Spring Professional Exam Tutorial v5.0

Question 41

Question 41 - How do you perform integration testing with @SpringBootTest for a web application?

Integration Test by definition, should check interactions between few components of the system (at least two real, not-mocked components) to check if those components are delivering expected functionalities when working together. In each case when writing Integration Test you should decide how many components should interact in the test for it to be meaningful. Usually you should decide on smallest possible amount of components that are enough to test specific functionality. Components that are not meaningful can be omitted, or mocked with usage of `@MockBean` annotation.

Web components tests (Controller Tests, Rest Controller Tests), if tested in Integration way, should be written in a way for test to make a HTTP Request and check HTTP Response. This kind of approach results in meaningful test, which delivers feedback that actually checks if component works correctly.

Question 41 - How do you perform integration testing with @SpringBootTest for a web application?

Spring Boot allows you to write Integration Tests for Web Components in two ways:

- MockMvc

```
@RunWith(SpringRunner.class)
@SpringBootTest
@AutoConfigureMockMvc
public class CityControllerWebMockMvcTest {

    @Autowired
    private MockMvc mvc;

    @Test
    public void should...() throws Exception {
        ...
    }
}
```

- Embedded Container

```
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
public class CityControllerWebIntegrationTest {

    @LocalServerPort
    private int port;

    @Autowired
    private TestRestTemplate restTemplate;

    @Test
    public void should...() {
        ...
    }
}
```