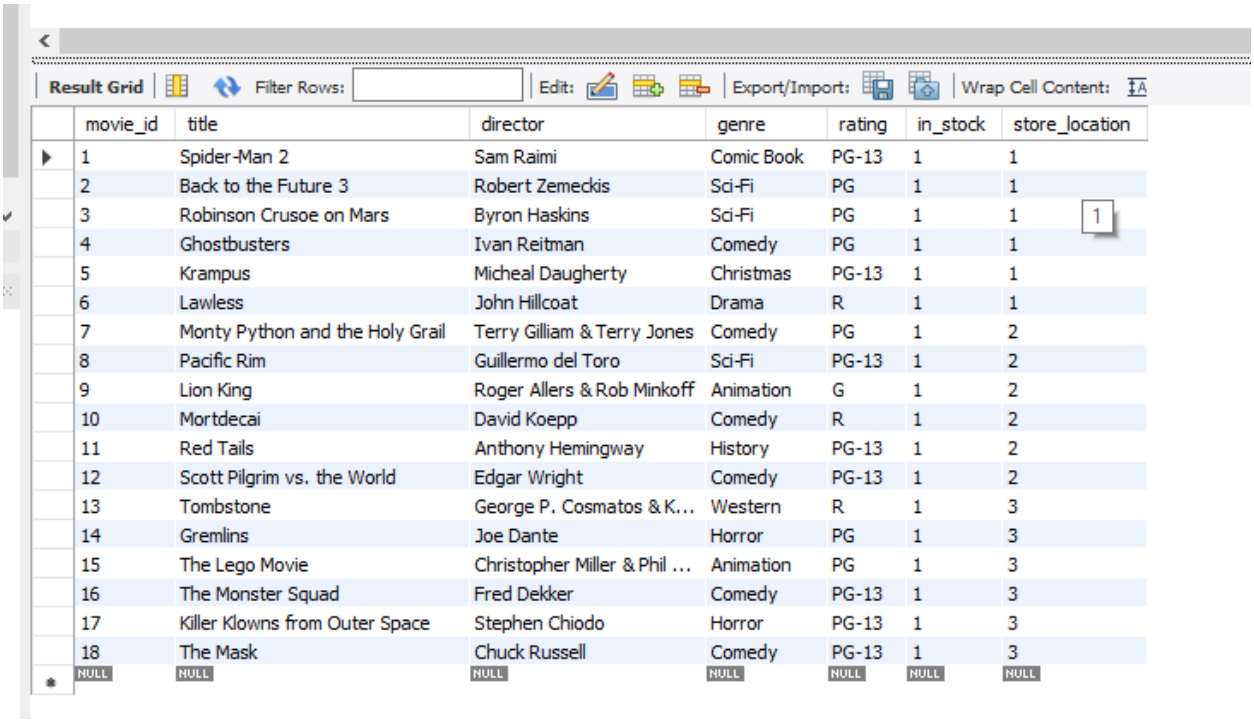
customer_idPK	first_name	last_name	address	phone	join_date
1	George	Washington	01 Virigina Street	555-234-6897	2021-08-29 10:50:26
2	Martha	Washington	01 Virigina Street	555-234-6897	2021-08-29 10:50:26
3	John	Adams	02 Massachusetts Street	555-845-6487	2021-08-29 10:50:26
4	Abigail	Adams	02 Massachusetts Street	555-845-6487	2021-08-29 10:50:26
5	Thomas	Jefferson	03 Virigina Street	555-234-1478	2021-08-29 10:50:26
6	Martha Wayles Skelton	Jefferson	03 Virigina Street	555-234-1478	2021-08-29 10:50:26
NULL	NULL	NULL	NULL	NULL	NULL

Movies:

Variable: Suicide Squad David Ayer Comic Book PG-13 03

Update variables: The Suicide Squad James Gunn Action/Adventure R 01

Starting Database –

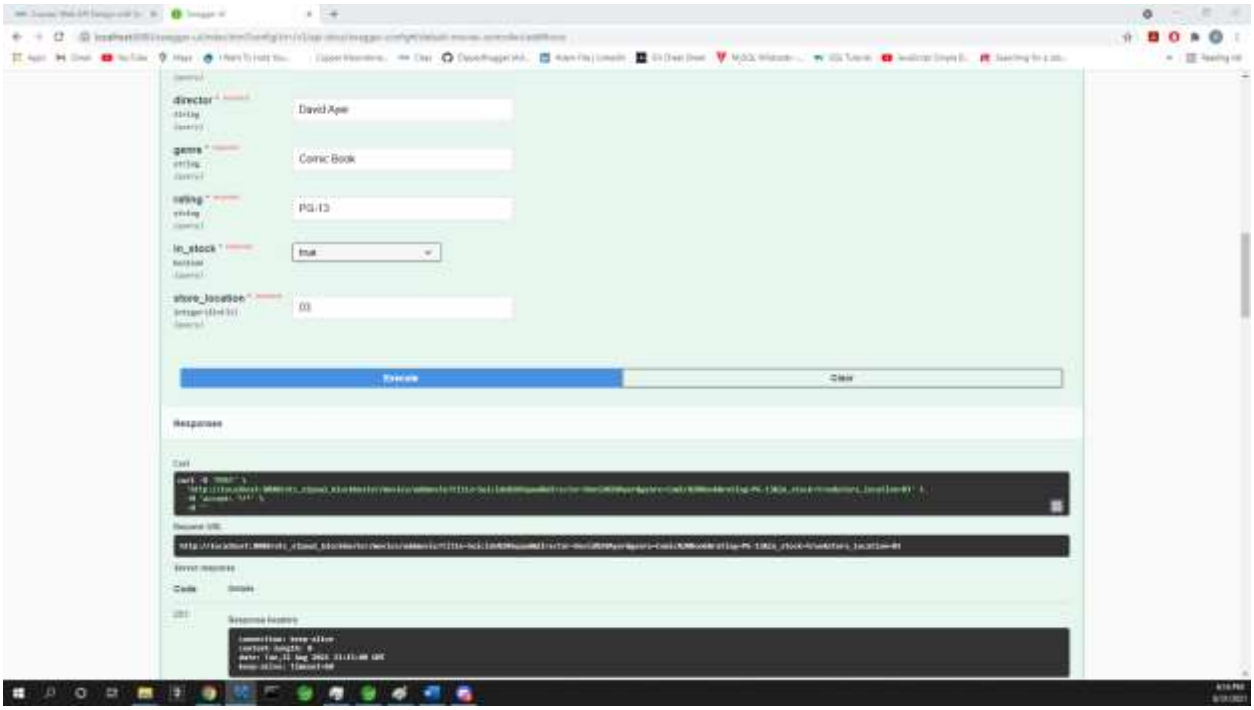


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

movie_id	title	director	genre	rating	in_stock	store_location
1	Spider-Man 2	Sam Raimi	Comic Book	PG-13	1	1
2	Back to the Future 3	Robert Zemeckis	Sci-Fi	PG	1	1
3	Robinson Crusoe on Mars	Byron Haskins	Sci-Fi	PG	1	1
4	Ghostbusters	Ivan Reitman	Comedy	PG	1	1
5	Krampus	Micheal Daugherty	Christmas	PG-13	1	1
6	Lawless	John Hillcoat	Drama	R	1	1
7	Monty Python and the Holy Grail	Terry Gilliam & Terry Jones	Comedy	PG	1	2
8	Pacific Rim	Guillermo del Toro	Sci-Fi	PG-13	1	2
9	Lion King	Roger Allers & Rob Minkoff	Animation	G	1	2
10	Mortdecai	David Koepp	Comedy	R	1	2
11	Red Tails	Anthony Hemingway	History	PG-13	1	2
12	Scott Pilgrim vs. the World	Edgar Wright	Comedy	PG-13	1	2
13	Tombstone	George P. Cosmatos & K...	Western	R	1	3
14	Gremlins	Joe Dante	Horror	PG	1	3
15	The Lego Movie	Christopher Miller & Phil ...	Animation	PG	1	3
16	The Monster Squad	Fred Dekker	Comedy	PG-13	1	3
17	Killer Klowns from Outer Space	Stephen Chiodo	Horror	PG-13	1	3
18	The Mask	Chuck Russell	Comedy	PG-13	1	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Create a movie –

-Swagger-

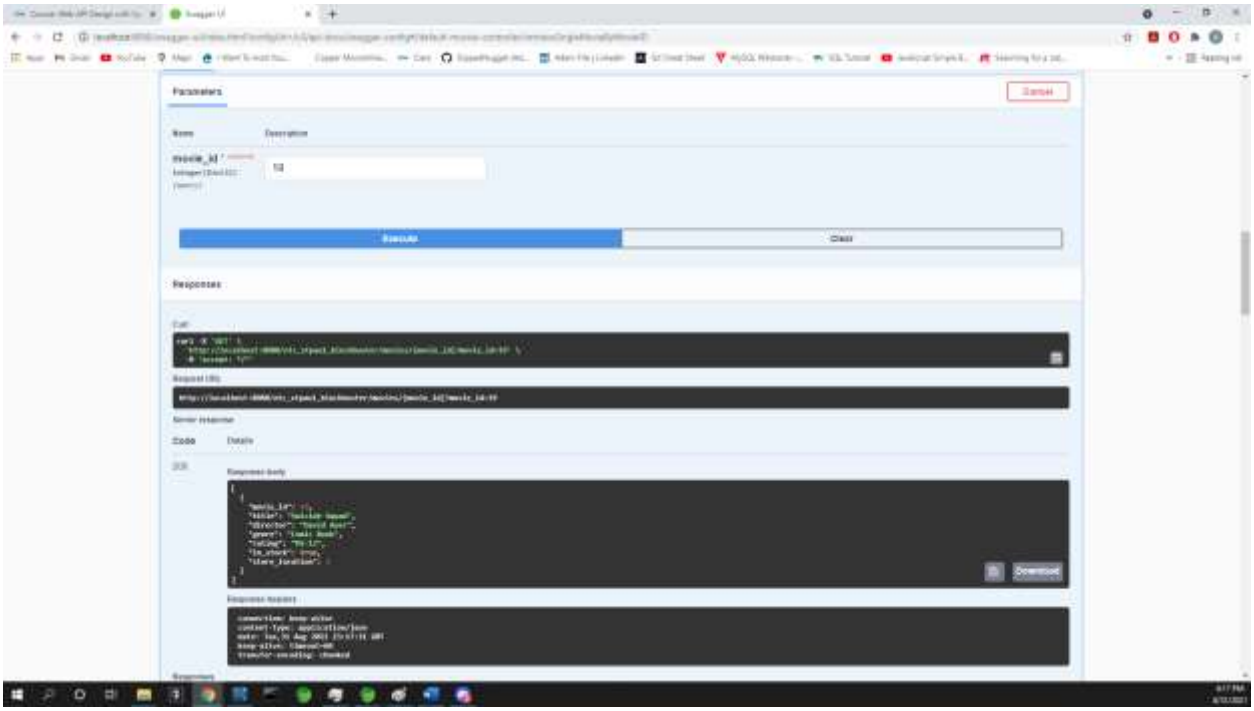


-Database-

	18	The Mask	Chuck Russell	Comedy	PG-13	1	3
	19	Suicide Squad	David Ayer	Comic Book	PG-13	1	3
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

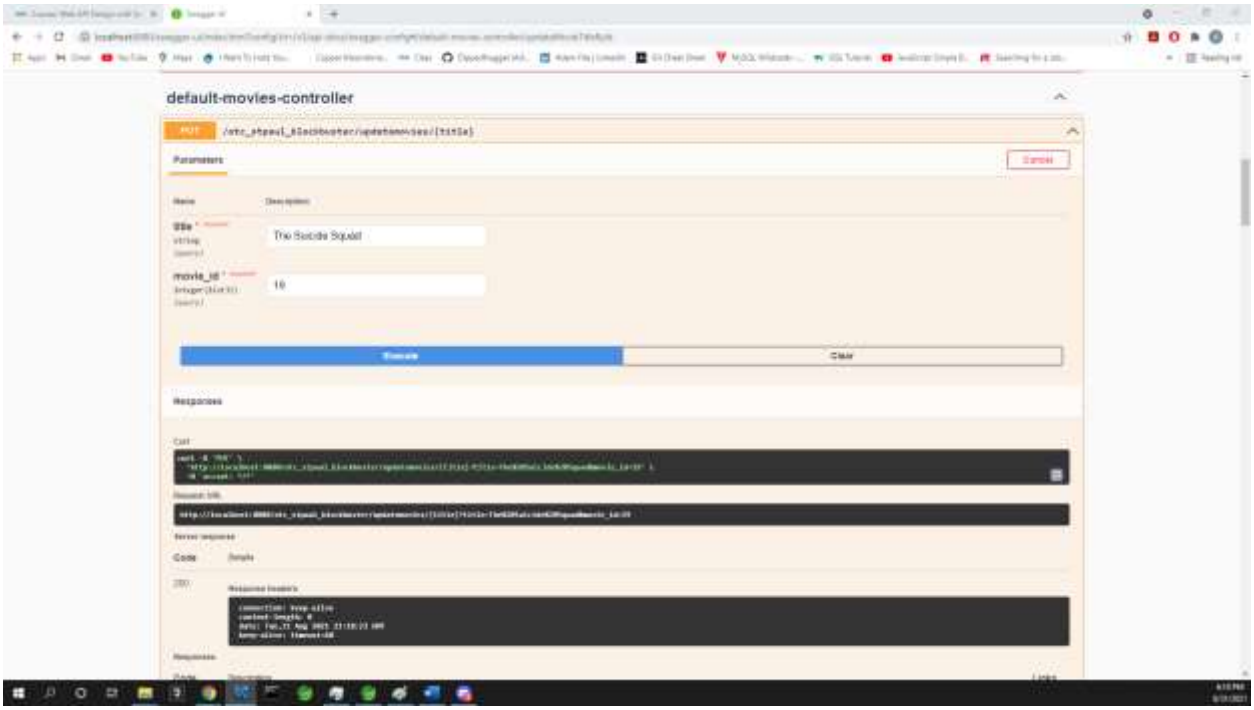
Get a movie –

-Swagger-



Update a Title-

-Swagger-



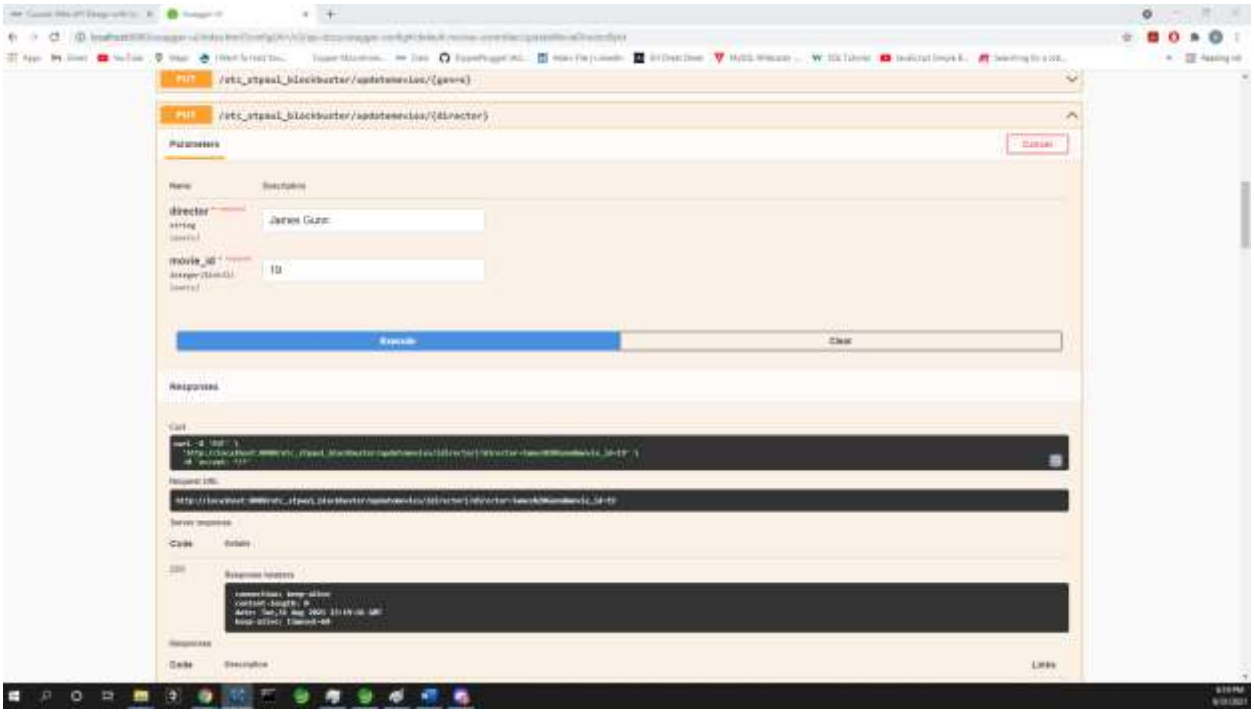
-Database-

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

movie_id	title	director	genre	rating	in_stock	store_location
1	Spider-Man 2	Sam Raimi	Comic Book	PG-13	1	1
2	Back to the Future 3	Robert Zemeckis	Sci-Fi	PG	1	1
3	Robinson Crusoe on Mars	Byron Haskins	Sci-Fi	PG	1	1
4	Ghostbusters	Ivan Reitman	Comedy	PG	1	1
5	Krampus	Micheal Daugherty	Christmas	PG-13	1	1
6	Lawless	John Hillcoat	Drama	R	1	1
7	Monty Python and the Holy Grail	Terry Gilliam & Terry Jones	Comedy	PG	1	2
8	Pacific Rim	Guillermo del Toro	Sci-Fi	PG-13	1	2
9	Lion King	Roger Allers & Rob Minkoff	Animation	G	1	2
10	Mortdecai	David Koepp	Comedy	R	1	2
11	Red Tails	Anthony Hemingway	History	PG-13	1	2
12	Scott Pilgrim vs. the World	Edgar Wright	Comedy	PG-13	1	2
13	Tombstone	George P. Cosmatos & K...	Western	R	1	3
14	Gremlins	Joe Dante	Horror	PG	1	3
15	The Lego Movie	Christopher Miller & Phil ...	Animation	PG	1	3
16	The Monster Squad	Fred Dekker	Comedy	PG-13	1	3
17	Killer Klowns from Outer Space	Stephen Chiodo	Horror	PG-13	1	3
18	The Mask	Chuck Russell	Comedy	PG-13	1	3
19	The Suicide Squad	David Ayer	Comic Book	PG-13	1	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Update a Director –

-Swagger-



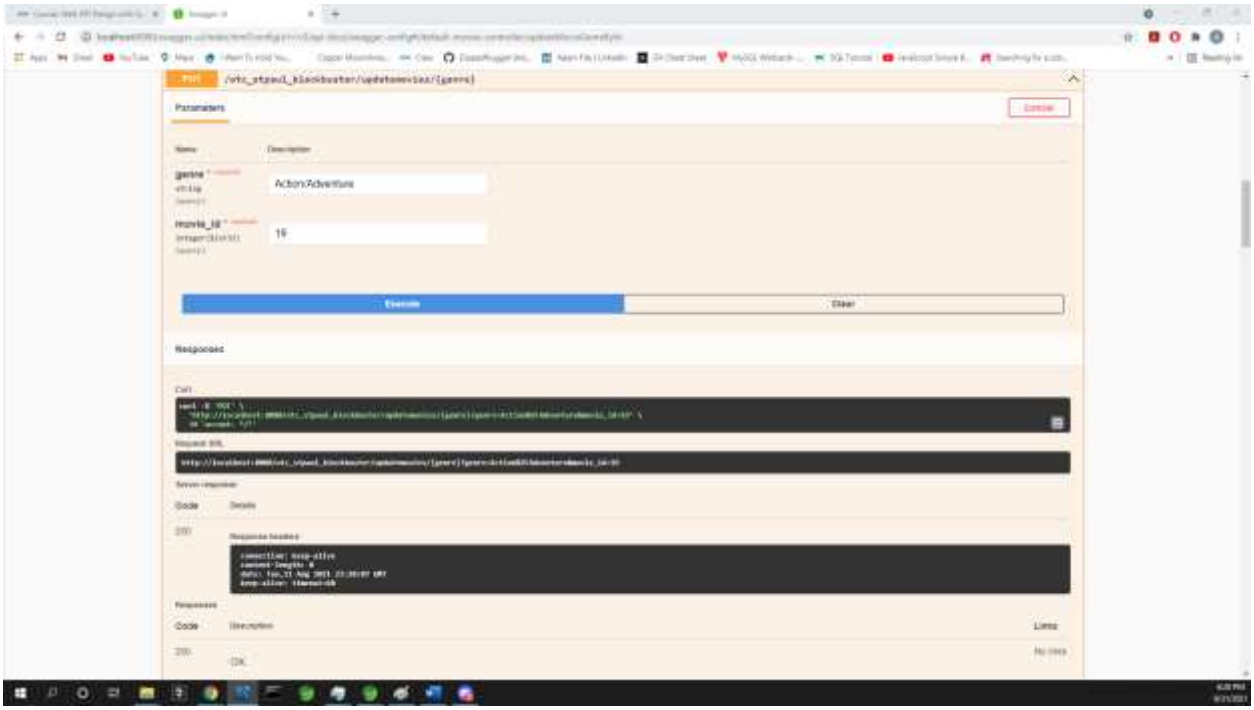
-Database-

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

movie_id	title	director	genre	rating	in_stock	store_location
1	Spider-Man 2	Spider-Man 2	Comic Book	PG-13	1	1
2	Back to the Future 3	Robert Zemeckis	Sci-Fi	PG	1	1
3	Robinson Crusoe on Mars	Byron Haskins	Sci-Fi	PG	1	1
4	Ghostbusters	Ivan Reitman	Comedy	PG	1	1
5	Krampus	Micheal Daugherty	Christmas	PG-13	1	1
6	Lawless	John Hillcoat	Drama	R	1	1
7	Monty Python and the Holy Grail	Terry Gilliam & Terry Jones	Comedy	PG	1	2
8	Pacific Rim	Guillermo del Toro	Sci-Fi	PG-13	1	2
9	Lion King	Roger Allers & Rob Minkoff	Animation	G	1	2
10	Mortdecai	David Koepp	Comedy	R	1	2
11	Red Tails	Anthony Hemingway	History	PG-13	1	2
12	Scott Pilgrim vs. the World	Edgar Wright	Comedy	PG-13	1	2
13	Tombstone	George P. Cosmatos & K...	Western	R	1	3
14	Gremlins	Joe Dante	Horror	PG	1	3
15	The Lego Movie	Christopher Miller & Phil ...	Animation	PG	1	3
16	The Monster Squad	Fred Dekker	Comedy	PG-13	1	3
17	Killer Klowns from Outer Space	Stephen Chiodo	Horror	PG-13	1	3
18	The Mask	Chuck Russell	Comedy	PG-13	1	3
19	The Suicide Squad	James Gunn	Comic Book	PG-13	1	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Update a Genre –

-Swagger-



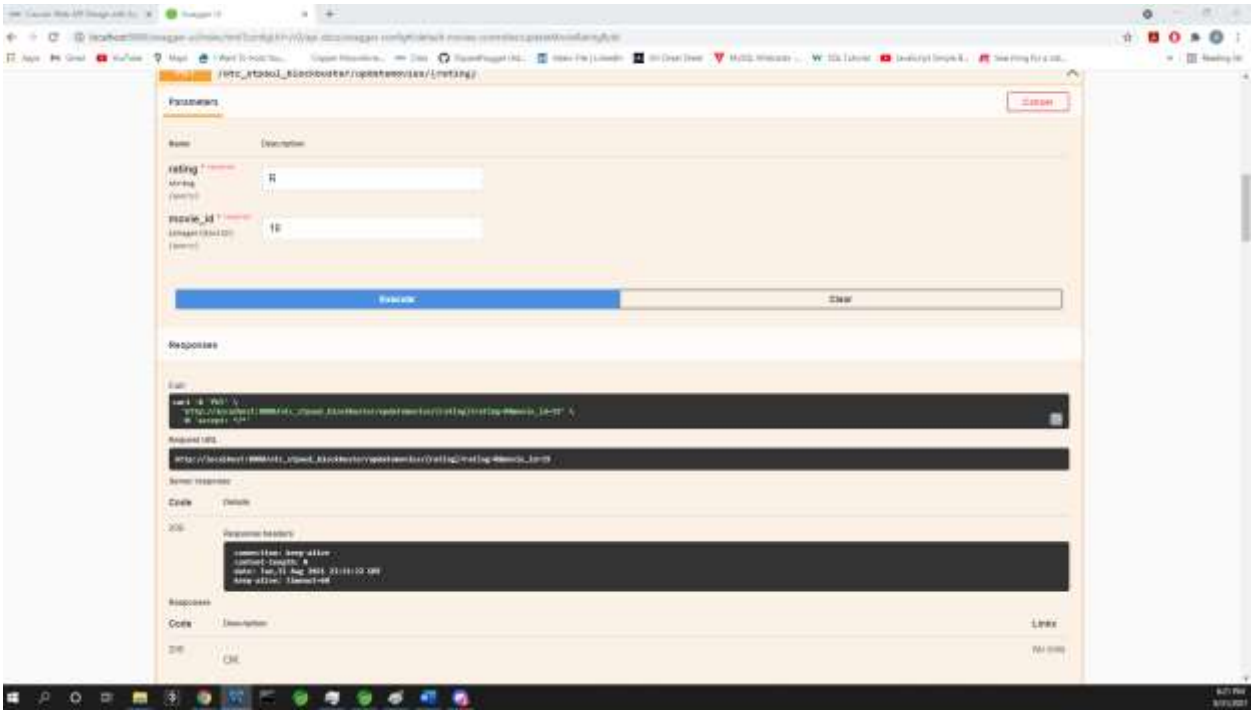
-Database-

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

movie_id	title	director	genre	rating	in_stock	store_location
1	Spider-Man 2	Sam Raimi	Comic Book	PG-13	1	1
2	Back to the Future 3	Robert Zemeckis	Sci-Fi	PG	1	1
3	Robinson Crusoe on Mars	Byron Haskins	Sci-Fi	PG	1	1
4	Ghostbusters	Ivan Reitman	Comedy	PG	1	1
5	Krampus	Micheal Daugherty	Christmas	PG-13	1	1
6	Lawless	John Hillcoat	Drama	R	1	1
7	Monty Python and the Holy Grail	Terry Gilliam & Terry Jones	Comedy	PG	1	2
8	Pacific Rim	Guillermo del Toro	Sci-Fi	PG-13	1	2
9	Lion King	Roger Allers & Rob Minkoff	Animation	G	1	2
10	Mortdecai	David Koepp	Comedy	R	1	2
11	Red Tails	Anthony Hemingway	History	PG-13	1	2
12	Scott Pilgrim vs. the World	Edgar Wright	Comedy	PG-13	1	2
13	Tombstone	George P. Cosmatos & K...	Western	R	1	3
14	Gremlins	Joe Dante	Horror	PG	1	3
15	The Lego Movie	Christopher Miller & Phil ...	Animation	PG	1	3
16	The Monster Squad	Fred Dekker	Comedy	PG-13	1	3
17	Killer Klowns from Outer Space	Stephen Chiodo	Horror	PG-13	1	3
18	The Mask	Chuck Russell	Comedy	PG-13	1	3
19	The Suicide Squad	James Gunn	Action/Adventure	PG-13	1	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Update a Rating

-Swagger-



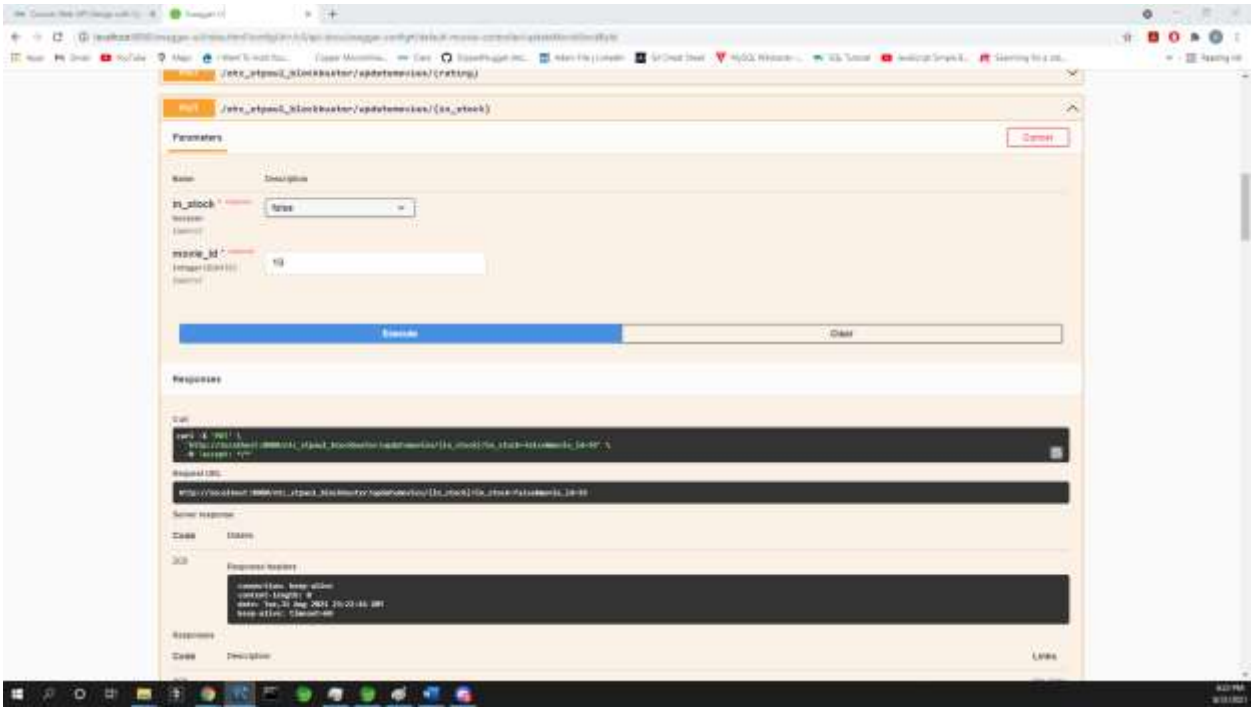
-Database-

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

movie_id	title	director	genre	rating	in_stock	store_location
1	Spider-Man 2	Sam Raimi	Comic Book	PG-13	1	1
2	Back to the Future 3	Robert Zemeckis	Sci-Fi	PG	1	1
3	Robinson Crusoe on Mars	Byron Haskins	Sci-Fi	PG	1	1
4	Ghostbusters	Ivan Reitman	Comedy	PG	1	1
5	Krampus	Micheal Daugherty	Christmas	PG-13	1	1
6	Lawless	John Hillcoat	Drama	R	1	1
7	Monty Python and the Holy Grail	Terry Gilliam & Terry Jones	Comedy	PG	1	2
8	Pacific Rim	Guillermo del Toro	Sci-Fi	PG-13	1	2
9	Lion King	Roger Allers & Rob Minkoff	Animation	G	1	2
10	Mortdecai	David Koepp	Comedy	R	1	2
11	Red Tails	Anthony Hemingway	History	PG-13	1	2
12	Scott Pilgrim vs. the World	Edgar Wright	Comedy	PG-13	1	2
13	Tombstone	George P. Cosmatos & K...	Western	R	1	3
14	Gremlins	Joe Dante	Horror	PG	1	3
15	The Lego Movie	Christopher Miller & Phil ...	Animation	PG	1	3
16	The Monster Squad	Fred Dekker	Comedy	PG-13	1	3
17	Killer Klowns from Outer Space	Stephen Chiodo	Horror	PG-13	1	3
18	The Mask	Chuck Russell	Comedy	PG-13	1	3
19	The Suicide Squad	James Gunn	Action/Adventure	R	1	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Update if In Stock-

-Swagger-



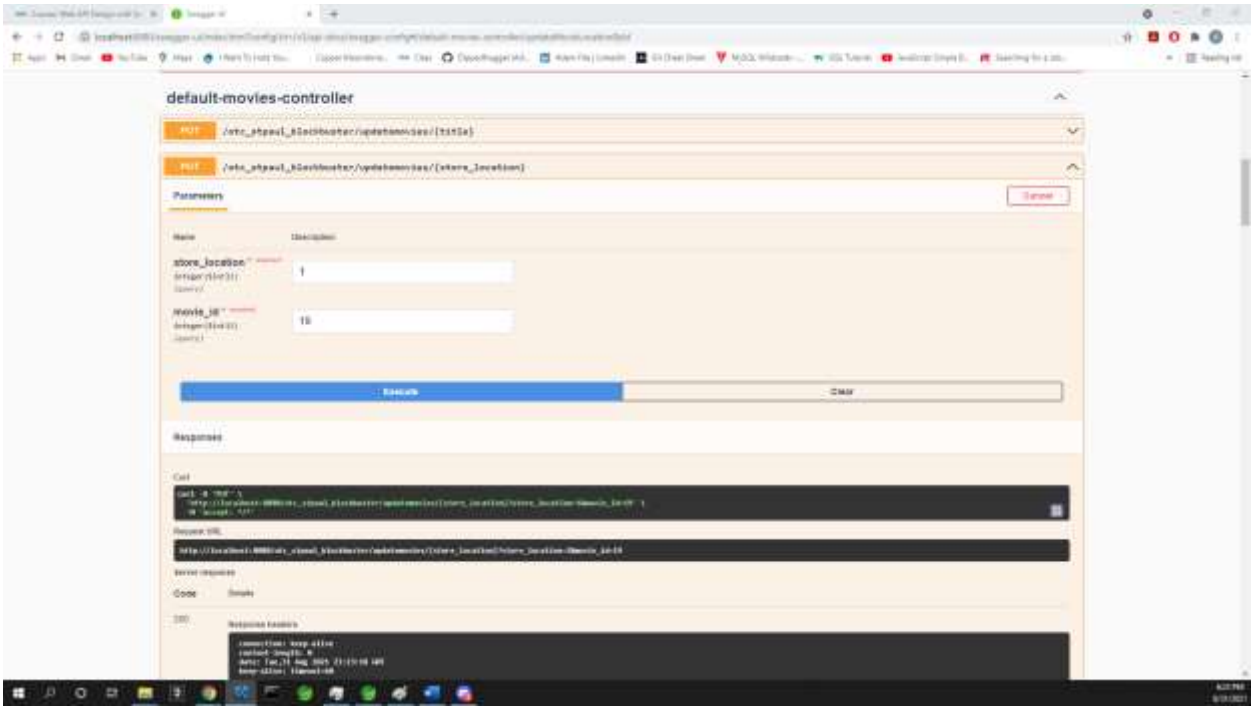
-Database-

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

movie_id	title	director	genre	rating	in_stock	store_location
1	Spider-Man 2	Sam Raimi	Comic Book	PG-13	1	1
2	Back to the Future 3	Robert Zemeckis	Sci-Fi	PG	1	1
3	Robinson Crusoe on Mars	Byron Haskins	Sci-Fi	PG	1	1
4	Ghostbusters	Ivan Reitman	Comedy	PG	1	1
5	Krampus	Micheal Daugherty	Christmas	PG-13	1	1
6	Lawless	John Hillcoat	Drama	R	1	1
7	Monty Python and the Holy Grail	Terry Gilliam & Terry Jones	Comedy	PG	1	2
8	Pacific Rim	Guillermo del Toro	Sci-Fi	PG-13	1	2
9	Lion King	Roger Allers & Rob Minkoff	Animation	G	1	2
10	Mortdecai	David Koepp	Comedy	R	1	2
11	Red Tails	Anthony Hemingway	History	PG-13	1	2
12	Scott Pilgrim vs. the World	Edgar Wright	Comedy	PG-13	1	2
13	Tombstone	George P. Cosmatos & K...	Western	R	1	3
14	Gremlins	Joe Dante	Horror	PG	1	3
15	The Lego Movie	Christopher Miller & Phil ...	Animation	PG	1	3
16	The Monster Squad	Fred Dekker	Comedy	PG-13	1	3
17	Killer Klowns from Outer Space	Stephen Chiodo	Horror	PG-13	1	3
18	The Mask	Chuck Russell	Comedy	PG-13	1	3
19	The Suicide Squad	James Gunn	Action/Adventure	R	0	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Update Store Location –

-Swagger-



-Database-

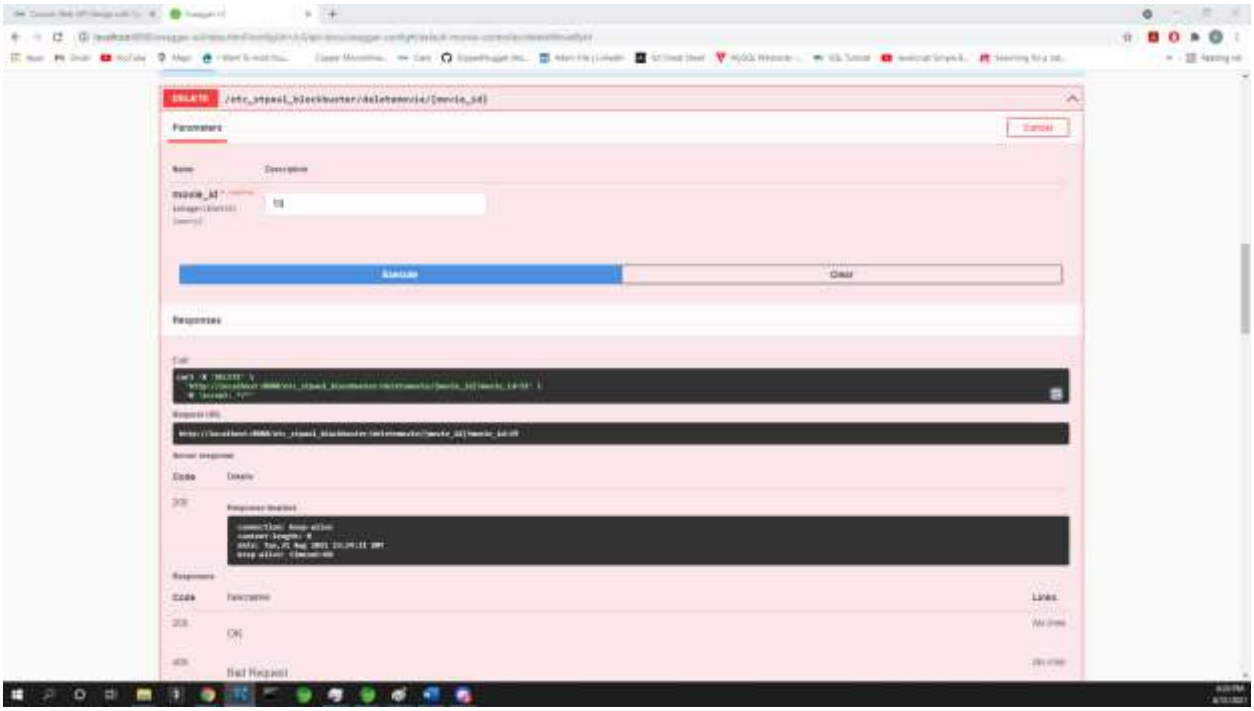
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

movie_id	title	director	genre	rating	in_stock	store_location
1	Spider-Man 2	Sam Raimi	ok	PG-13	1	1
2	Back to the Future 3	Robert Zemeckis	Sci-Fi	PG	1	1
3	Robinson Crusoe on Mars	Byron Haskins	Sci-Fi	PG	1	1
4	Ghostbusters	Ivan Reitman	Comedy	PG	1	1
5	Krampus	Micheal Daugherty	Christmas	PG-13	1	1
6	Lawless	John Hillcoat	Drama	R	1	1
7	Monty Python and the Holy Grail	Terry Gilliam & Terry Jones	Comedy	PG	1	2
8	Pacific Rim	Guillermo del Toro	Sci-Fi	PG-13	1	2
9	Lion King	Roger Allers & Rob Minkoff	Animation	G	1	2
10	Mortdecai	David Koepp	Comedy	R	1	2
11	Red Tails	Anthony Hemingway	History	PG-13	1	2
12	Scott Pilgrim vs. the World	Edgar Wright	Comedy	PG-13	1	2
13	Tombstone	George P. Cosmatos & K...	Western	R	1	3
14	Gremlins	Joe Dante	Horror	PG	1	3
15	The Lego Movie	Christopher Miller & Phil ...	Animation	PG	1	3
16	The Monster Squad	Fred Dekker	Comedy	PG-13	1	3
17	Killer Klowns from Outer Space	Stephen Chiodo	Horror	PG-13	1	3
18	The Mask	Chuck Russell	Comedy	PG-13	1	3
19	The Suicide Squad	James Gunn	Action/Adventure	R	0	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Edit current row

Delete a Movie –

-Swagger-



-Database-

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

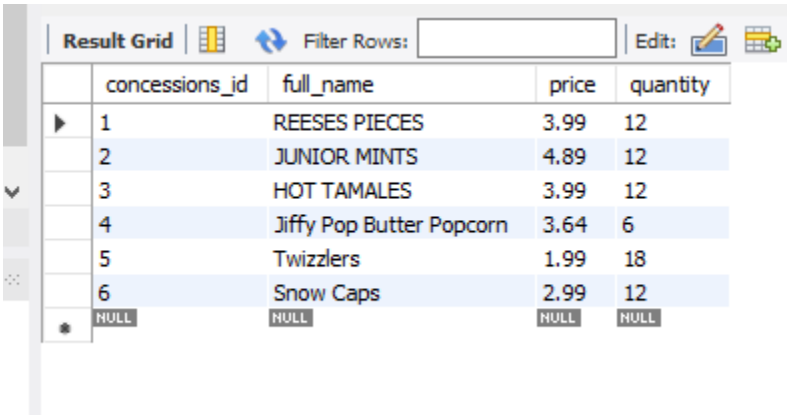
movie_id	title	director	genre	rating	in_stock	store_location
1	Spider-Man 2	Sam Raimi	Comic Book	PG-13	1	1
2	Back to the Future 3	Robert Zemeckis	Sci-Fi	PG	1	1
3	Robinson Crusoe on Mars	Byron Haskins	Sci-Fi	PG	1	1
4	Ghostbusters	Ivan Reitman	Comedy	PG	1	1
5	Krampus	Micheal Daugherty	Christmas	PG-13	1	1
6	Lawless	John Hillcoat	Drama	R	1	1
7	Monty Python and the Holy Grail	Terry Gilliam & Terry Jones	Comedy	PG	1	2
8	Pacific Rim	Guillermo del Toro	Sci-Fi	PG-13	1	2
9	Lion King	Roger Allers & Rob Minkoff	Animation	G	1	2
10	Mortdecai	David Koepp	Comedy	R	1	2
11	Red Tails	Anthony Hemingway	History	PG-13	1	2
12	Scott Pilgrim vs. the World	Edgar Wright	Comedy	PG-13	1	2
13	Tombstone	George P. Cosmatos & K...	Western	R	1	3
14	Gremlins	Joe Dante	Horror	PG	1	3
15	The Lego Movie	Christopher Miller & Phil ...	Animation	PG	1	3
16	The Monster Squad	Fred Dekker	Comedy	PG-13	1	3
17	Killer Klowns from Outer Space	Stephen Chiodo	Horror	PG-13	1	3
18	The Mask	Chuck Russell	Comedy	PG-13	1	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Concessions:

Variables: Gum Drops 1.99 12

Update Variables: Dots .99 18

Starting Database –

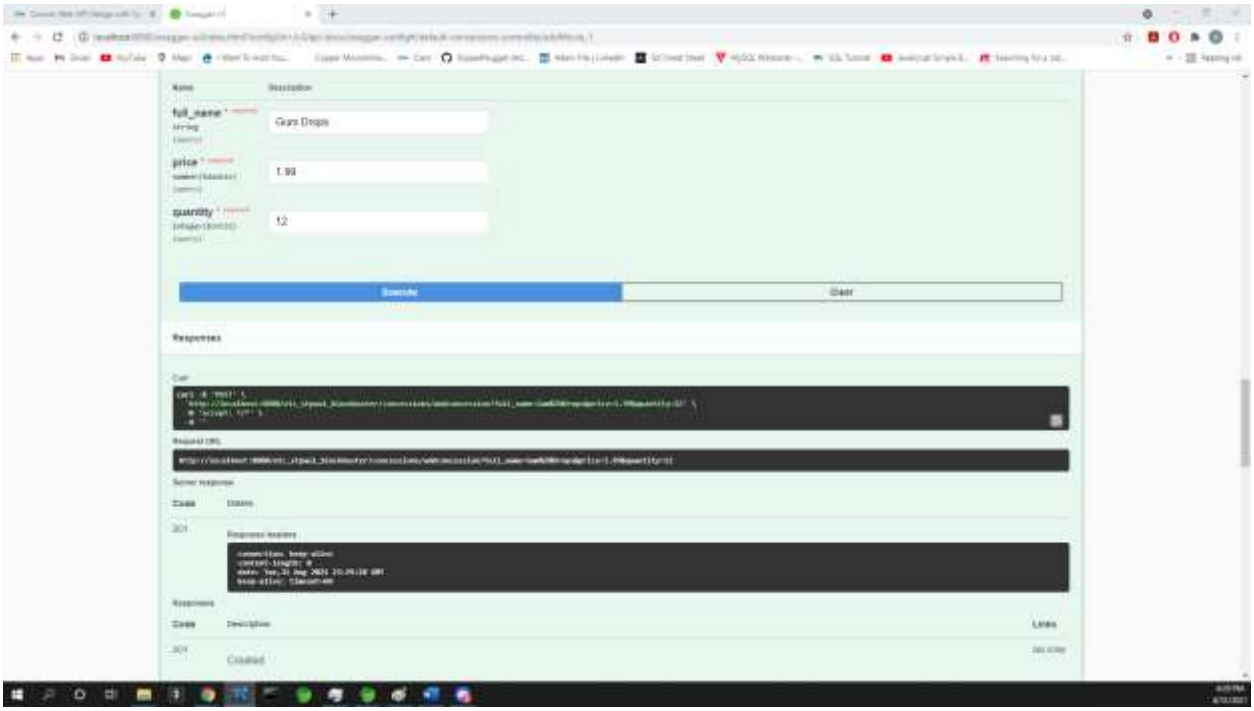


Result Grid | Filter Rows: | Edit:

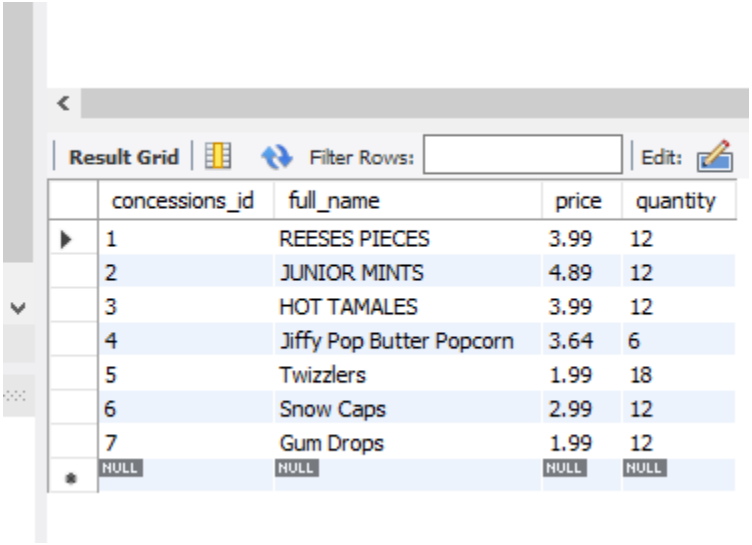
concessions_id	full_name	price	quantity
1	REESES PIECES	3.99	12
2	JUNIOR MINTS	4.89	12
3	HOT TAMALES	3.99	12
4	Jiffy Pop Butter Popcorn	3.64	6
5	Twizzlers	1.99	18
6	Snow Caps	2.99	12
NULL	NULL	NULL	NULL

Create a Concession –

-Swagger-



-Database-

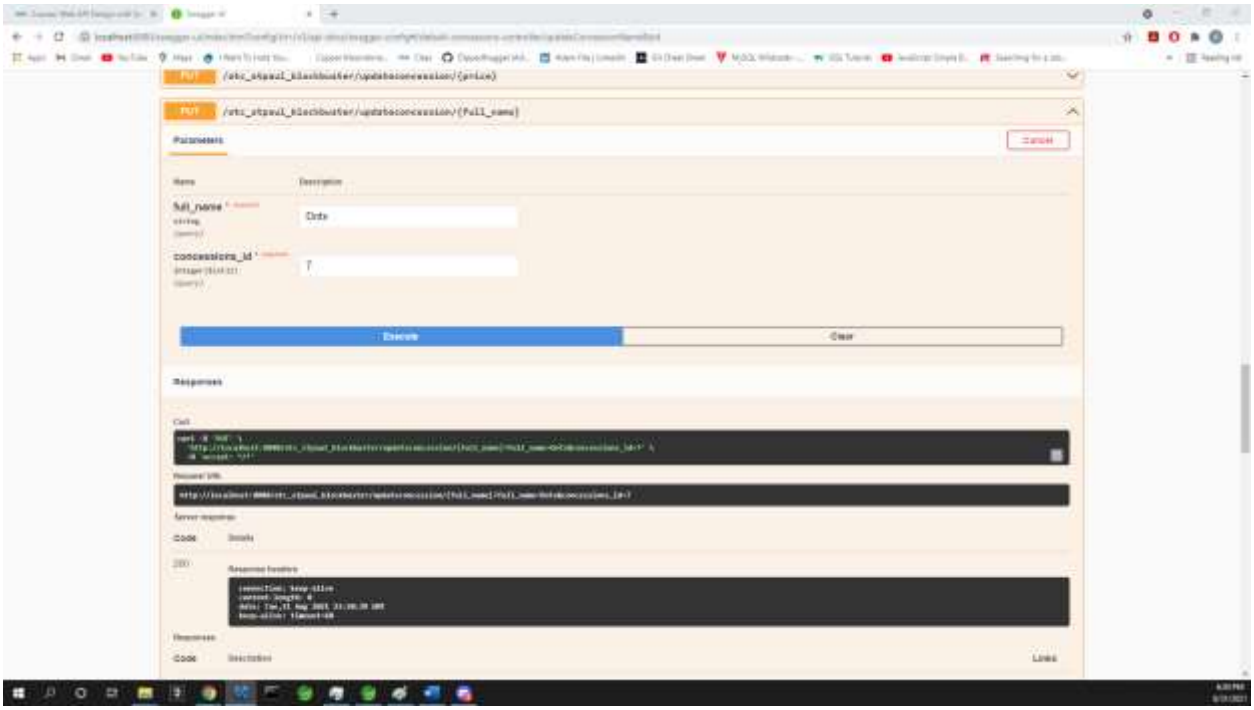


Result Grid | Filter Rows: | Edit:

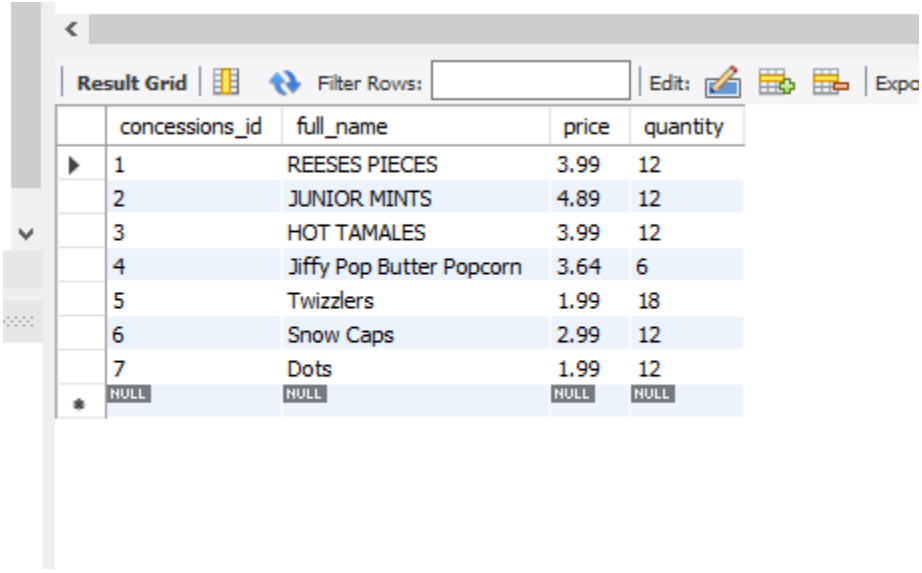
concessions_id	full_name	price	quantity
1	REESES PIECES	3.99	12
2	JUNIOR MINTS	4.89	12
3	HOT TAMALES	3.99	12
4	Jiffy Pop Butter Popcorn	3.64	6
5	Twizzlers	1.99	18
6	Snow Caps	2.99	12
7	Gum Drops	1.99	12
NULL	NULL	NULL	NULL

Update Full Name –

-Swagger-



-Database-

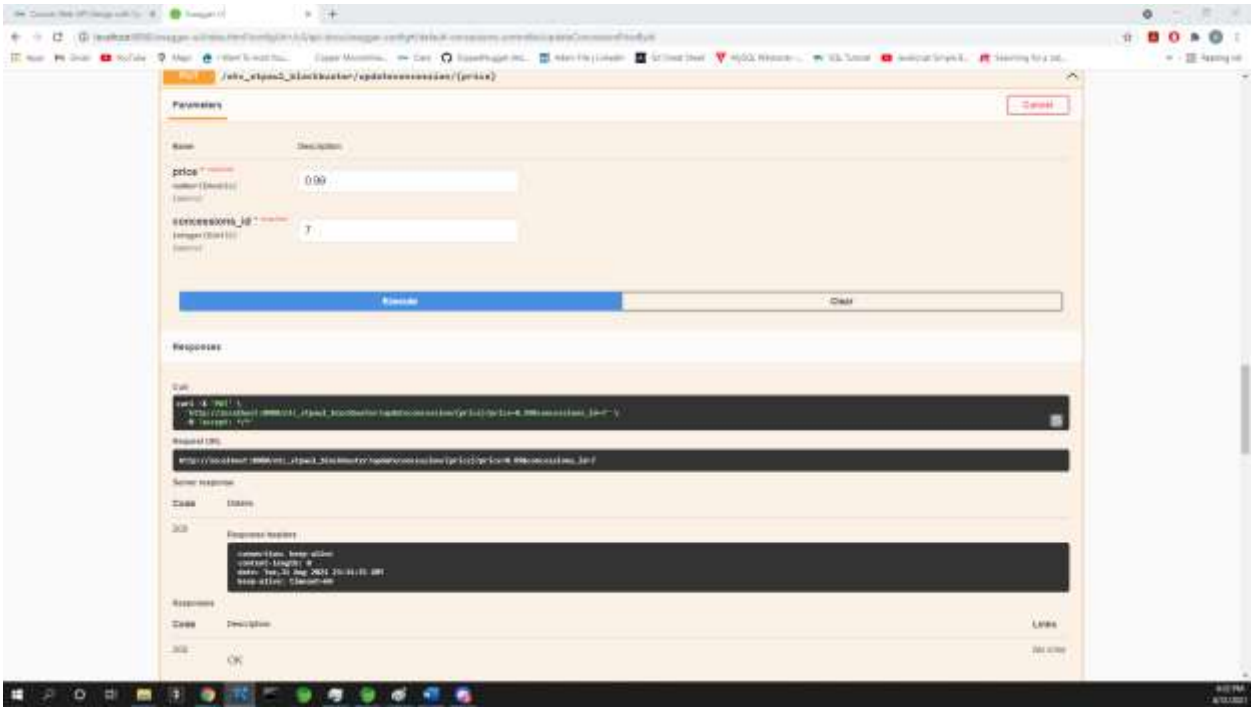


Result Grid | Filter Rows: | Edit: | Expo

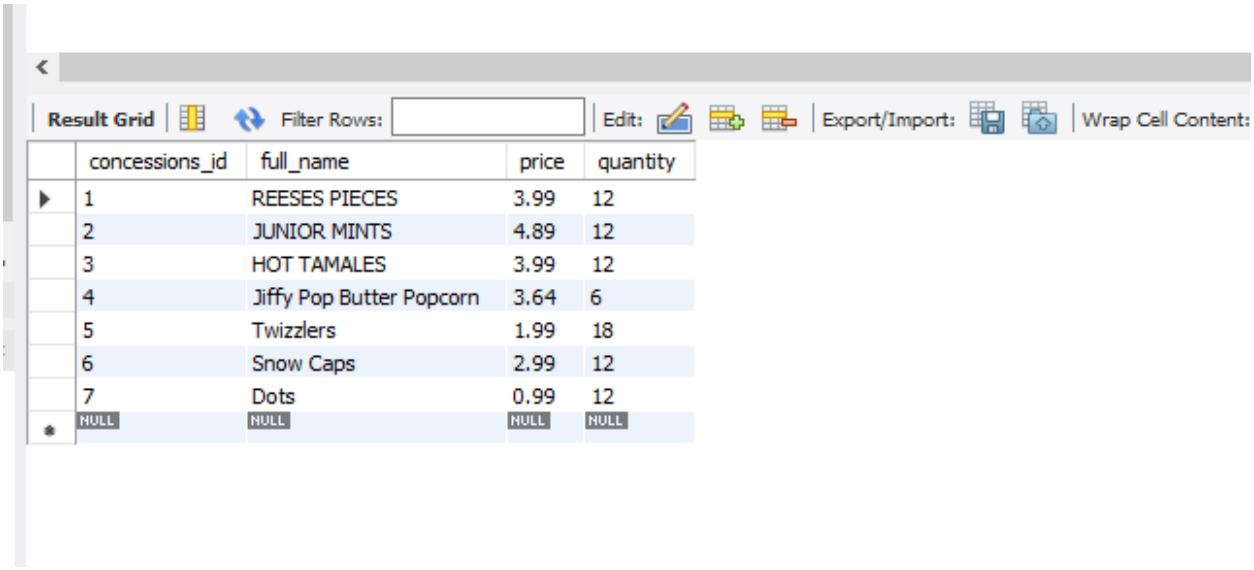
concessions_id	full_name	price	quantity
1	REESES PIECES	3.99	12
2	JUNIOR MINTS	4.89	12
3	HOT TAMALES	3.99	12
4	Jiffy Pop Butter Popcorn	3.64	6
5	Twizzlers	1.99	18
6	Snow Caps	2.99	12
7	Dots	1.99	12
NULL	NULL	NULL	NULL

Update Price –

-Swagger-



-Database-

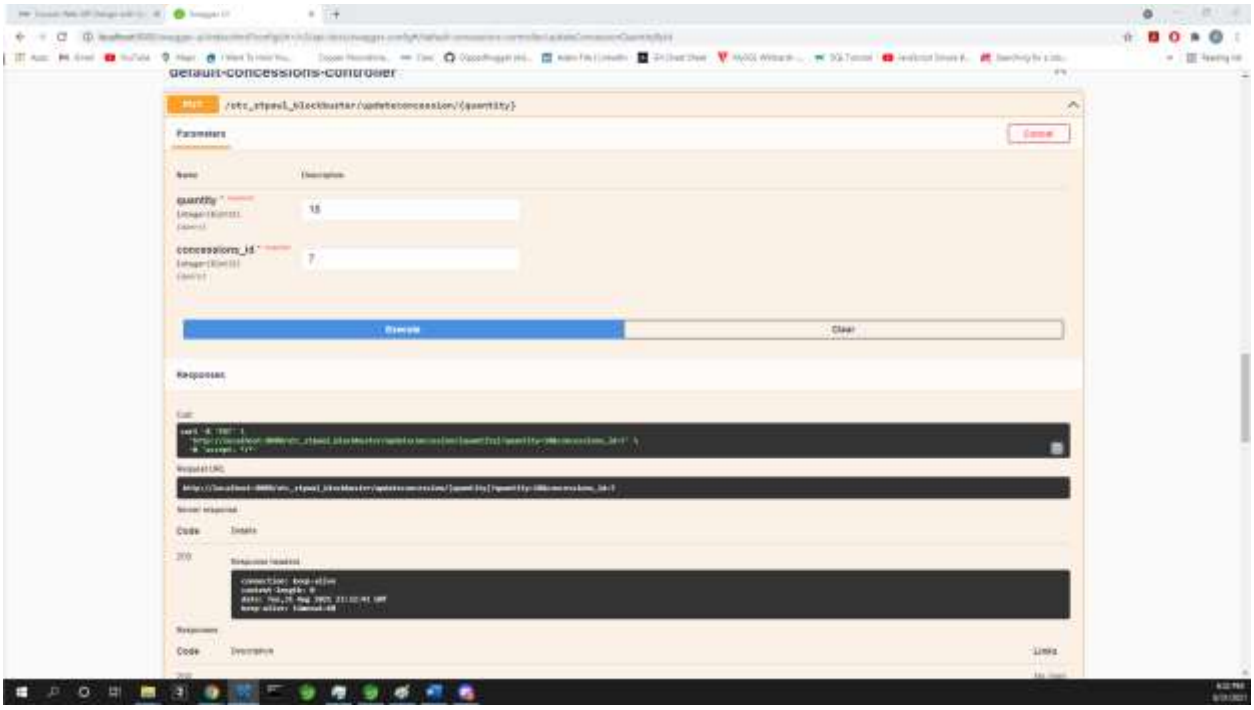


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

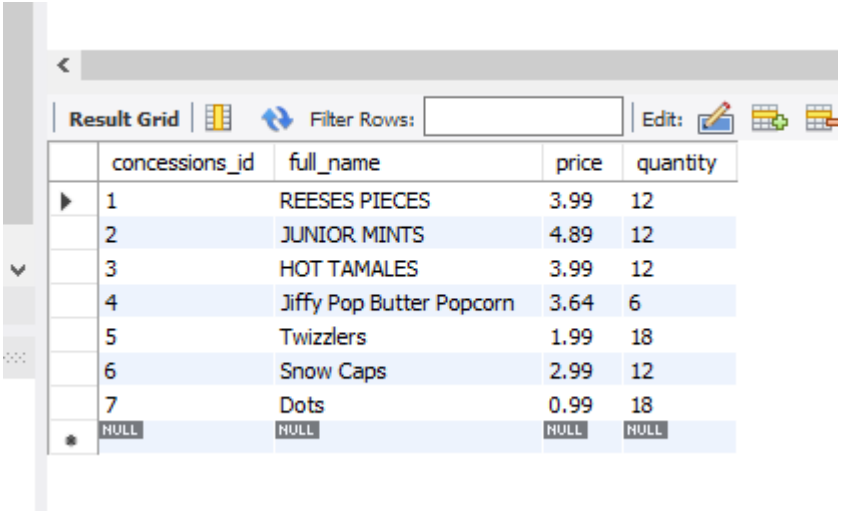
concessions_id	full_name	price	quantity
1	REESES PIECES	3.99	12
2	JUNIOR MINTS	4.89	12
3	HOT TAMALES	3.99	12
4	Jiffy Pop Butter Popcorn	3.64	6
5	Twizzlers	1.99	18
6	Snow Caps	2.99	12
7	Dots	0.99	12
NULL	NULL	NULL	NULL

Update Quantity –

-Swagger-



-Database-

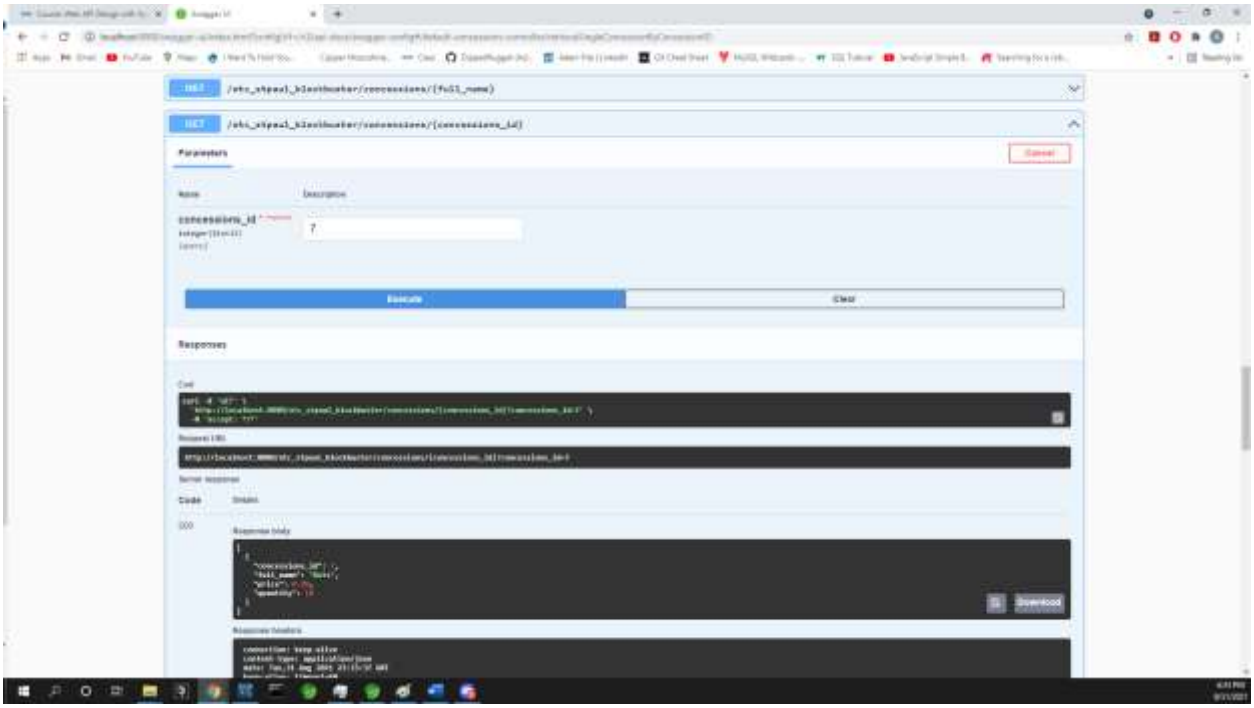


Result Grid | Filter Rows: | Edit:

concessions_id	full_name	price	quantity
1	REESES PIECES	3.99	12
2	JUNIOR MINTS	4.89	12
3	HOT TAMALES	3.99	12
4	Jiffy Pop Butter Popcorn	3.64	6
5	Twizzlers	1.99	18
6	Snow Caps	2.99	12
7	Dots	0.99	18
NULL	NULL	NULL	NULL

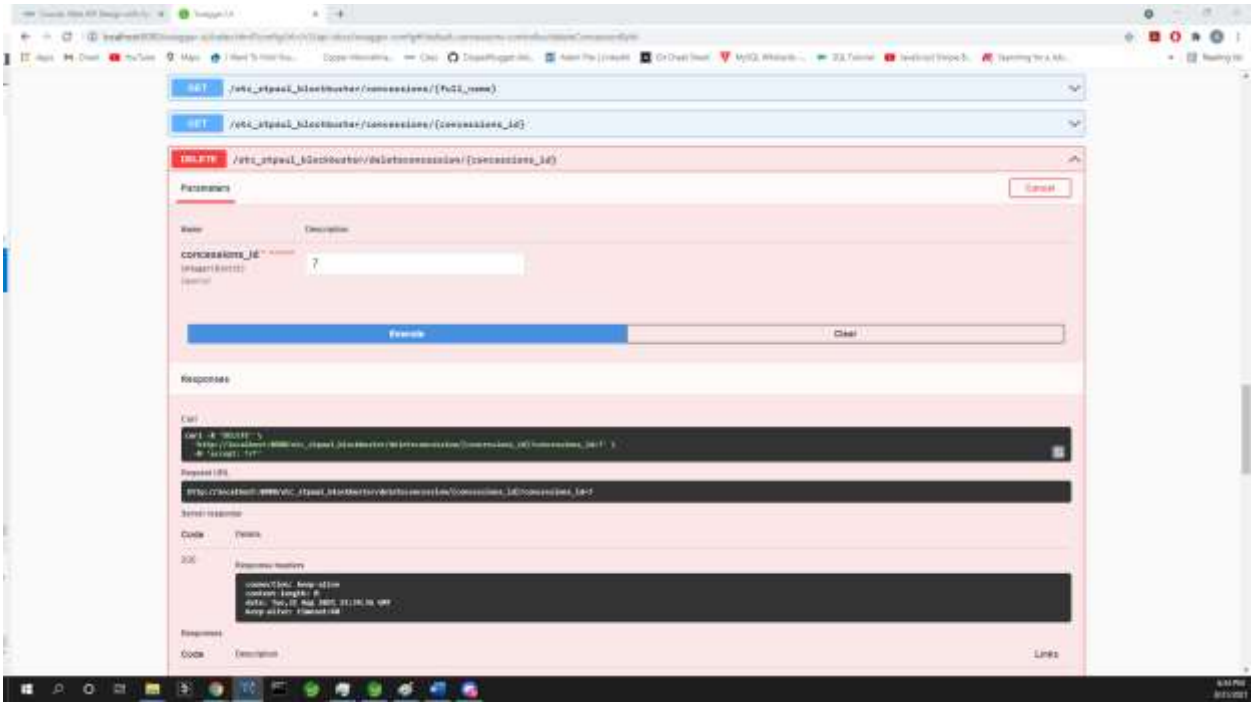
Get Concession-

-Swagger-

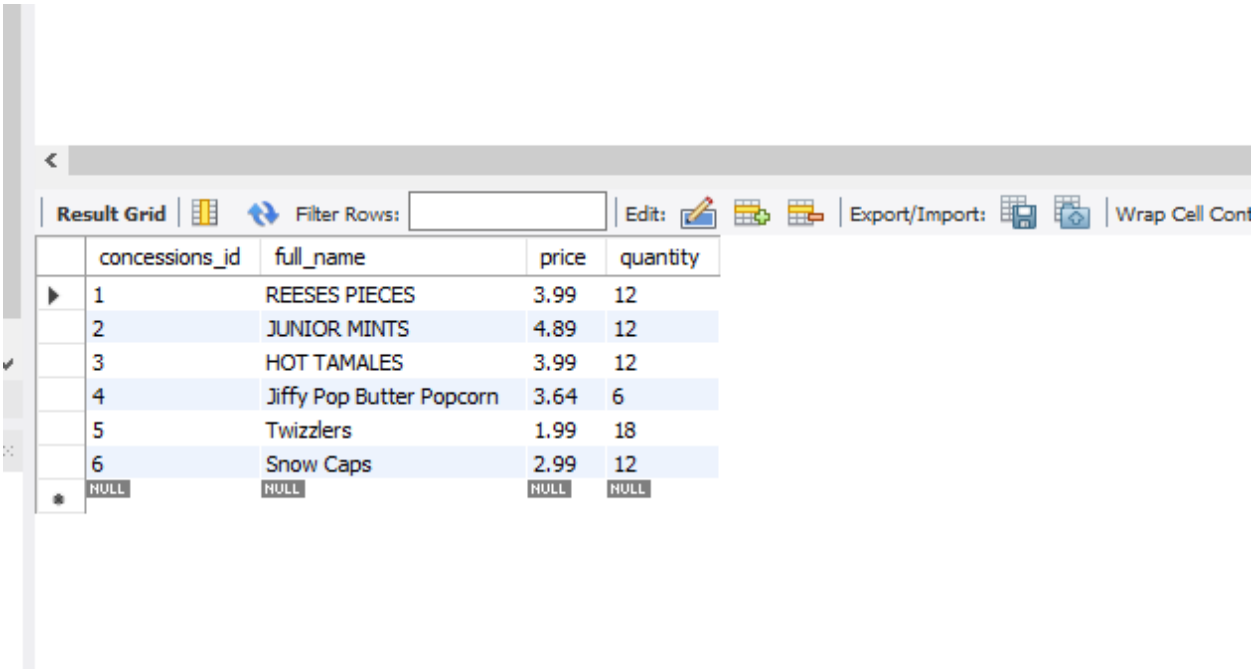


Delete Concession –

-Swagger-



-Database-



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Cont

concessions_id	full_name	price	quantity
1	REESES PIECES	3.99	12
2	JUNIOR MINTS	4.89	12
3	HOT TAMALES	3.99	12
4	Jiffy Pop Butter Popcorn	3.64	6
5	Twizzlers	1.99	18
6	Snow Caps	2.99	12
NULL	NULL	NULL	NULL

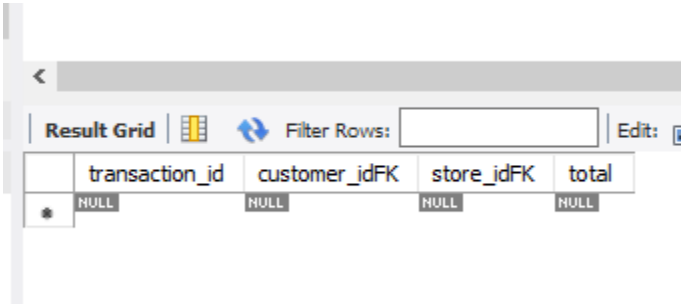
Transaction:

The Running of the transaction code Is done via a TEST.

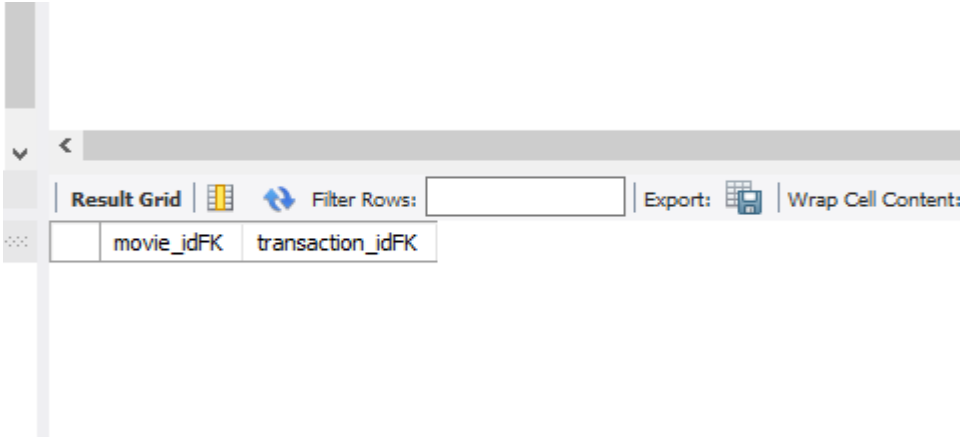
That is correct we actually managed to get a TEST running.

Starting Database –

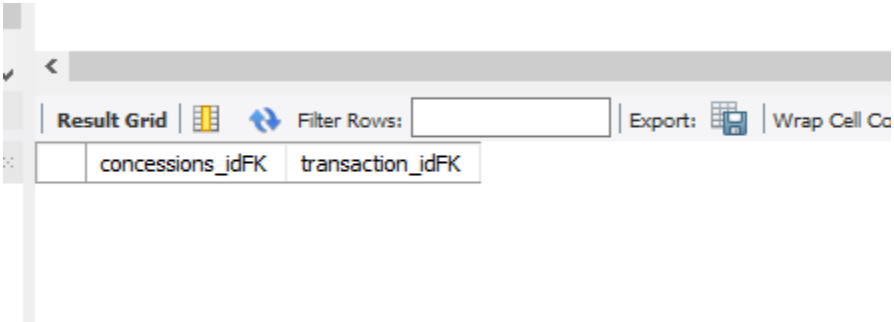
-Transactions-



-Stored Transaction Movie List-



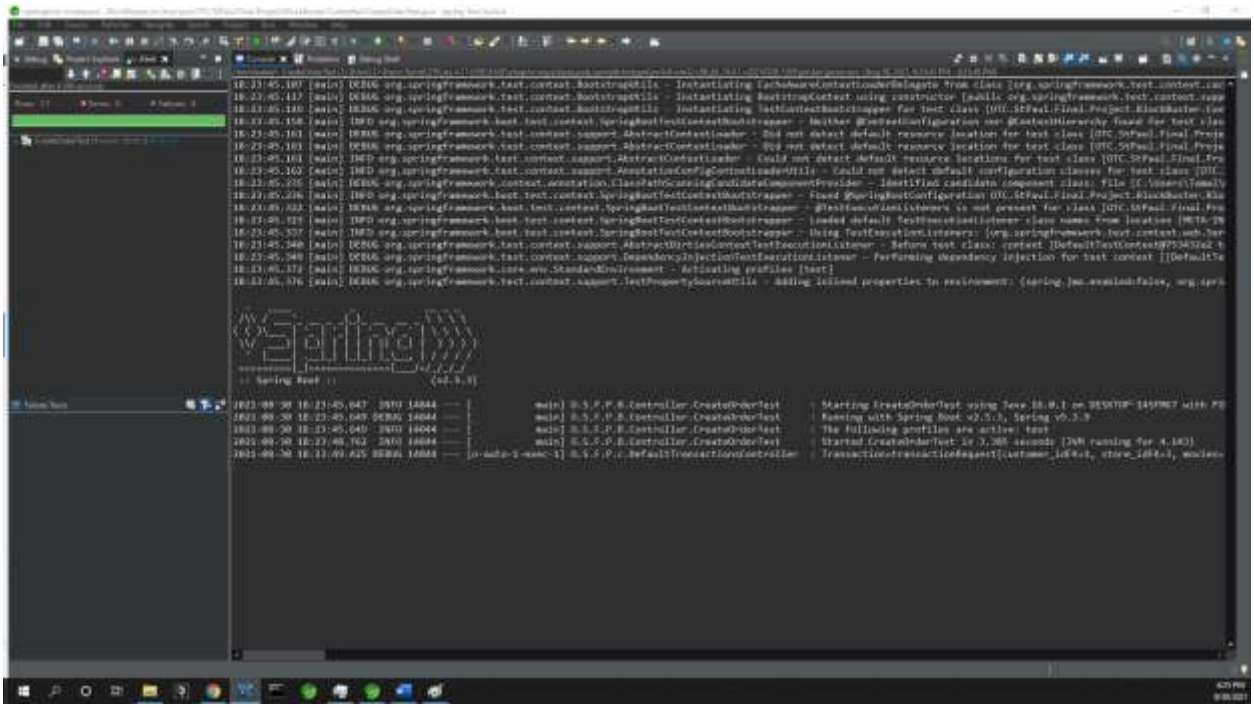
-Stored Transaction Concession List-



Test Name-

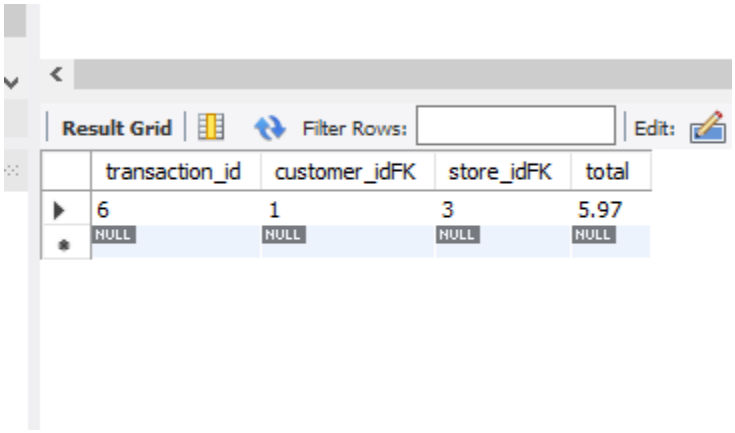
```
@Test
void testCreateOrderReturnsSuccess201() {
```

Green Bar –



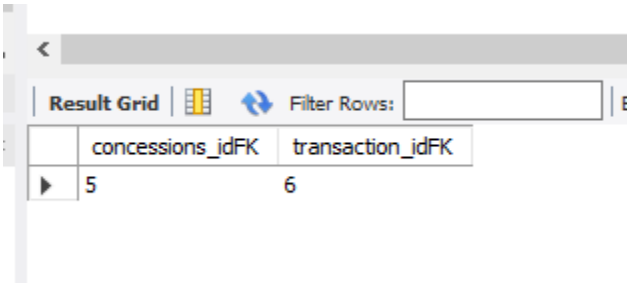
Created Database –

-Transactions-



-Stored Transaction Movie List-

-Stored Transaction Concession List-



THE END!

“Lets all go to the lobby, lets all go to the lobby, lets all go to the lobby, AND grab ourselves a SNACK!”