Week 2: Microservices

Unit 3: Creating Automated Component Tests

SAP Run Simple

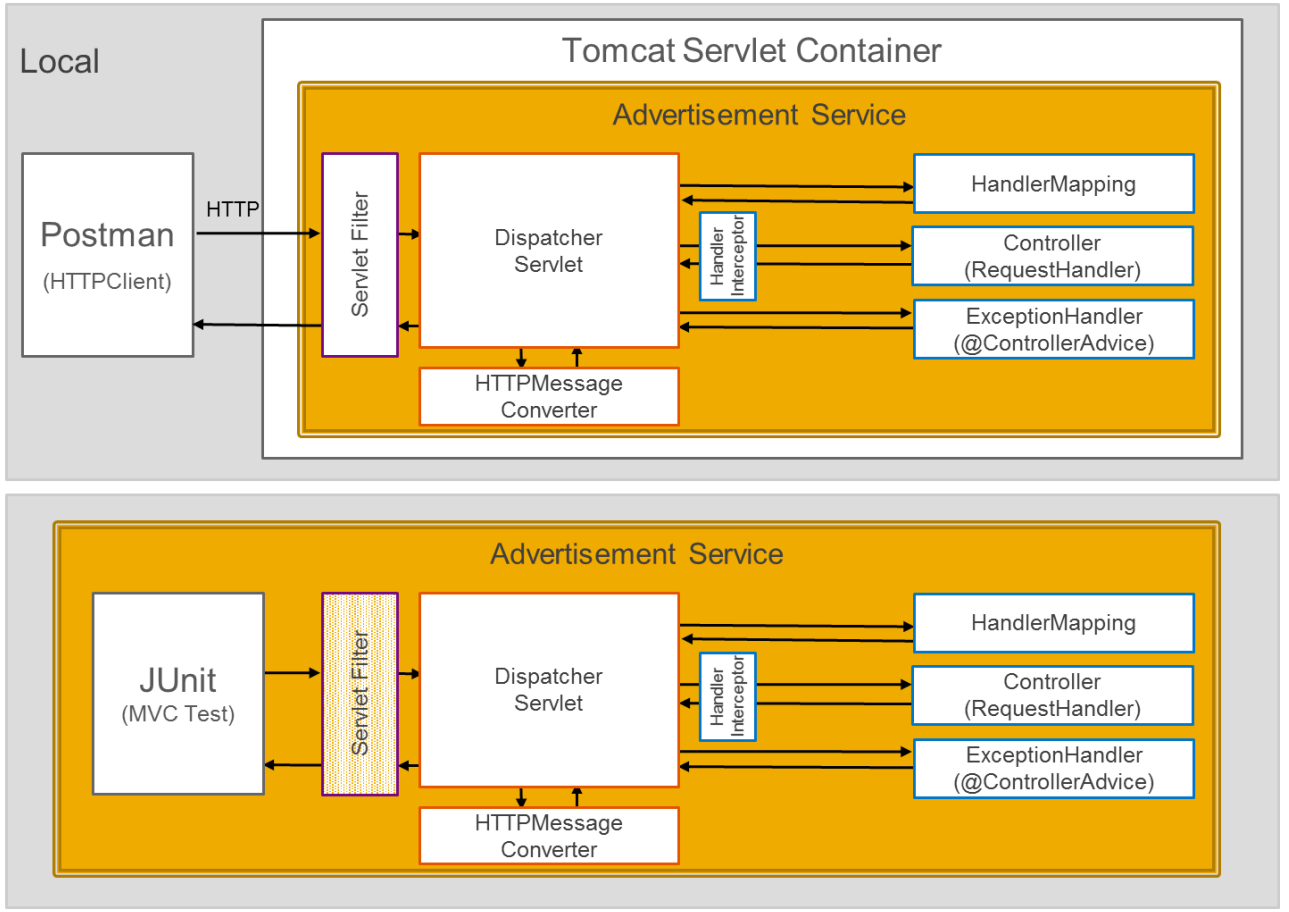
openSAP
open.sap.com

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.sf.test.web.servlet.request.MockMvcRequestBuilders.*;
import static org.sf.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):

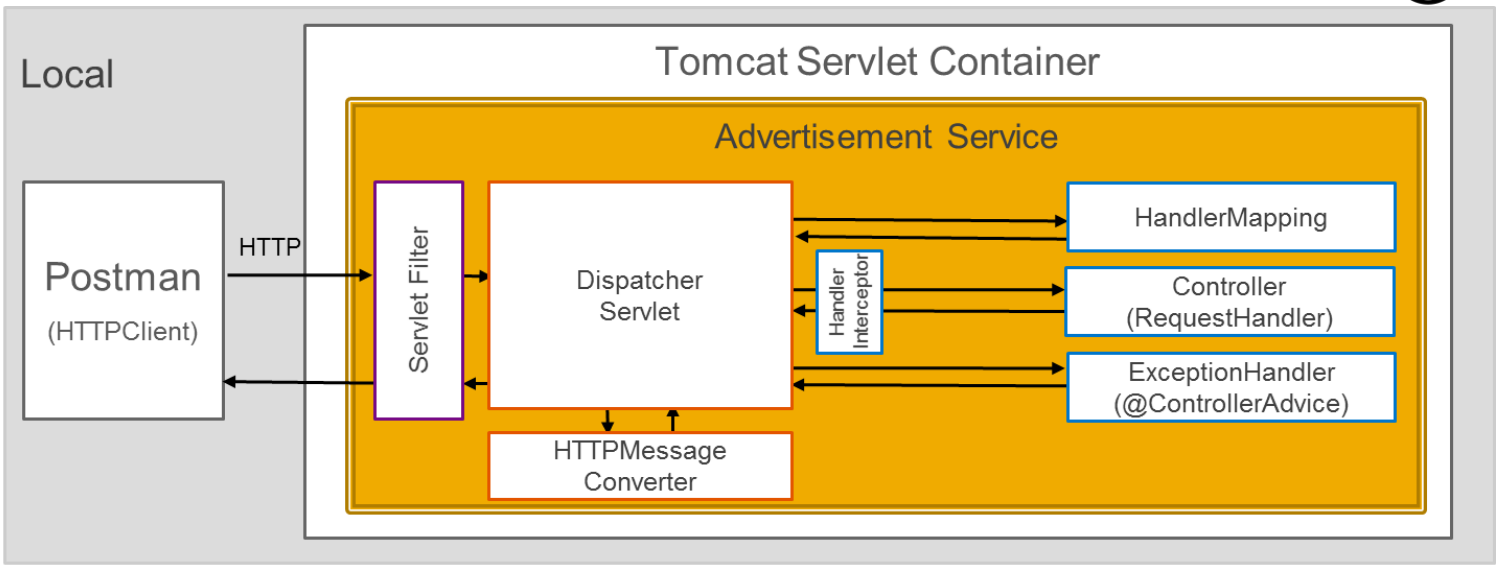
```
String jsonBody = new ObjectMapper().writeValueAsString(anyObject);
```

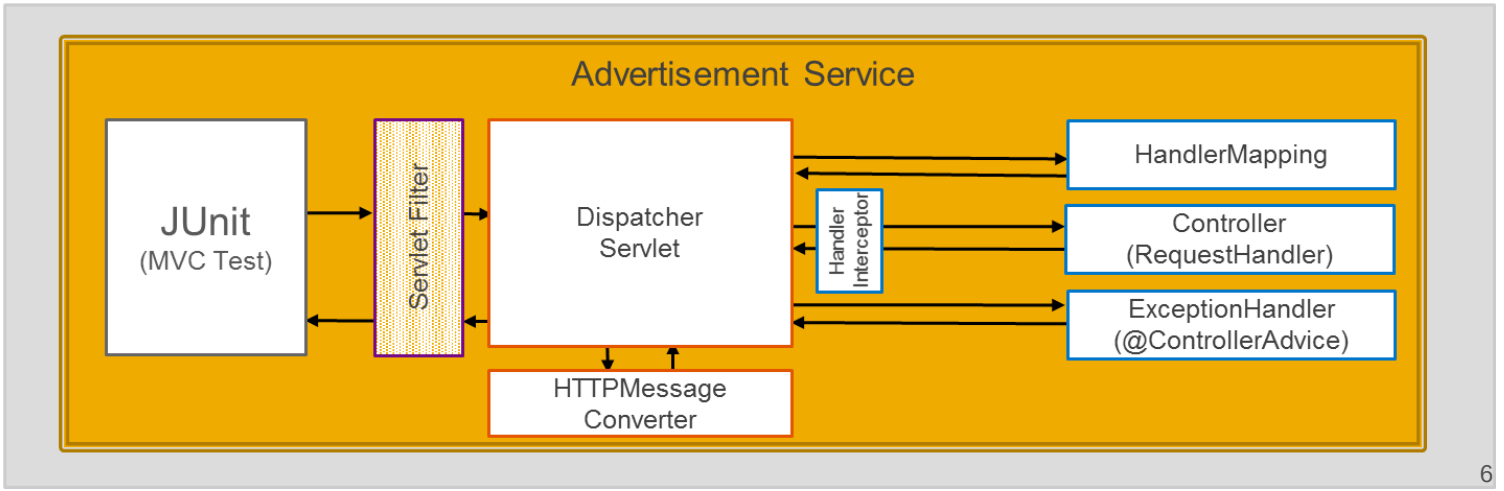
Creating Automated Component Tests

Exercise 4

[Exercise 4:
Create Automated Service Tests]

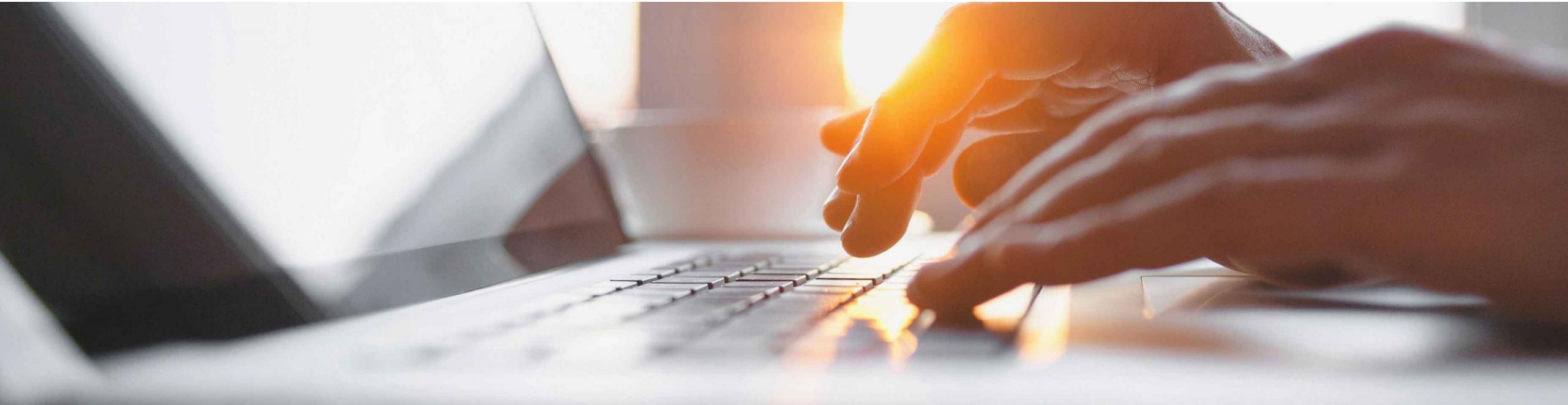
[[Optional] Exercise 4 Part 2:
Create remaining endpoints
test-driven]





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