測試驅動程式開發實作

徐勊群

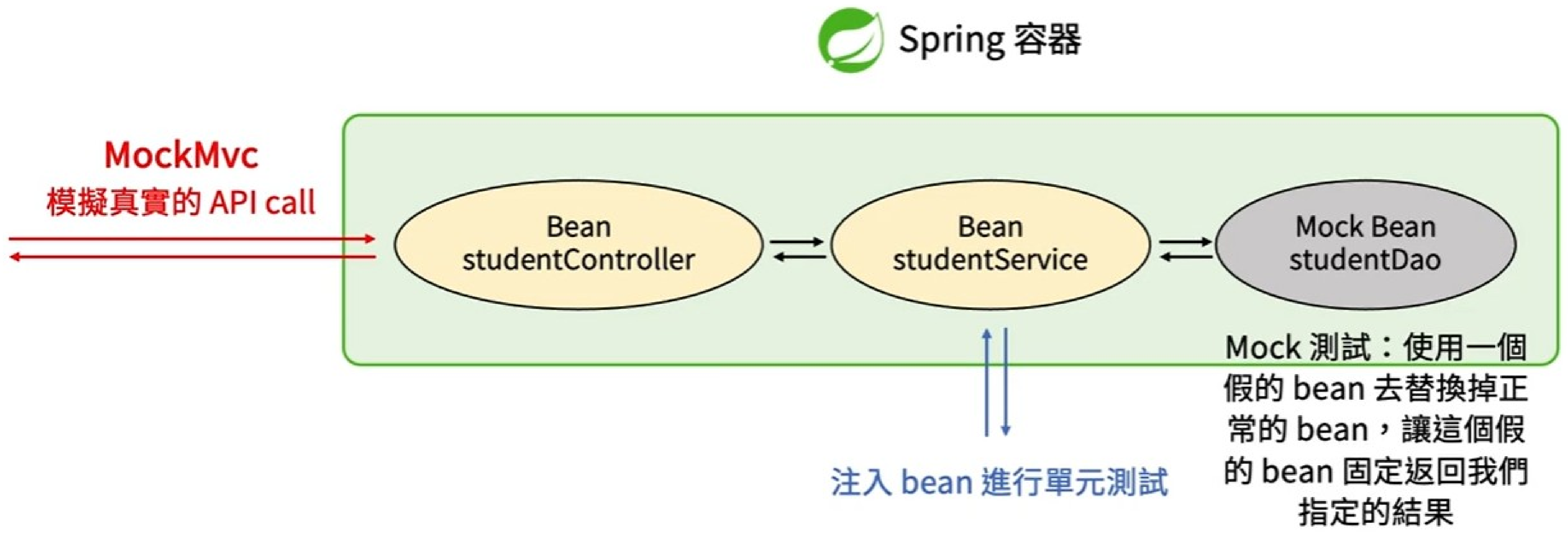
Jack Hsu

jack23h67@hotmail.com

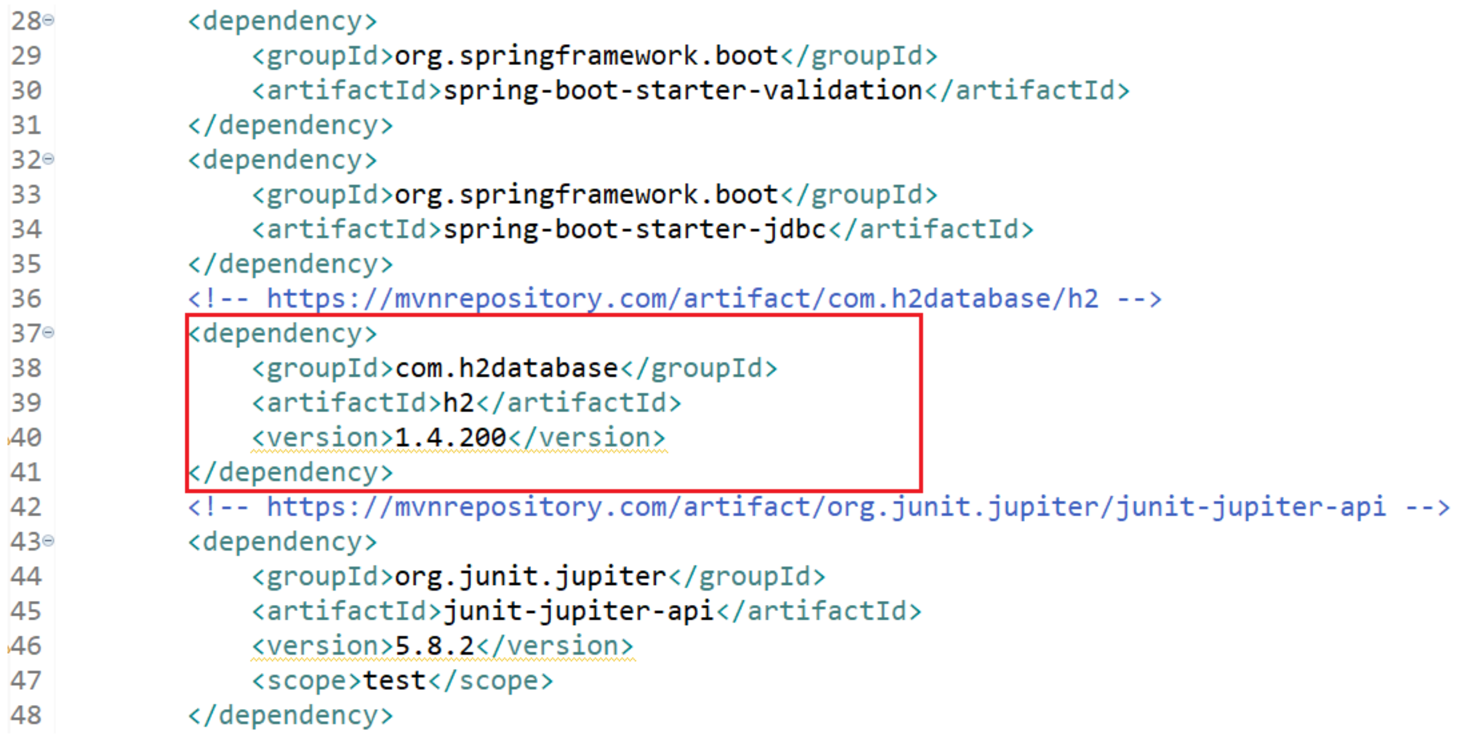
範例下載連結

- https://github.com/jack23h67/demoForTest02

測試程式架構圖

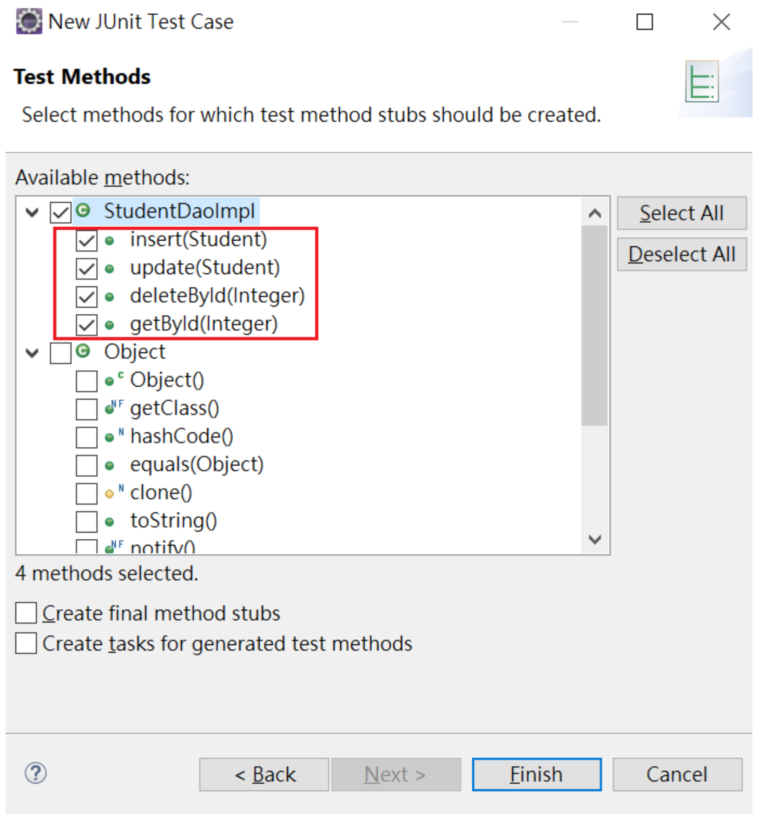

Spring 容器
MockMvc
模擬真實的 API call
Bean
studentController
Bean
studentService
Mock Bean
studentDao
注入 bean 進行單元測試
Mock 測試：使用一個假的 bean 去替換掉正常的 bean，讓這個假的 bean 固定返回我們指定的結果

虛擬資料庫 h2-pom.xml



```xml
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-jdbc</artifactId>
        </dependency>
        <!-- https://mvnrepository.com/artifact/com.h2database/h2 -->
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <version>1.4.200</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>5.8.2</version>
            <scope>test</scope>
        </dependency>
```

Dao 層建立測試程式 (1)


New JUnit Test Case

Test Methods

Select methods for which test method stubs should be created.

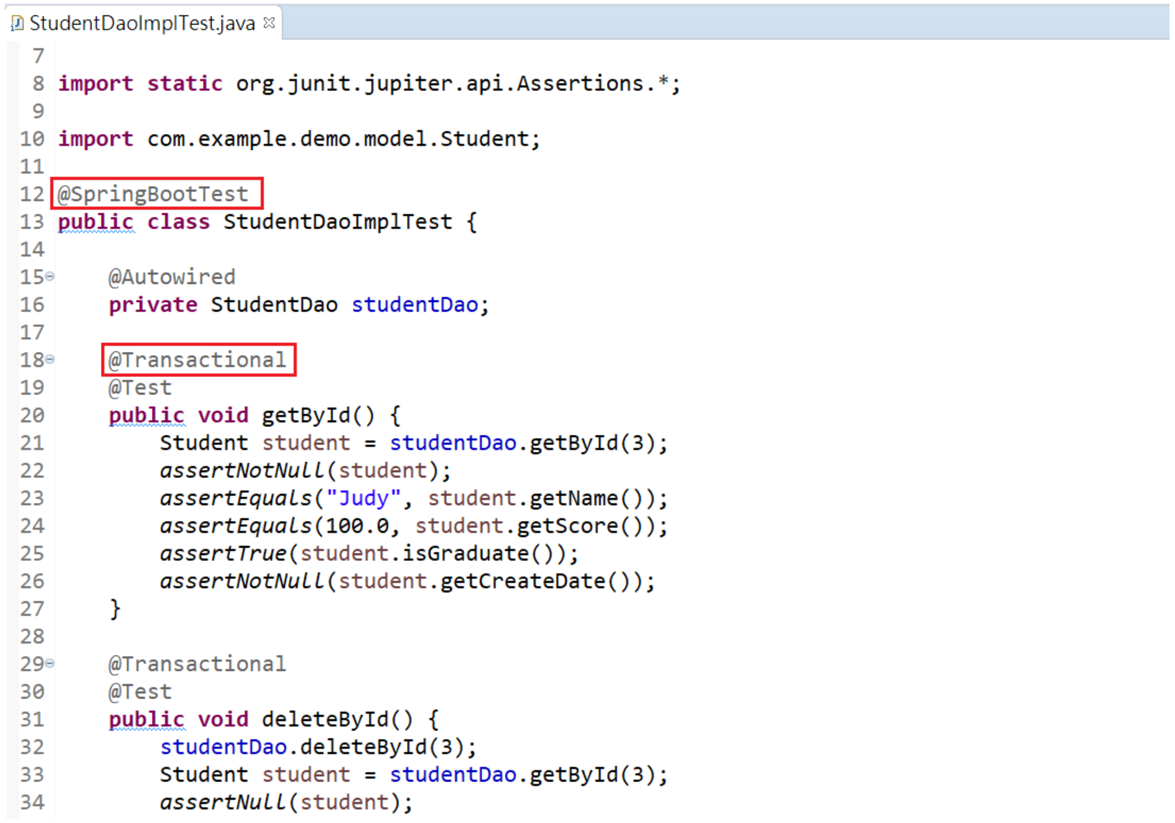
Available methods:

- [✓] StudentDaoImpl
  - [✓] insert(Student)
  - [✓] update(Student)
  - [✓] deleteById(Integer)
  - [✓] getById(Integer)
- [ ] Object
  - [ ] Object()
  - [ ] getClass()
  - [ ] hashCode()
  - [ ] equals(Object)
  - [ ] clone()
  - [ ] toString()
  - [ ] notify()

4 methods selected.

- [ ] Create final method stubs
- [ ] Create tasks for generated test methods

Dao 層建立測試程式 (2) @Transactional



StudentDaoImplTest.java

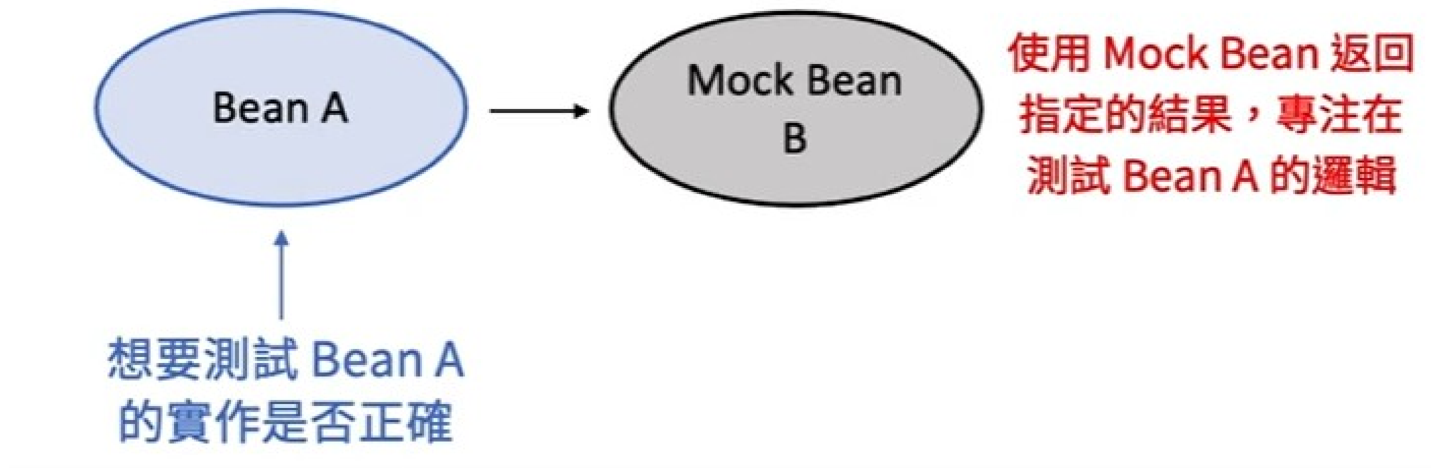
```java

import static org.junit.jupiter.api.Assertions.*;

import com.example.demo.model.Student;

@SpringBootTest
public class StudentDaoImplTest {

    @Autowired
    private StudentDao studentDao;

    @Transactional
    @Test
    public void getById() {
        Student student = studentDao.getById(3);
        assertNotNull(student);
        assertEquals("Judy", student.getName());
        assertEquals(100.0, student.getScore());
        assertTrue(student.isGraduate());
        assertNotNull(student.getCreateDate());
    }

    @Transactional
    @Test
    public void deleteById() {
        studentDao.deleteById(3);
        Student student = studentDao.getById(3);
        assertNull(student);
```

Dao 層
測試檔加上註解 -@Transactional

- @Transactional ：在測試檔的測試方法上，加上此註解後，在執行完此測試方法後， spring boot 就會 rollback 整個方法所執行過的資料庫操作
- 用法：可以加在方法上、也可以加在 class 上
- 用途：在單元測試結束後，會 rollback 所有資料庫操作，將數據恢復原狀
- @Transactional 註解在測試檔與正式程式中，用途不同

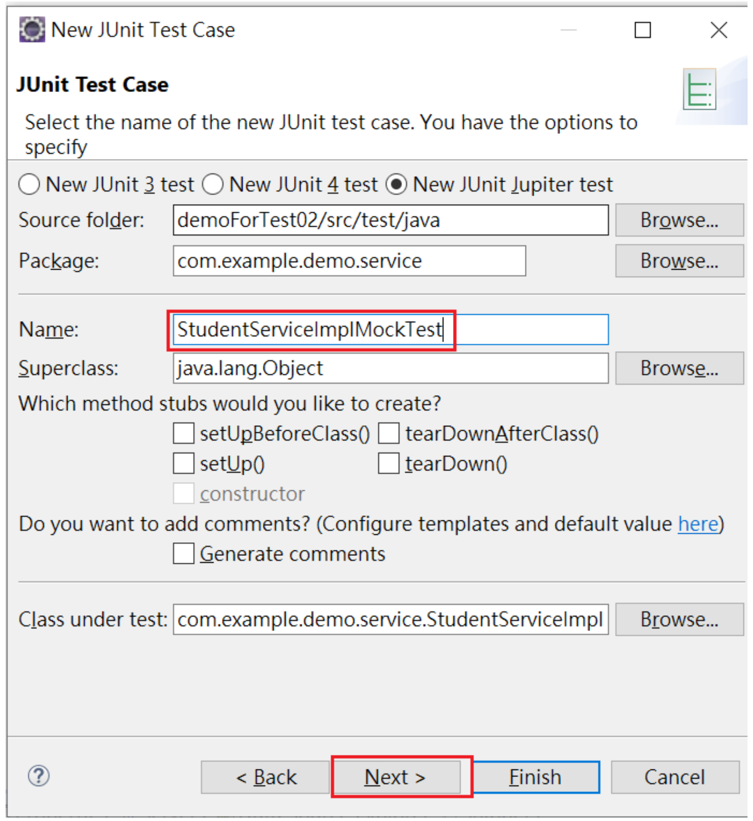
Service 層 Mock 測試

- 目的：避免為了測試某一個單元測試，而去建立了整個 bean 的依賴性
- 作法：創造一個假的 bean ，去替換掉 Spring 容器中原有的 Bean



Bean A
Mock Bean B
使用 Mock Bean 返回指定的結果，專注在測試 Bean A 的邏輯
想要測試 Bean A 的實作是否正確

Service 層建立測試程式 (1)


New JUnit Test Case

JUnit Test Case

Select the name of the new JUnit test case. You have the options to specify

( ) New JUnit 3 test ( ) New JUnit 4 test [✓ New JUnit Jupiter test]

Source folder:	demoForTest02/src/test/java	Browse...
Package:	com.example.demo.service	Browse...
Name:	StudentServiceImplMockTest	
Superclass:	java.lang.Object	Browse...

Which method stubs would you like to create?

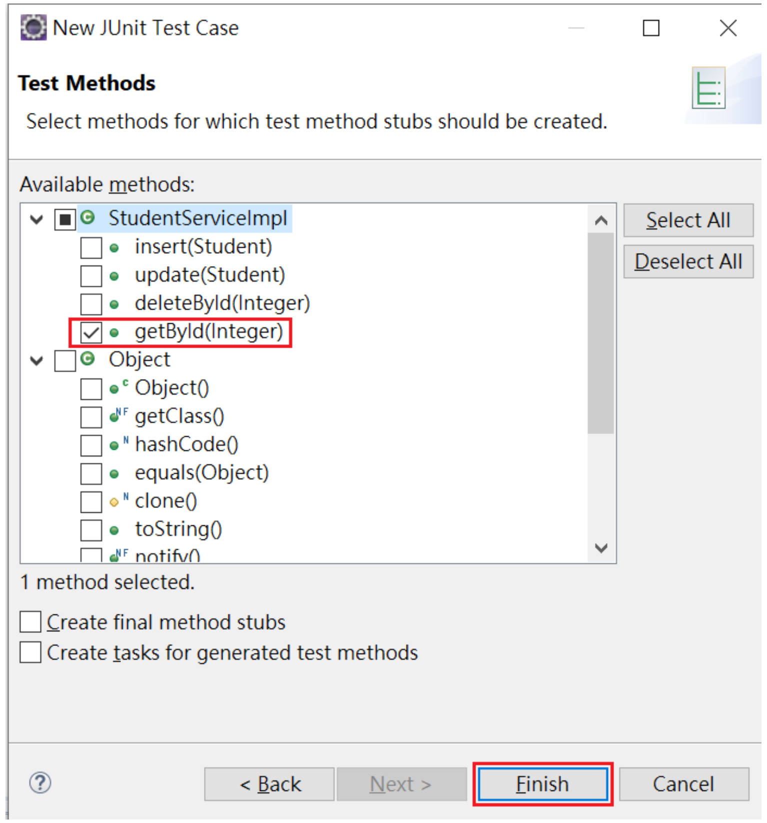
☐ setUpBeforeClass() ☐ tearDownAfterClass()
☐ setUp() ☐ tearDown()
☐ constructor

Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

Class under test: com.example.demo.service.StudentServiceImpl Browse...

Service 層建立測試程式 (2)


New JUnit Test Case

Test Methods

Select methods for which test method stubs should be created.

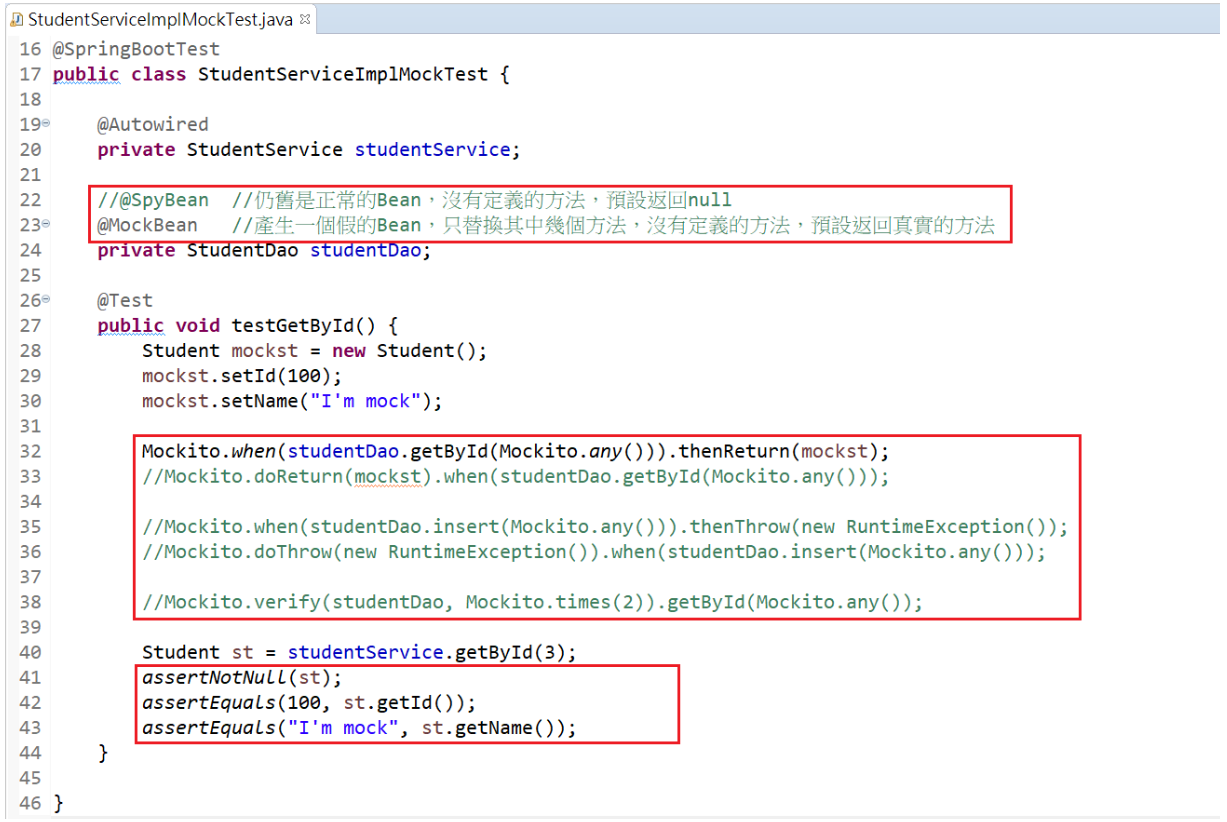
Available methods:

- [■] StudentServiceImpl
  - [ ] insert(Student)
  - [ ] update(Student)
  - [ ] deleteById(Integer)
  - [✓] getById(Integer)
- [ ] Object
  - [ ] Object()
  - [ ] getClass()
  - [ ] hashCode()
  - [ ] equals(Object)
  - [ ] clone()
  - [ ] toString()
  - [ ] notify()

1 method selected.

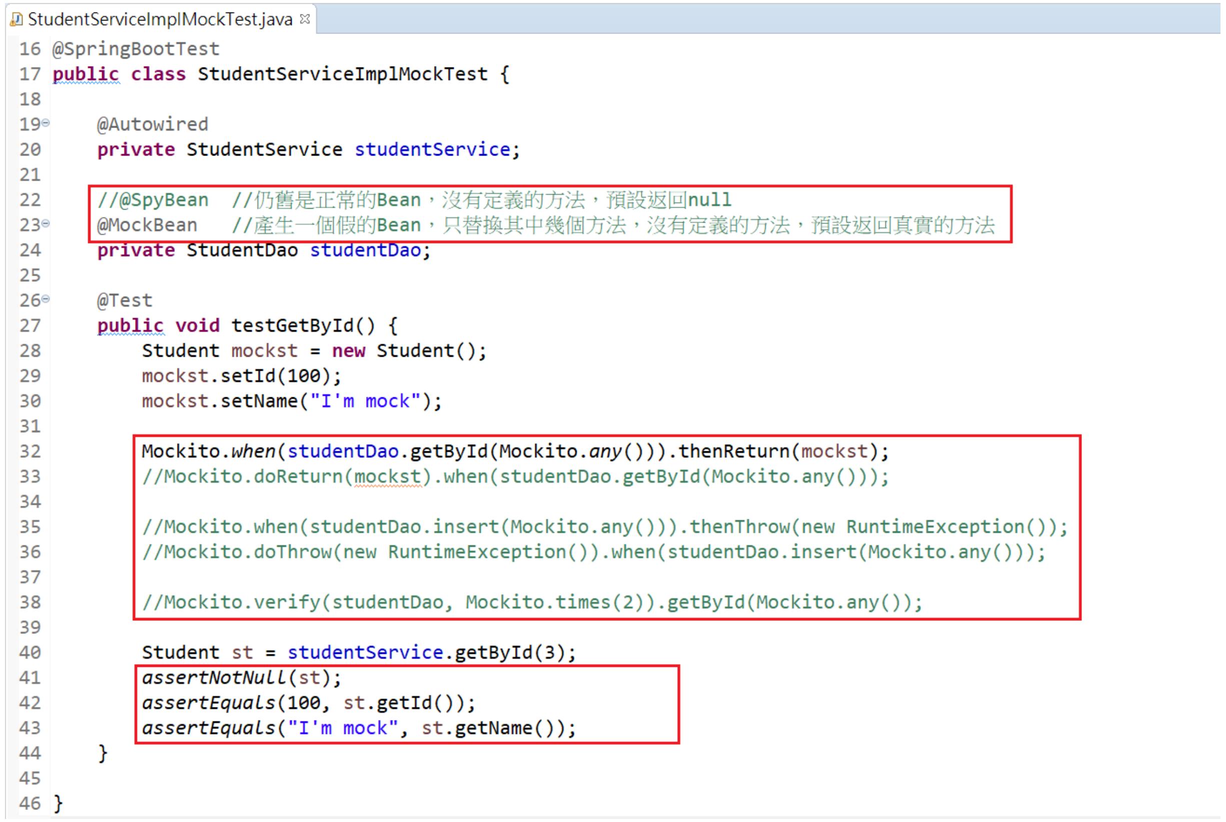
[ ] Create final method stubs
[ ] Create tasks for generated test methods

Service 層建立測試程式 (3)


StudentServiceImplMockTest.java

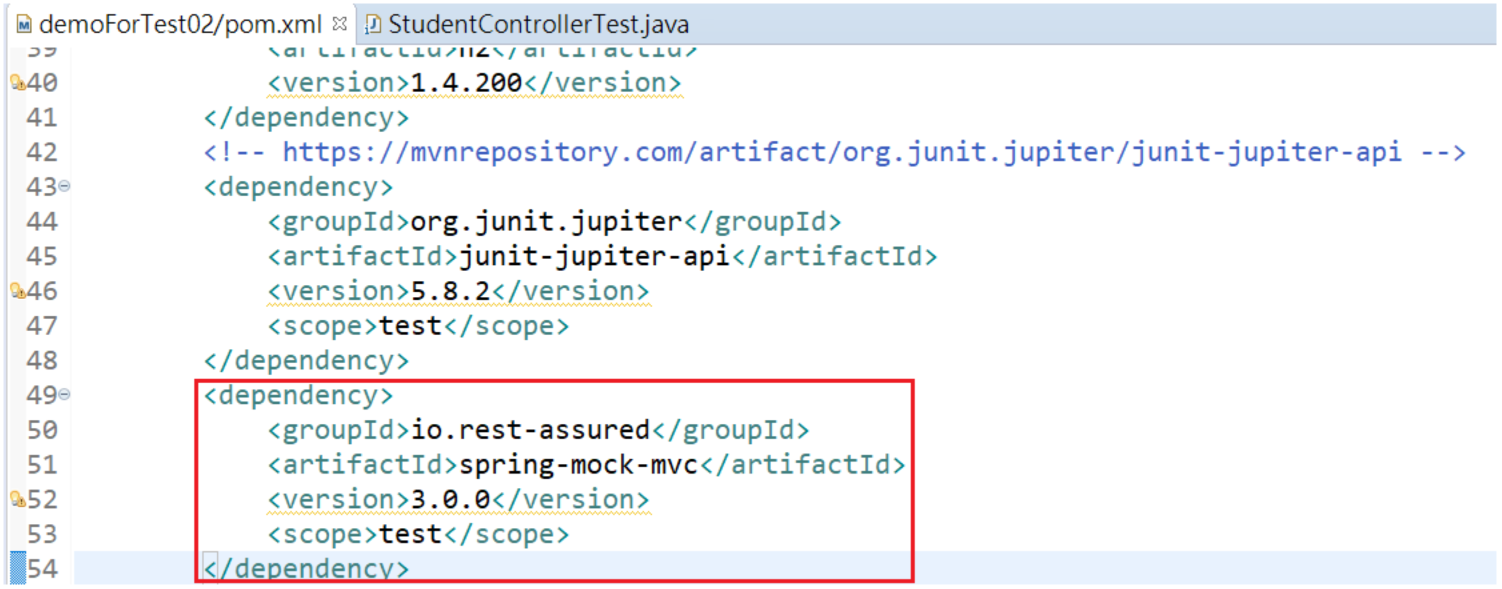
```java
@SpringBootTest
public class StudentServiceImplMockTest {

    @Autowired
    private StudentService studentService;

    //@SpyBean  //仍舊是正常的Bean，沒有定義的方法，預設返回null
    @MockBean   //產生一個假的Bean，只替換其中幾個方法，沒有定義的方法，預設返回真實的方法
    private StudentDao studentDao;

    @Test
    public void testGetById() {
        Student mockst = new Student();
        mockst.setId(100);
        mockst.setName("I'm mock");

        Mockito.when(studentDao.getById(Mockito.any())).thenReturn(mockst);
        //Mockito.doReturn(mockst).when(studentDao.getById(Mockito.any()));

        //Mockito.when(studentDao.insert(Mockito.any())).thenThrow(new RuntimeException());
        //Mockito.doThrow(new RuntimeException()).when(studentDao.insert(Mockito.any()));

        //Mockito.verify(studentDao, Mockito.times(2)).getById(Mockito.any());

        Student st = studentService.getById(3);
        assertNotNull(st);
        assertEquals(100, st.getId());
        assertEquals("I'm mock", st.getName());
    }

}
```

@MockBean 、 @SpyBean Mockito when() thenReturn()


StudentServiceImplMockTest.java

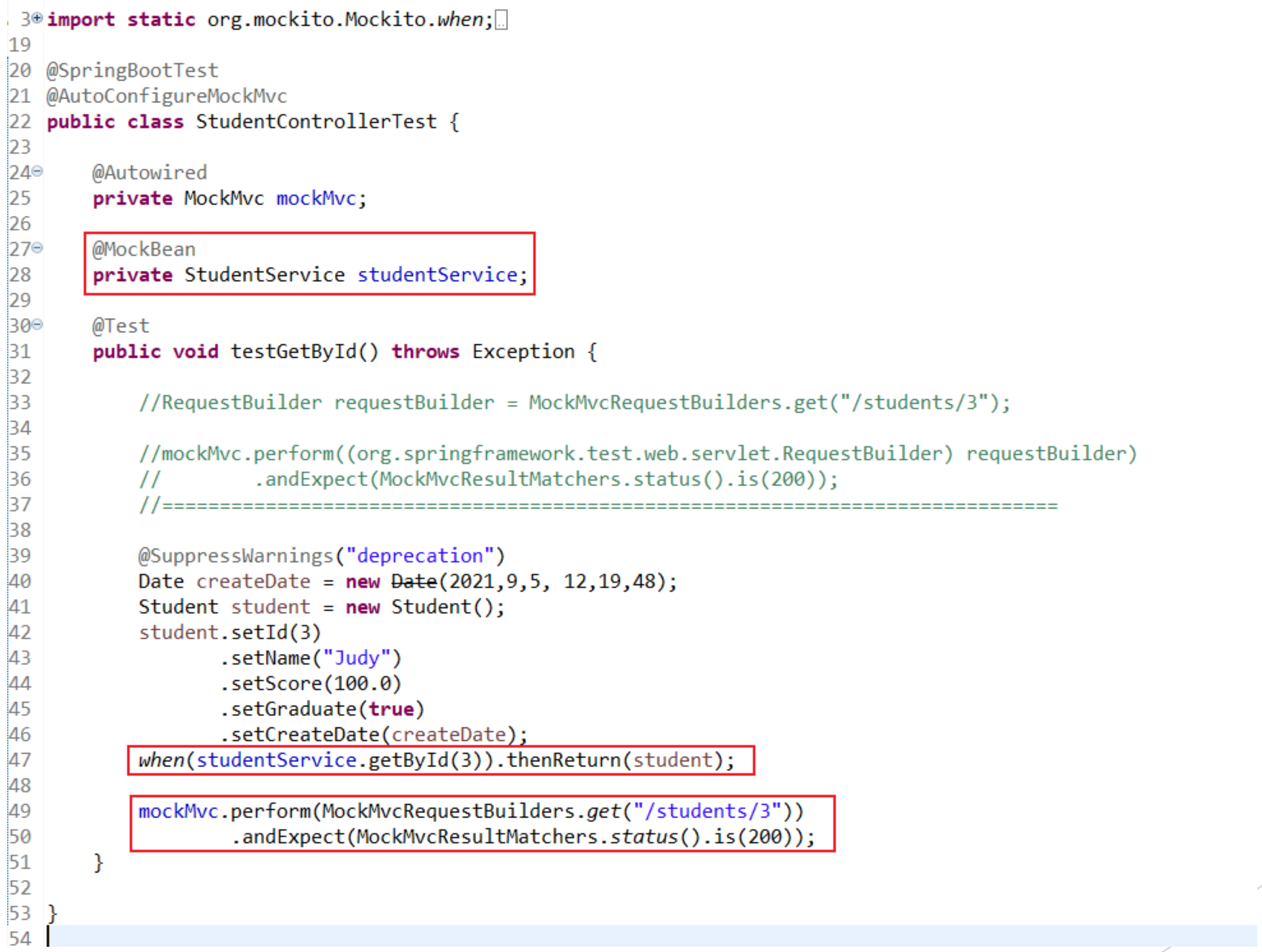
```java
@SpringBootTest
public class StudentServiceImplMockTest {

    @Autowired
    private StudentService studentService;

    //@SpyBean  //仍舊是正常的Bean，沒有定義的方法，預設返回null
    @MockBean   //產生一個假的Bean，只替換其中幾個方法，沒有定義的方法，預設返回真實的方法
    private StudentDao studentDao;

    @Test
    public void testGetById() {
        Student mockst = new Student();
        mockst.setId(100);
        mockst.setName("I'm mock");

        Mockito.when(studentDao.getById(Mockito.any())).thenReturn(mockst);
        //Mockito.doReturn(mockst).when(studentDao.getById(Mockito.any()));

        //Mockito.when(studentDao.insert(Mockito.any())).thenThrow(new RuntimeException());
        //Mockito.doThrow(new RuntimeException()).when(studentDao.insert(Mockito.any()));

        //Mockito.verify(studentDao, Mockito.times(2)).getById(Mockito.any());

        Student st = studentService.getById(3);
        assertNotNull(st);
        assertEquals(100, st.getId());
        assertEquals("I'm mock", st.getName());
    }

}
```

Controller 層建立測試程式 (1)-pom.xml



```
            <artifactId>h2</artifactId>
            <version>1.4.200</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>5.8.2</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>io.rest-assured</groupId>
            <artifactId>spring-mock-mvc</artifactId>
            <version>3.0.0</version>
            <scope>test</scope>
        </dependency>
```

Controller 層建立測試程式 (2)- 測試程式 MockMvc 、 perform() 、 andExpect()



```java
import static org.mockito.Mockito.when;

@SpringBootTest
@AutoConfigureMockMvc
public class StudentControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @MockBean
    private StudentService studentService;

    @Test
    public void testGetById() throws Exception {

        //RequestBuilder requestBuilder = MockMvcRequestBuilders.get("/students/3");

        //mockMvc.perform((org.springframework.test.web.servlet.RequestBuilder) requestBuilder)
        //          .andExpect(MockMvcResultMatchers.status().is(200));
        //==============================================================================

        @SuppressWarnings("deprecation")
        Date createDate = new Date(2021,9,5, 12,19,48);
        Student student = new Student();
        student.setId(3)
               .setName("Judy")
               .setScore(100.0)
               .setGraduate(true)
               .setCreateDate(createDate);
        when(studentService.getById(3)).thenReturn(student);

        mockMvc.perform(MockMvcRequestBuilders.get("/students/3"))
                .andExpect(MockMvcResultMatchers.status().is(200));
    }

}
```

Controller 層建立測試程式 (3)- 測試程式 MockMvc 、 perform() 、 andExpect()

- MockMvc: 支援 Spring MVC 伺服器端測試的主要入口點
- perform(request): 此方法為要做一個請求的建立，這是一個模擬請求的方式
- andExpect(): 此方法用來驗證結果

測試驅動開發應用 -Lab

- 完成 service 層增刪改查的操作測試檔
- 查找一下在 Controller 層中的 andExpect() 的其他用法
- 進階作業：完成 Controller 層的增刪改操作測試檔

參考資源

- https://kucw.github.io/blog/2020/2/spring-unit-test-mockito/
- https://hahow.in/courses/5fe22e7fe810e10fc483dd78/main
- https://ithelp.ithome.com.tw/articles/10196729

thanks for listening