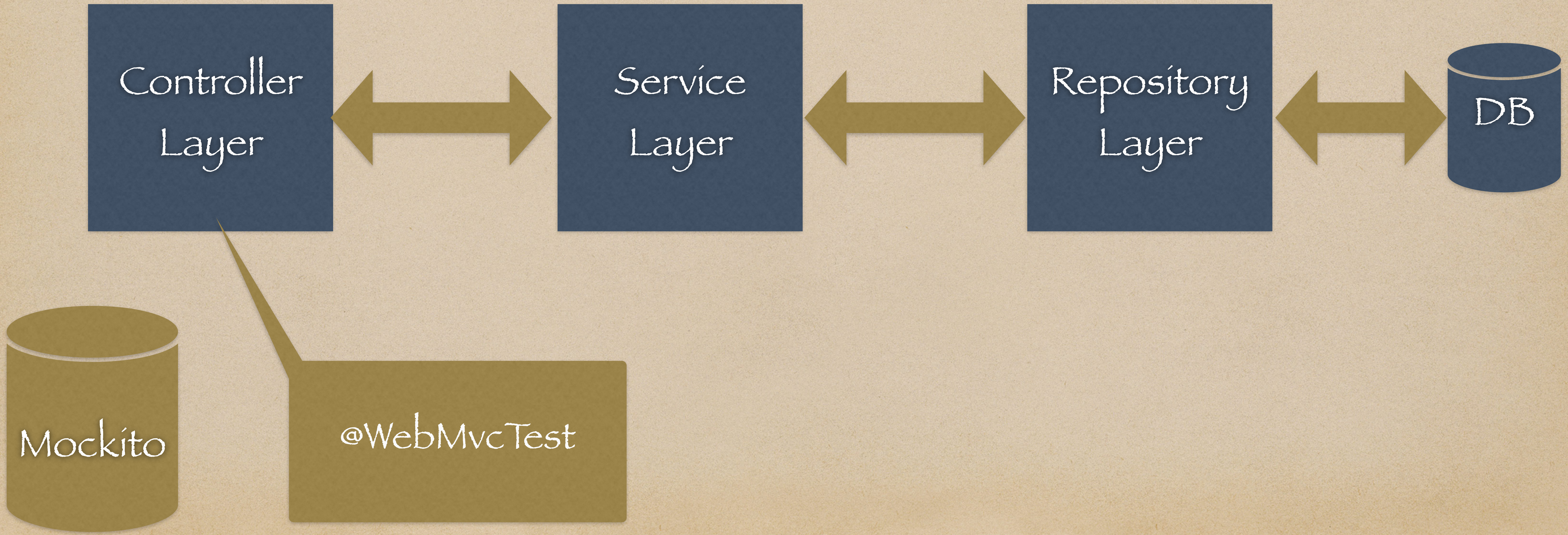
Spring Boot Application Controller Layer Testing

By Ramesh Fadatare (Java Guides)

Spring Boot Application Controller Layer Testing



Controller Layer Testing

1. **@MockMvcTest Annotation**
2. **Build createEmployee REST API**
3. **Unit test createEmployee REST API**
4. **Build GetAllEmployees REST API**
5. **Unit test GetAllEmployees REST API**
6. **Refactoring JUnit test to use static imports**
7. **Build getEmployeeById REST API**
8. **Unit test getEmployeeById REST API - Positive Scenario**
9. **Unit test getEmployeeById REST API - Negative Scenario**
10. **Build updateEmployee REST API**
11. **Unit test updateEmployee REST API - Positive Scenario**
12. **Unit test updateEmployee REST API - Negative Scenario**
13. **Build deleteEmployee REST API**
14. **Unit test deleteEmployee REST API**

Hamcrest Library

Hamcrest is the well-known framework used for unit testing in the Java ecosystem. It's bundled in JUnit and simply put, it uses existing predicates – called matcher classes – for making assertions.

Hamcrest is commonly used with *JUnit* and other testing frameworks for making assertions. Specifically, instead of using *JUnit's* numerous *assert* methods, we only use the API's single *assertThat* statement with appropriate matchers.

Hamcrest is() method: If we want to verify that the expected value (or object) is equal to the actual value (or object), we have to create our Hamcrest matcher by invoking the *is()* method of the Matchers class.
Syntax:
assertThat(ACTUAL, is(EXPECTED));

JsonPath Library

A Java DSL for reading JSON documents.

JsonPath expressions always refer to a JSON structure in the same way as XPath expression are used in combination with an XML document. The "root member object" in JsonPath is always referred to as $ regardless if it is an object or array.

JSON

```
{
   "firstName": "Ramesh",
   “lastName": “Fadatare",
    “email": "ramesh@gmail.com"
}
```

JsonPath Expressions

$ - root member of a JSON structure whether it is an object or array.

$.firstName = “Ramesh”

$.lastName = “Fadatare”

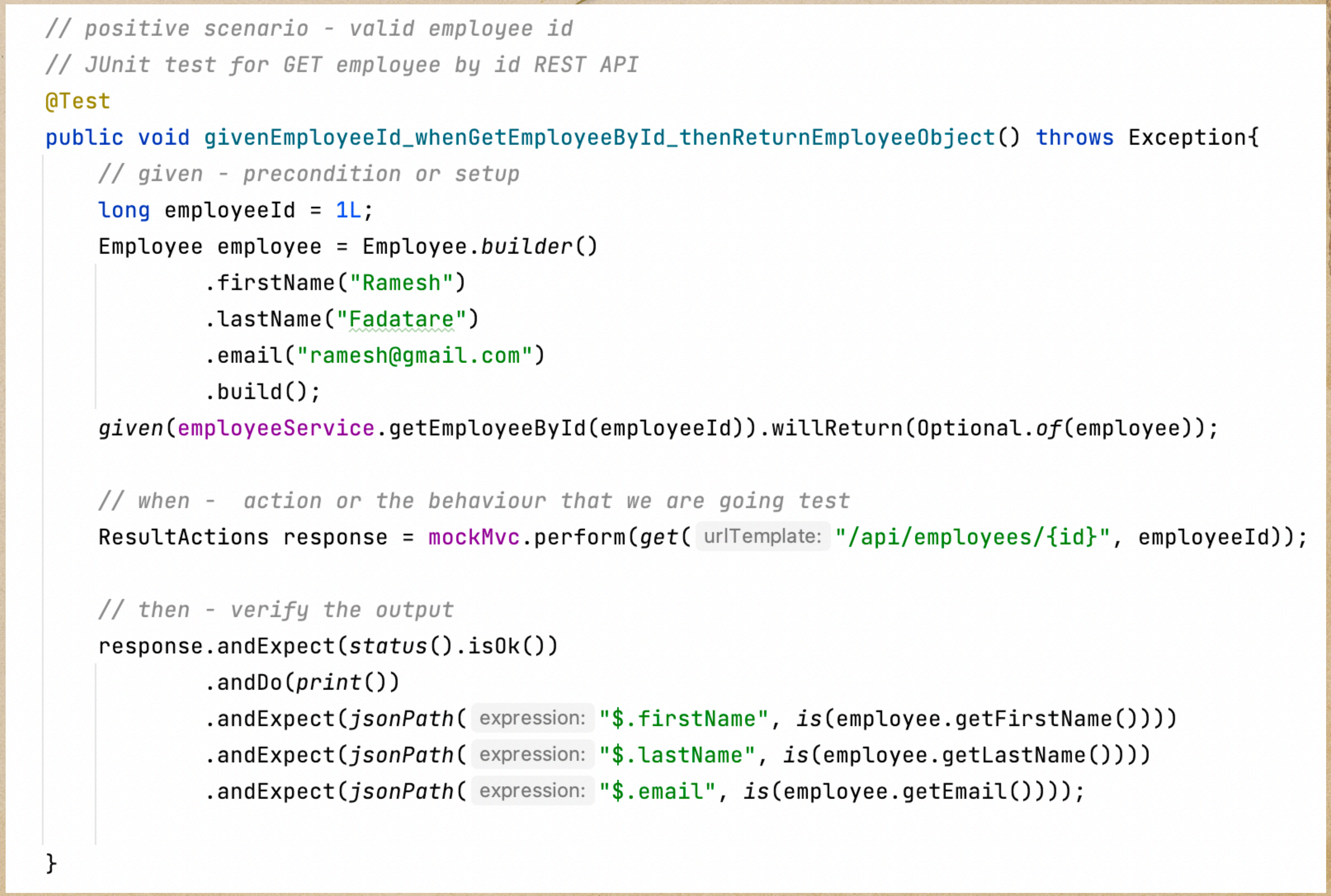
$.email = “ramesg@gmail.com”

JUnit tests in BDD Style

Syntax

```
@Test
public void given_when_then() {
     // given - precondition or setup

     // when - action or the behaviour we're testing

     // then - verify the output
}
```

Example

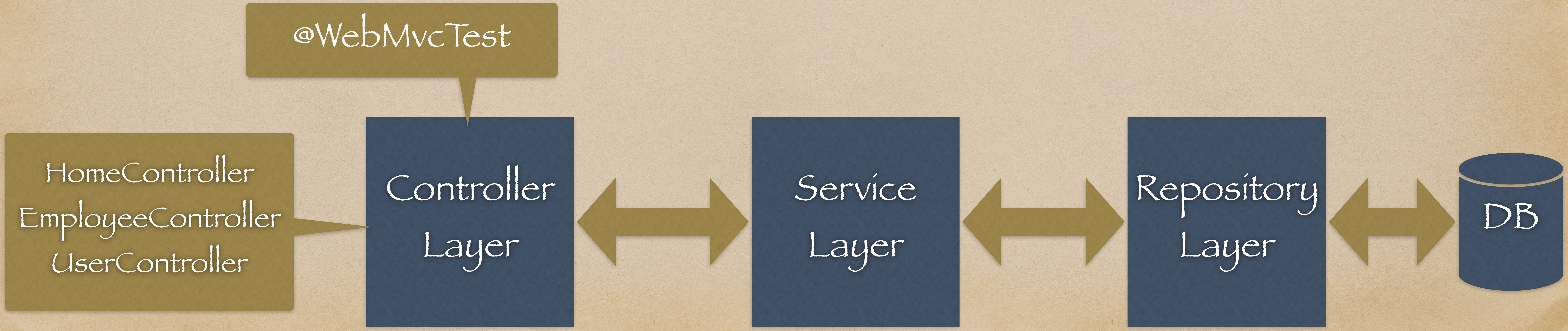


```
// positive scenario - valid employee id
// JUnit test for GET employee by id REST API
@Test
public void givenEmployeeId_whenGetEmployeeById_thenReturnEmployeeObject() throws Exception{
    // given - precondition or setup
    long employeeId = 1L;
    Employee employee = Employee.builder()
            .firstName("Ramesh")
            .lastName("Fadatare")
            .email("ramesh@gmail.com")
            .build();
    given(employeeService.getEmployeeById(employeeId)).willReturn(Optional.of(employee));

    // when -  action or the behaviour that we are going test
    ResultActions response = mockMvc.perform(get( urlTemplate: "/api/employees/{id}", employeeId));

    // then - verify the output
    response.andExpect(status().isOk())
            .andDo(print())
            .andExpect(jsonPath( expression: "$.firstName", is(employee.getFirstName())))
            .andExpect(jsonPath( expression: "$.lastName", is(employee.getLastName())))
            .andExpect(jsonPath( expression: "$.email", is(employee.getEmail())));

}
```

@WebMvcTest Annotation

SpringBoot provides **@WebMvcTest** annotation to test Spring MVC Controllers.
Also, **@WebMvcTest** based tests runs faster as it will load only the specified controller and its dependencies only without loading the entire application.

Spring Boot instantiates only the web layer rather than the whole application context. In an application with multiple controllers, you can even ask for only one to be instantiated by using, for example, **@WebMvcTest(HomeController.class).**



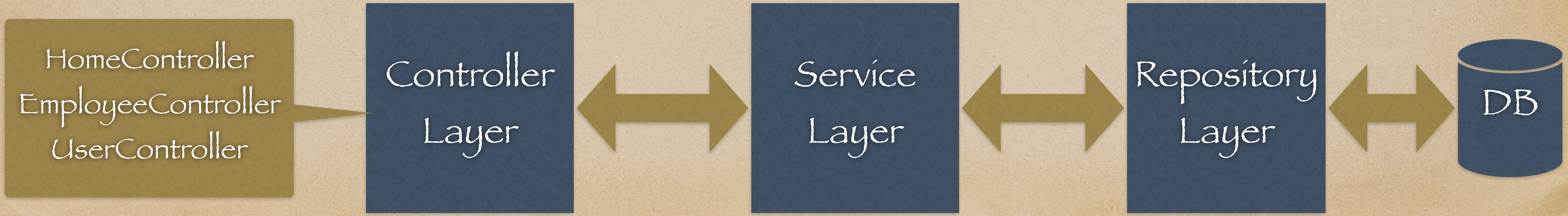
@WebMvcTest vs @SpringBootTest

Spring Boot provides @WebMvcTest annotation to test Spring MVC controllers. This annotation creates an application context that contains all the beans necessary for testing a Spring web controller.

Spring Boot provides @SpringBootTest annotation for Integration testing. This annotation creates an application context and loads full application context.

Unit testing - @WebMvcTest annotation

Integration testing - @SpringBootTest


HomeController
EmployeeController
UserController
Controller Layer
Service Layer
Repository Layer
DB