Spring testing
infoShare ACADEMY

Hello

Mateusz Cygert

1. Spring Test

Spring test

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class FooTest {
   // test code...
}
```

Wstrzykiwanie zależności

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class MovieServiceTest {

   @Autowired
   private MovieService movieService;

}
```

Przykładowy test

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class MovieServiceTest {
   @Autowired
   private MovieService movieService;
   @Autowired
   private MovieRepository movieRepository;

   @Test
   public void shouldSaveNewMovie() {
      Movie movie = new Movie();
      Movie savedMovie = movieService.addNewMovie(movie);

      assertThat(movieRepository.findById(savedMovie.getId())).isPresent()
                                                              .hasValue(movie);
   }
}
```

Ćwiczenie 1.

1. W pakiecie com.example.test.services stwórz klasę MovieIdGeneratorService
2. Oznacz klasę stereotypem springowym @Service
3. Dodaj publiczną metodę o nazwie generateId zwracającą typ Long
4. Każdorazowe wywołanie metody powinno zwrócić nową, wcześniej nie zwracaną liczbę.

Ćwiczenie 2.

1. Stwórz test dla klasy MovieIdGeneratorService
2. Oznacz klasę za pomocą adnotacji:
   a. @RunWith(SpringRunner.class)
   b. @SpringBootTest
3. W teście wstrzyknij beana MovieIdGeneratorService.
4. Przetestuj metodę ‘generateId’.

Ćwiczenie 3.

1. W pakiecie com.example.test.services stwórz klasę MovieInsertService.
2. Oznacz klasę stereotypem springowym @Service.
3. Wstrzyknij bean MovieRepository.
4. Dodaj metodę ‘addNewMovie’ przyjmującą parametr typu Movie.
5. Przekazany film zapisz w bazie danych wykorzystując MovieRepository.

Ćwiczenie 4.

1. Napisz testy do klasy MovieInsertService

Ćwiczenie 5.

1. W serwisie MovieInsertService przed zapisem filmu do bazy danych:
   a. Wygeneruj nowy id dla filmu wykorzystując do tego serwis MovieIdGeneratorService.
   b. Zwaliduj obiekt filmu wykorztując metodę validateMovie w klasie MovieValidator.
2. Popraw testy.

Ćwiczenie 6.

1. Poczekaj na prowadzącego :)

Ćwiczenie 7.

1. Popraw testy.
2. Zastanów się, dlaczego zmiana w serwisie niezależnym od MovieInsertService spowodowała błędy w testach.
3. Czy masz pomysł jak moglibyśmy tego uniknąć?

Mockowanie

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class MovieInsertServiceTest {

    @MockBean
    private MovieValidator movieValidator;

    @MockBean
    private MovieIdGeneratorService movieIdGeneratorService;

    @MockBean
    private MovieRepository movieRepository;

    @Autowired
    private MovieService movieService;
```

Mockowanie

Jeżeli metoda ‘save’ zostanie zawołana z parametrem ‘movie’, to zwróć obiekt ‘movie’.

```
Mockito.when(movieRepository.save(movie)).thenReturn(movie);
```

Jeżeli metoda ‘findById’ zostanie zawołana z jakimkolwiek parametrem, to zawsze zwróć ‘optional’ obiektu ‘movie’.

```
Mockito.when(movieRepository.findById(any())).thenReturn(Optional.of(movie));
```

Zweryfikuj, że metoda validateMovie została zawołana dokładnie raz, z parametrem typu ‘Movie’.

```
Mockito.verify(movieValidator, Mockito.times(1)).validateMovie(any(Movie.class));
```

Ćwiczenie 8.

1. Stwórz klasę MovieInsertServiceMockTest.
2. Przerób test napisany w klasie MovieInsertServiceTest, tak aby MovieInsertService wykorzystywał jedynie zmockowane beany.

Mockowanie bez kontekstu Springa

```
@RunWith(MockitoJUnitRunner.class)
public class MovieServiceTest {

    @Mock
    private MovieValidator movieValidator;

    @Mock
    private MovieIdGeneratorService movieIdGeneratorService;

    @Mock
    private MovieRepository movieRepository;

    @InjectMocks
    private MovieService movieService;
```

Ćwiczenie 9.

1. Stwórz klasę MovieInsertServiceMockWithoutContextTest.
2. Przerób test napisany w klasie MovieInsertServiceMockTest, tak aby test ten nie musiał stawiać pełnego kontekstu springowego.

Dzieki

mateusz.cygert@gmail.com