```
package com.example.springboot.controller;

import static org.mockito.Mockito.any;
import static org.mockito.Mockito.anyLong;
import static org.mockito.Mockito.when;

import com.example.springboot.dto.UserRequestDTO;
import com.example.springboot.model.Employee;
import com.example.springboot.service.EmployeeService;
import com.fasterxml.jackson.databind.ObjectMapper;

import java.util.ArrayList;

import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.http.MediaType;
import org.springframework.test.context.ContextConfiguration;
import
org.springframework.test.context.junit.jupiter.SpringExtension;
import
org.springframework.test.web.servlet.request.MockHttpServletReque
stBuilder;
import
org.springframework.test.web.servlet.request.MockMvcRequestBuilde
rs;
import
org.springframework.test.web.servlet.result.MockMvcResultMatchers
;
import
org.springframework.test.web.servlet.setup.MockMvcBuilders;

@ContextConfiguration(classes = {EmployeeController.class,
        UserRequestDTO.class})
@ExtendWith(SpringExtension.class)
class EmployeeControllerTest {
    @Autowired
    private UserRequestDTO userRequestDTO;

    @Autowired
    private EmployeeController employeeController;

    @MockBean
    private EmployeeService employeeService;
```

```
    /**
     * Method under test: {@link
EmployeeController#getEmployeeList()}
     */
    @Test
    void testGetEmployeeList() throws Exception {
        when(employeeService.getEmployeeList()).thenReturn(new
ArrayList<>());
        MockHttpServletRequestBuilder requestBuilder =
MockMvcRequestBuilders.get("/api/employee");
        MockMvcBuilders.standaloneSetup(employeeController)
                .build()
                .perform(requestBuilder)
                .andExpect(MockMvcResultMatchers.status().isOk())

.andExpect(MockMvcResultMatchers.content().contentType("applicati
on/json"))

.andExpect(MockMvcResultMatchers.content().string("[]"));
    }

    /**
     * Method under test: {@link
EmployeeController#getEmployeeList()}
     */
    @Test
    void testGetEmployeeList2() throws Exception {
        when(employeeService.getEmployeeList()).thenReturn(new
ArrayList<>());
        MockHttpServletRequestBuilder getResult =
MockMvcRequestBuilders.get("/api/employee");
        getResult.characterEncoding("Encoding");
        MockMvcBuilders.standaloneSetup(employeeController)
                .build()
                .perform(getResult)
                .andExpect(MockMvcResultMatchers.status().isOk())

.andExpect(MockMvcResultMatchers.content().contentType("applicati
on/json"))

.andExpect(MockMvcResultMatchers.content().string("[]"));
    }

    /**
```

```
     * Method under test: {@link
EmployeeController#getEmployeeById(long)}
     */
    @Test
    void testGetEmployeeById() throws Exception {
        Employee employee = new Employee();
        employee.setEmail("jane.doe@example.org");
        employee.setFirstName("Jane");
        employee.setId(123L);
        employee.setLastName("Doe");

when(employeeService.getEmployeeById(anyLong())).thenReturn(emplo
yee);
        MockHttpServletRequestBuilder requestBuilder =
MockMvcRequestBuilders.get("/api/employee/{id}", 123L);
        MockMvcBuilders.standaloneSetup(employeeController)
                .build()
                .perform(requestBuilder)
                .andExpect(MockMvcResultMatchers.status().isOk())

.andExpect(MockMvcResultMatchers.content().contentType("applicati
on/json"))
                .andExpect(MockMvcResultMatchers.content()
                        .string("
{\"id\":123,\"firstName\":\"Jane\",\"lastName\":\"Doe\",\"email\"
:\"jane.doe@example.org\"}"));
    }

    /**
     * Method under test: {@link
EmployeeController#updateEmployee(long, UserRequestDTO)}
     */
    @Test
    void testUpdateEmployee() throws Exception {
        Employee employee = new Employee();
        employee.setEmail("jane.doe@example.org");
        employee.setFirstName("Jane");
        employee.setId(123L);
        employee.setLastName("Doe");
        when(employeeService.updateEmployee((UserRequestDTO)
any(), anyLong())).thenReturn(employee);

        UserRequestDTO userRequestDTO1 = new UserRequestDTO();
        userRequestDTO1.setEmail("jane.doe@example.org");
        userRequestDTO1.setFirstName("Jane");
```

```
        userRequestDTO1.setLastName("Doe");
        String content = (new
ObjectMapper()).writeValueAsString(userRequestDTO1);
        MockHttpServletRequestBuilder requestBuilder =
MockMvcRequestBuilders.put("/api/employee/{id}", 123L)
                .contentType(MediaType.APPLICATION_JSON)
                .content(content);
        MockMvcBuilders.standaloneSetup(employeeController)
                .build()
                .perform(requestBuilder)
                .andExpect(MockMvcResultMatchers.status().isOk())

.andExpect(MockMvcResultMatchers.content().contentType("applicati
on/json"))
                .andExpect(MockMvcResultMatchers.content()
                        .string("
{\"id\":123,\"firstName\":\"Jane\",\"lastName\":\"Doe\",\"email\"
:\"jane.doe@example.org\"}"));
    }

    /**
     * Method under test: {@link
EmployeeController#deleteEmployee(long)}
     */
    @Test
    void testDeleteEmployee() throws Exception {
        Employee employee = new Employee();
        employee.setEmail("jane.doe@example.org");
        employee.setFirstName("Jane");
        employee.setId(123L);
        employee.setLastName("Doe");

when(employeeService.deleteEmployee(anyLong())).thenReturn(employ
ee);
        MockHttpServletRequestBuilder requestBuilder =
MockMvcRequestBuilders.delete("/api/employee/{id}", 123L);
        MockMvcBuilders.standaloneSetup(employeeController)
                .build()
                .perform(requestBuilder)
                .andExpect(MockMvcResultMatchers.status().isOk())

.andExpect(MockMvcResultMatchers.content().contentType("text/plai
n;charset=ISO-8859-1"))

.andExpect(MockMvcResultMatchers.content().string("Employee
```

```
deleted successfully"));
    }

    /**
     * Method under test: {@link
EmployeeController#saveEmployee(UserRequestDTO)}
     */
    @Test
    void testSaveEmployee() throws Exception {
        when(employeeService.getEmployeeList()).thenReturn(new
ArrayList<>());

        UserRequestDTO userRequestDTO1 = new UserRequestDTO();
        userRequestDTO1.setEmail("jane.doe@example.org");
        userRequestDTO1.setFirstName("Jane");
        userRequestDTO1.setLastName("Doe");
        String content = (new
ObjectMapper()).writeValueAsString(userRequestDTO1);
        MockHttpServletRequestBuilder requestBuilder =
MockMvcRequestBuilders.get("/api/employee")
                .contentType(MediaType.APPLICATION_JSON)
                .content(content);
        MockMvcBuilders.standaloneSetup(employeeController)
                .build()
                .perform(requestBuilder)
                .andExpect(MockMvcResultMatchers.status().isOk())

.andExpect(MockMvcResultMatchers.content().contentType("applicati
on/json"))

.andExpect(MockMvcResultMatchers.content().string("[]"));
    }
}
```