



Week 2: Microservices

Unit 3: Creating Automated Component Tests

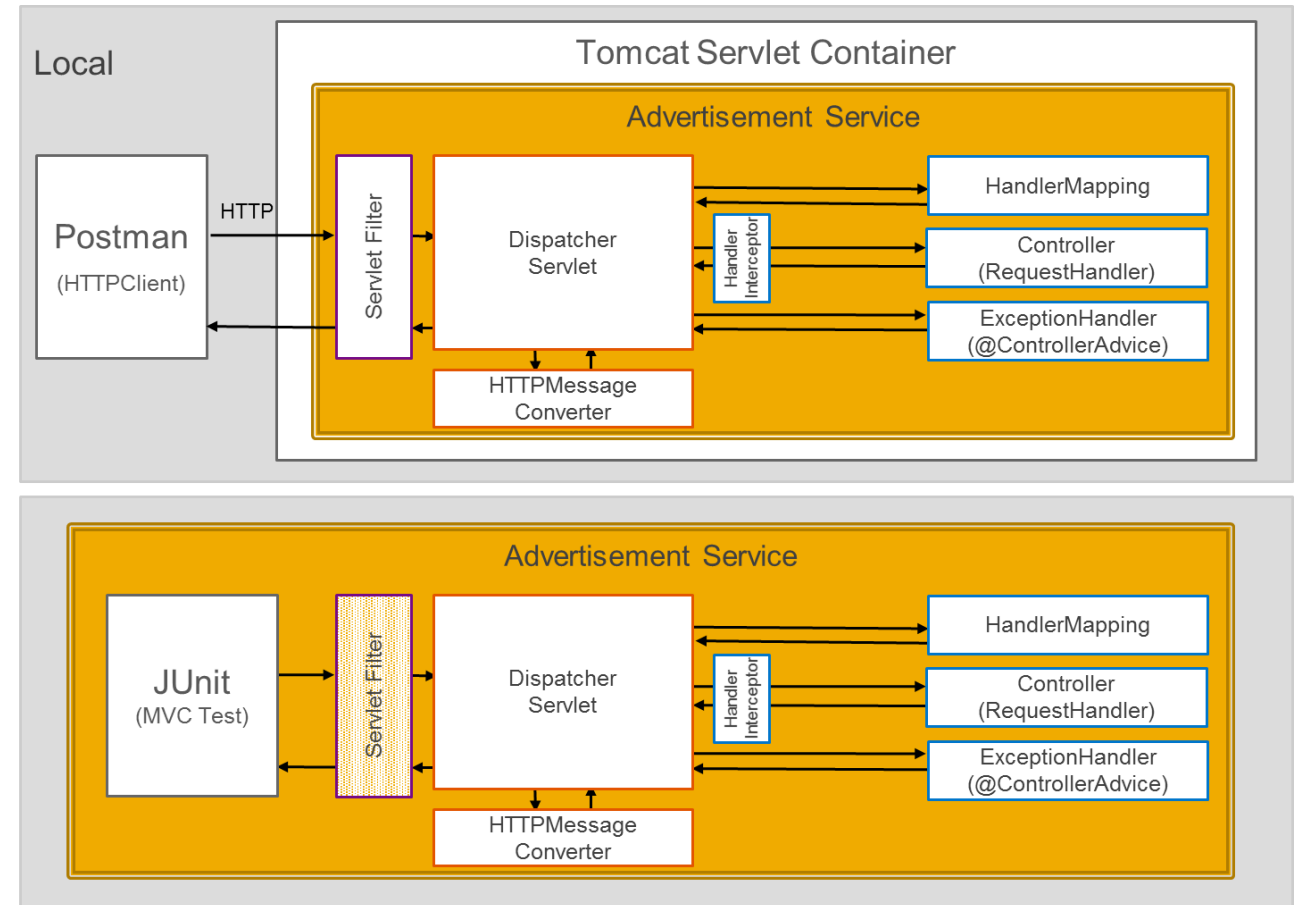
Creating Automated Component Tests

Controller Test with Spring MVC Test

Spring Web MVC Test

- Effective way of testing controllers: perform requests and generate responses through the actual *DispatcherServlet*
- No Servlet Container ⇒ fast feedback
- Offers comprehensive set of [mock objects](#)

[Differences to end-to-end integration tests](#)



Creating Automated Component Tests

Controller Test with Spring MVC Test

Example: Spring MVC Test class

```
@RunWith(SpringJUnit4ClassRunner.class) // same to @RunWith(SpringRunner.class)
@ContextConfiguration(classes = { WebAppContextConfig.class })
@WebAppConfiguration
public class ControllerTest {

    @Inject
    private WebApplicationContext applicationContext; // reused for each test
    private MockMvc mockMvc;

    @Before
    public void setUp() throws Exception {
        this.mockMvc = MockMvcBuilders.webAppContextSetup(applicationContext)
            .addFilter(springSecurityFilterChain).build();
    }
}
```

Creating Automated Component Tests

Controller Test with Spring MVC Test

Example: Spring MVC Test case

```
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.*;

private MockMvc mockMvc;

@Test
public void readById() throws Exception {
    mockMvc.perform(buildGetRequest(id))
        .andExpect(status().isOk())
        .andExpect(content().contentType(APPLICATION_JSON_UTF8))
        .andExpect(jsonPath("$.title", is("buy me"))); // com.jayway.jsonpath:json-path
}

private MockHttpServletRequestBuilder buildGetRequest(String id) throws Exception {
    return get("api/v1.0/ads/" + id).header(AUTHORIZATION, jwt);
}
```

- Use JUnit Test Framework/Runner
- Use Hamcrest and MockMvcResultMatchers for readable assertions

Creating Automated Component Tests

Addendum: Spring MVC Test

- Build requests using the [MockHttpServletRequestBuilder](#)

```
requestBuilder = post("path").content(jsonBody).contentType(APPLICATION_JSON);
```

or

```
requestBuilder = get("path/" + id).header("Header", "Value");
```

- Invoke request, check and retrieve [MockHttpServletResponse](#)

```
MockHttpServletResponse response = mockMvc.perform(requestBuilder)
    .andExpect(status().is2xxSuccessful())
    .andReturn().getResponse();
```

- Create message content (JSON body):

```
String jsonBody = new ObjectMapper().writeValueAsString(anyObject);
```

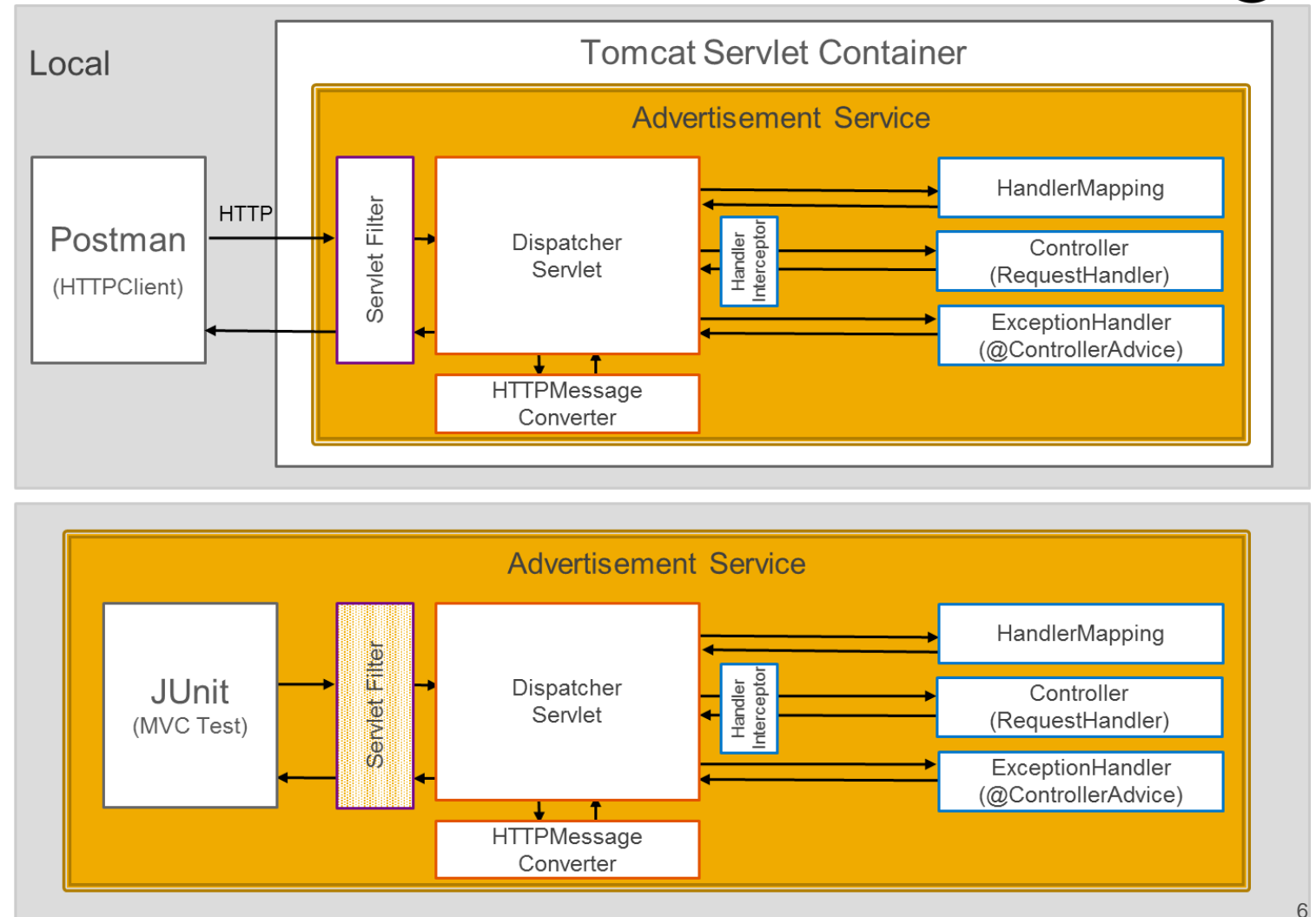
Creating Automated Component Tests

Exercise 4



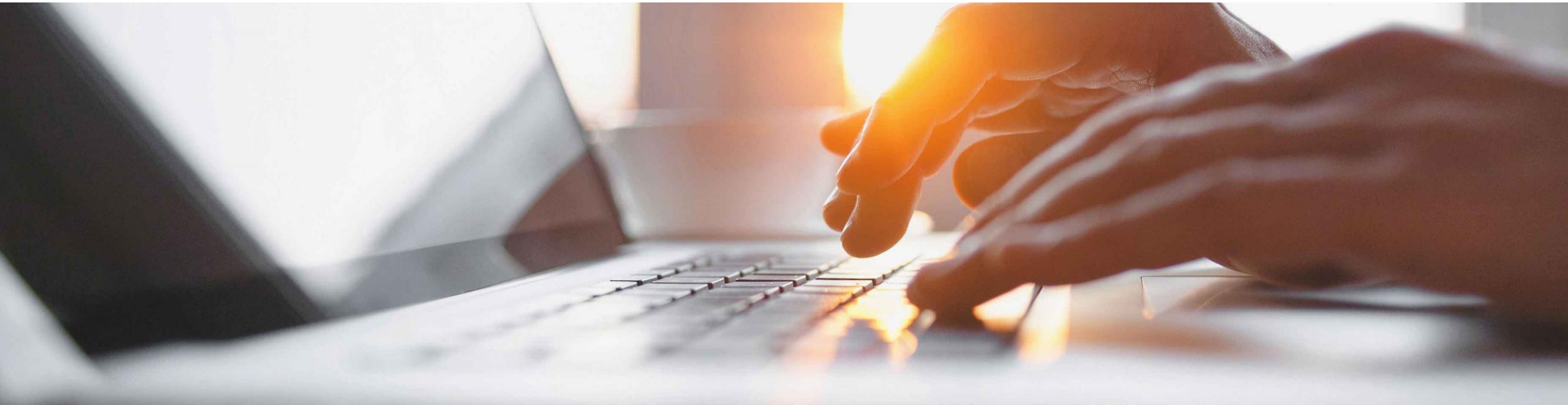
Exercise 4: Create Automated Service Tests

[Optional] Exercise 4 Part 2: Create remaining endpoints test-driven



Creating Automated Component Tests

Review of sample solution



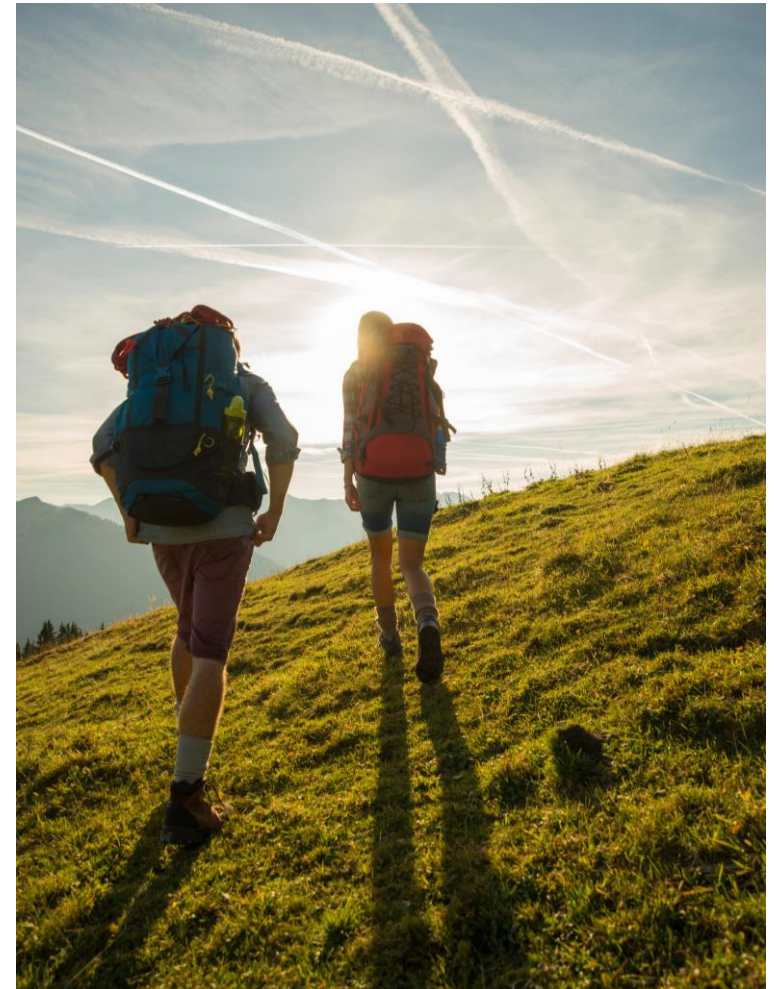
Creating Automated Component Tests

Further reading



Additional Material

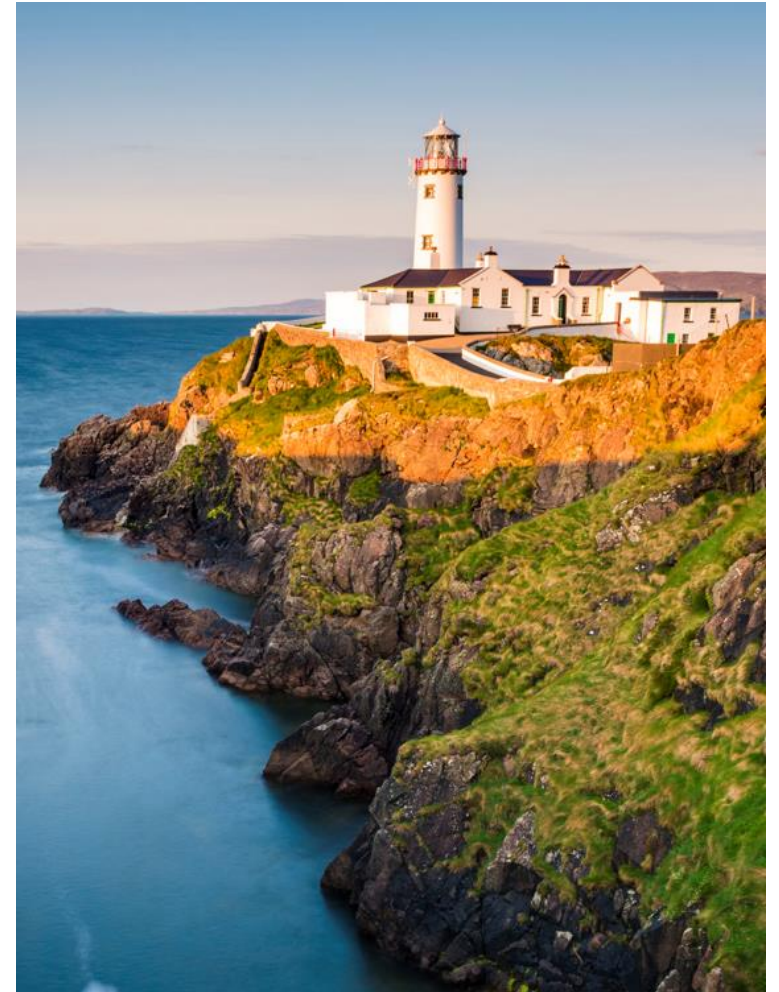
- [Spring Reference: Testing](#)
- [Hamcrest library](#)
- [Hamcrest quick reference](#)
- [JayWay JsonPath library](#)



Creating Automated Component Tests

What you've learned in this unit

- Automated Component Tests
 - What they are
 - How to create them
- Spring MVC Test
 - What it is
 - How you can use it
- What the differences to end-to-end integration tests are



Thank you.

Contact information:

open@sap.com

© 2018 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

See <http://global.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.