Building an Integration Test Suite with JUnit 5.0

INTRODUCTION

Steven Haines
PRINCIPAL SOFTWARE ARCHITECT

@geekcap www.geekcap.com

Overview

Integration Testing Strategy

Product Service Integration Tests

Review Service Integration Tests

Inventory Service Integration Tests

"Integration tests determine if independently developed units of software work correctly when they are connected to each other."

Martin Fowler

Why Integration Testing?

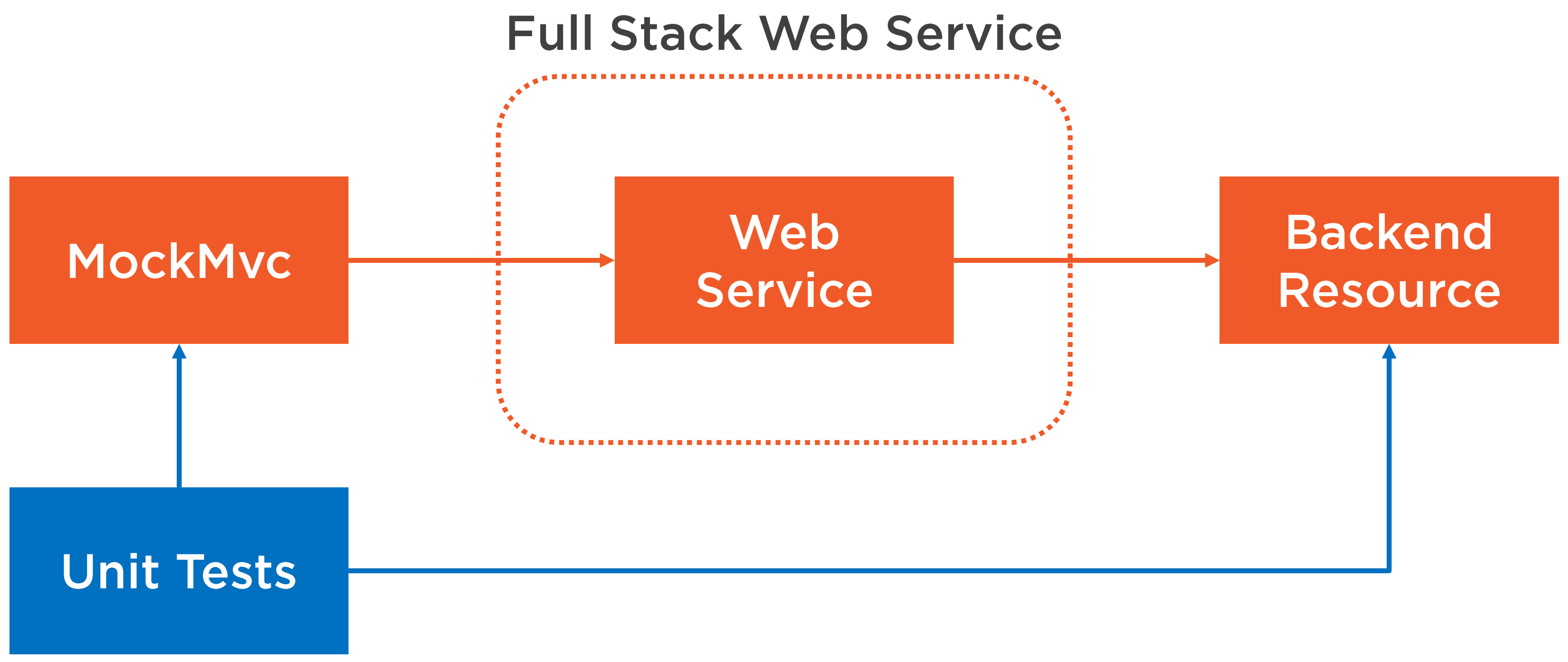
Correct Configuration

Are all of your components wired together properly?

End-to-end Functionality

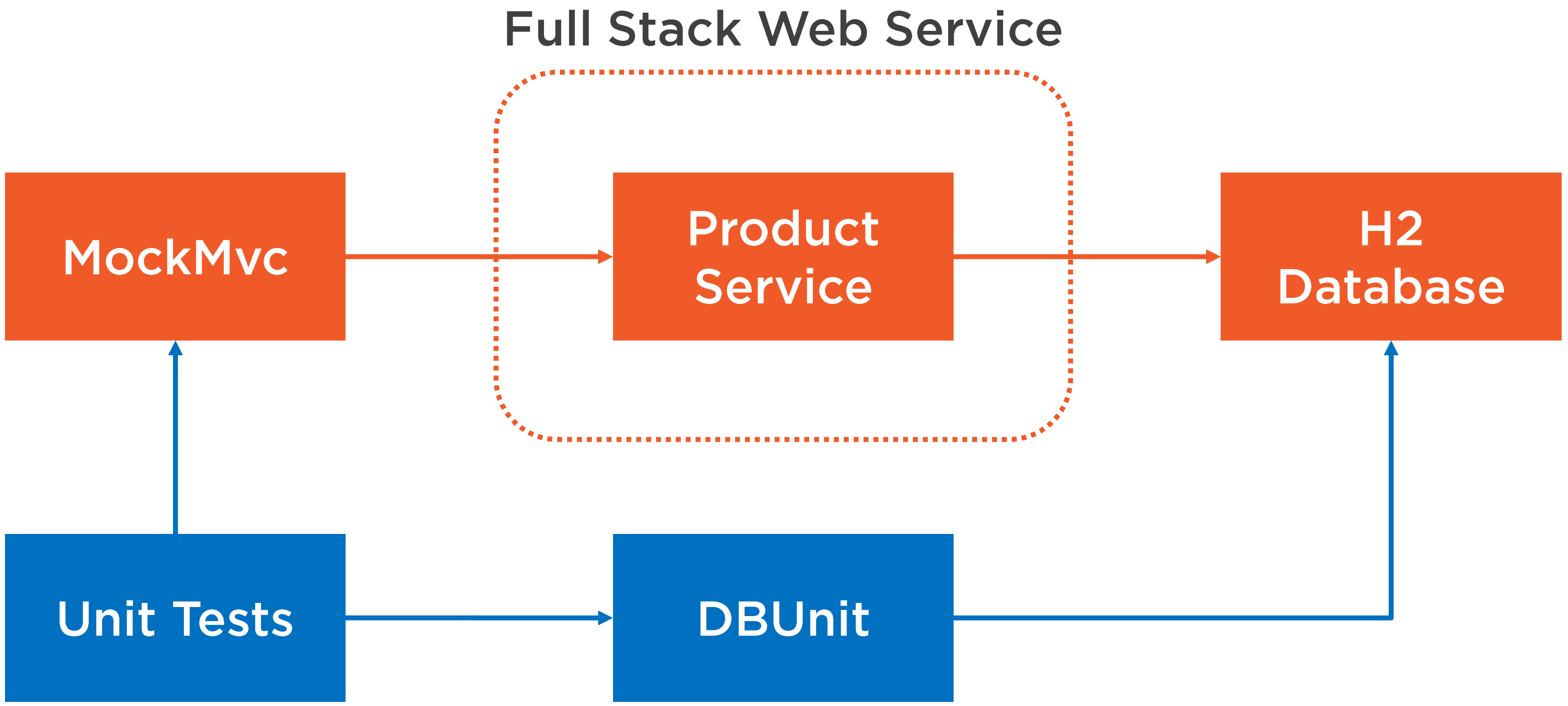
Does your application work correctly end-to-end?

Integration Testing Strategy



Integration Testing the Product Service

Product Service Integration Test



```java
@ExtendWith({DBUnitExtension.class,
             SpringExtension.class})

@SpringBootTest

@ActiveProfiles("test")

@AutoConfigureMockMvc
class ProductServiceIntegrationTest {
  @Autowired
  private MockMvc mockMvc;

  @Autowired
  private DataSource dataSource;

  public ConnectionHolder
                 getConnectionHolder() {
    return () -> dataSource.getConnection();
  }

  @Test
  @DataSet("products.yml")
  void testXXX() { ... }
}
```

◀ Include DBUnitExtension

◀ Full SpringBootTest Context

◀ Spring Profile: “test”

◀ Setup MockMvc

◀ Allow DBUnit to get a connection

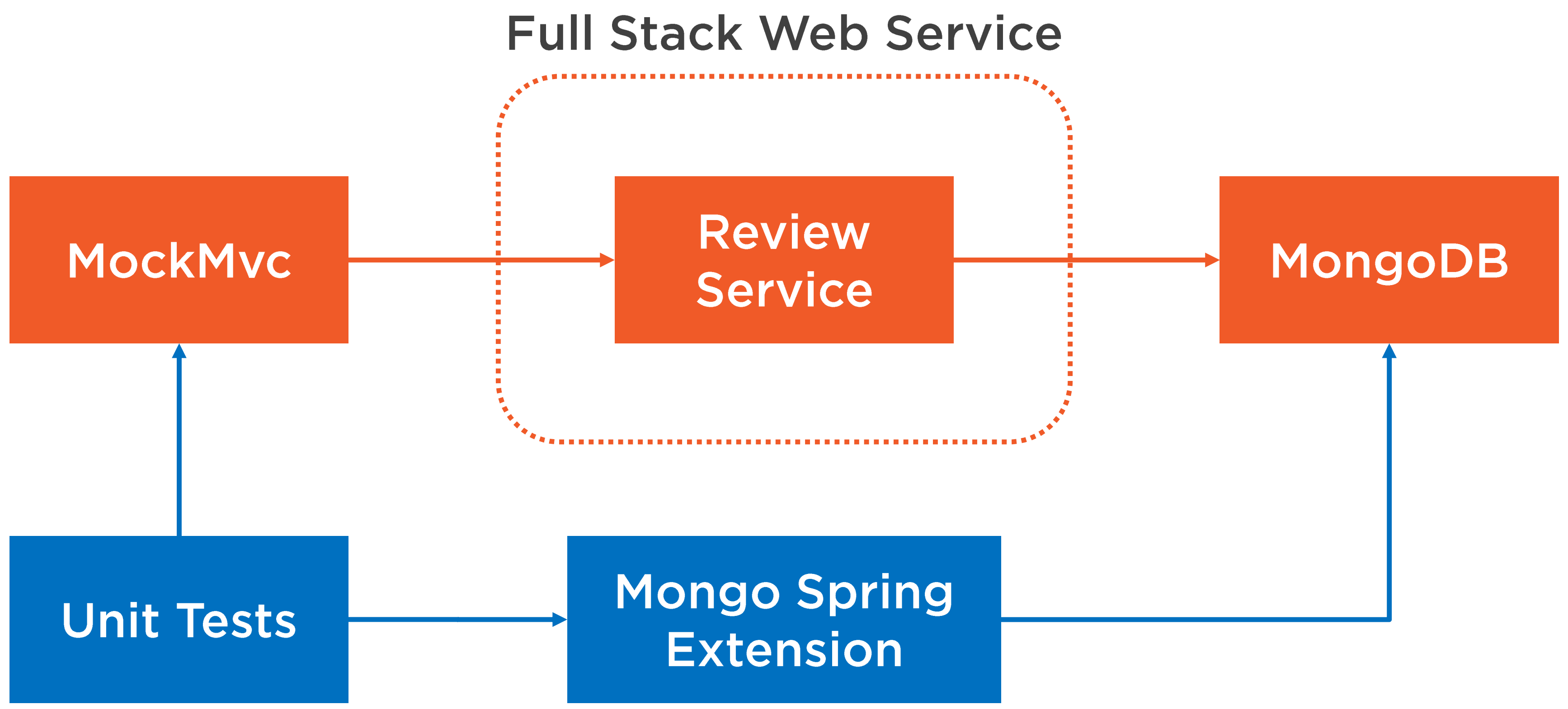
◀ Add DataSet to test

Demo

Product Service Integration Test Code Walkthrough

Integration Testing the Review Service

Review Service Integration Test



```java
@ExtendWith({MongoSpringExtension.class,
             SpringExtension.class})

@SpringBootTest

@AutoConfigureMockMvc
class ReviewServiceIntegrationTest {
  @Autowired
  private MockMvc mockMvc;

  @Autowired
  private MongoTemplate template;
  public MongoTemplate getMongoTemplate() {
    return template;
  }

  @Test
  @MongoDataFile(value = "sample.json",
                 classType = Review.class,
                 collectionName = "Reviews")
  void testXXX() { ... }
}
```

◀ Include MongoSpringExtension

◀ Full SpringBootTest Context

◀ Setup MockMvc

◀ Provide the MongoSpringExtension a MongoTemplate

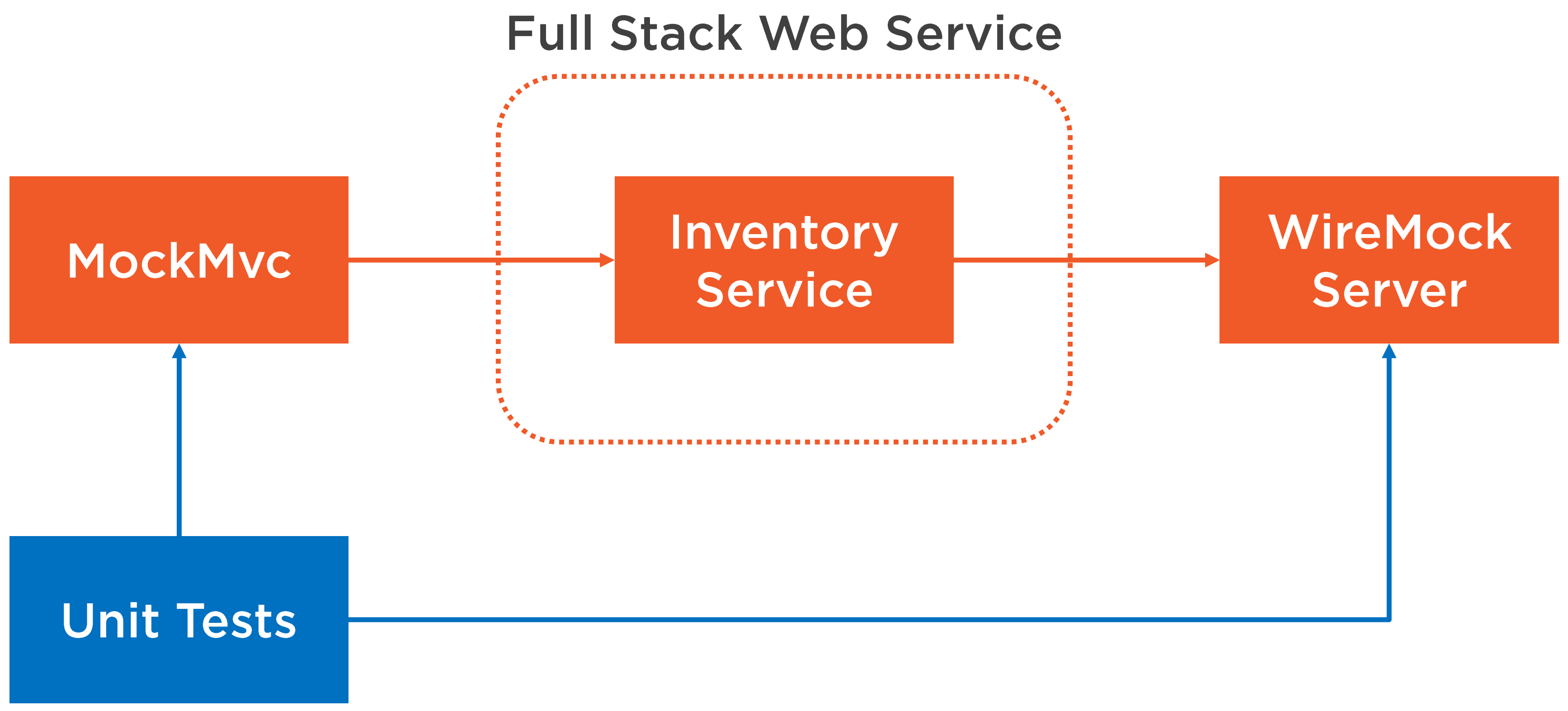
◀ Setup MongoDB

Demo

Review Service Integration Test Code Walkthrough

Integration Testing the Inventory Service

Inventory Service Integration Test



```java
@ExtendWith(SpringExtension.class)
@SpringBootTest

@TestPropertySource(locations =
                  "classpath:test.properties")

@AutoConfigureMockMvc
class InventoryServiceIntegrationTest {
  @Autowired
  private MockMvc mockMvc;

  private WireMockServer wireMockServer;

  @BeforeEach void beforeEach() {
    wireMockServer = new WireMockServer(9999);
    wireMockServer.start();
  }

  @AfterEach void afterEach() {
    wireMockServer.stop();
  }

  @Test
  void testXXX() { ... }
}
```

◀ Full SpringBootTest Context

◀ Override service properties

◀ Configure MockMvc

◀ Define WireMockServer

◀ Create and start WireMockServer

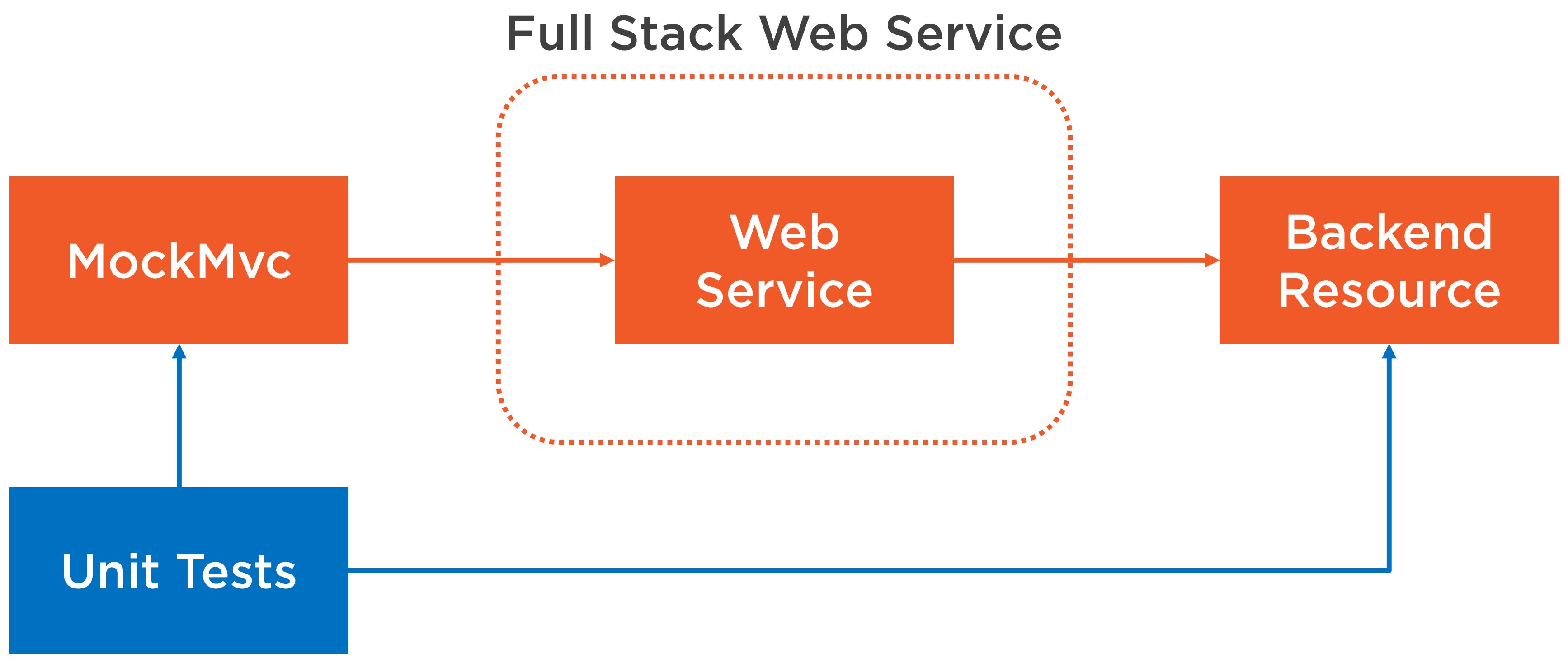
◀ Stop WireMockServer

Demo

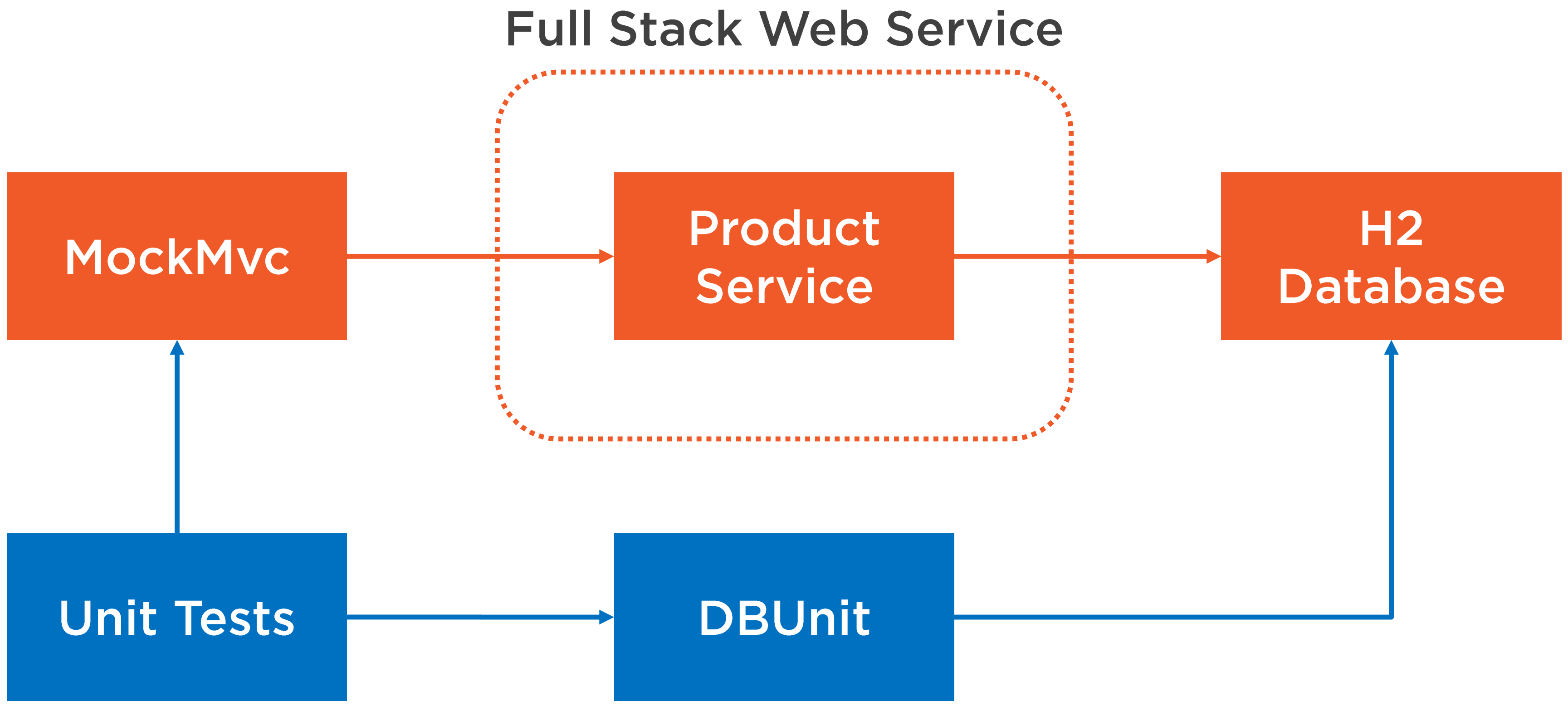
Inventory Service Integration Test Code Walkthrough

Summary

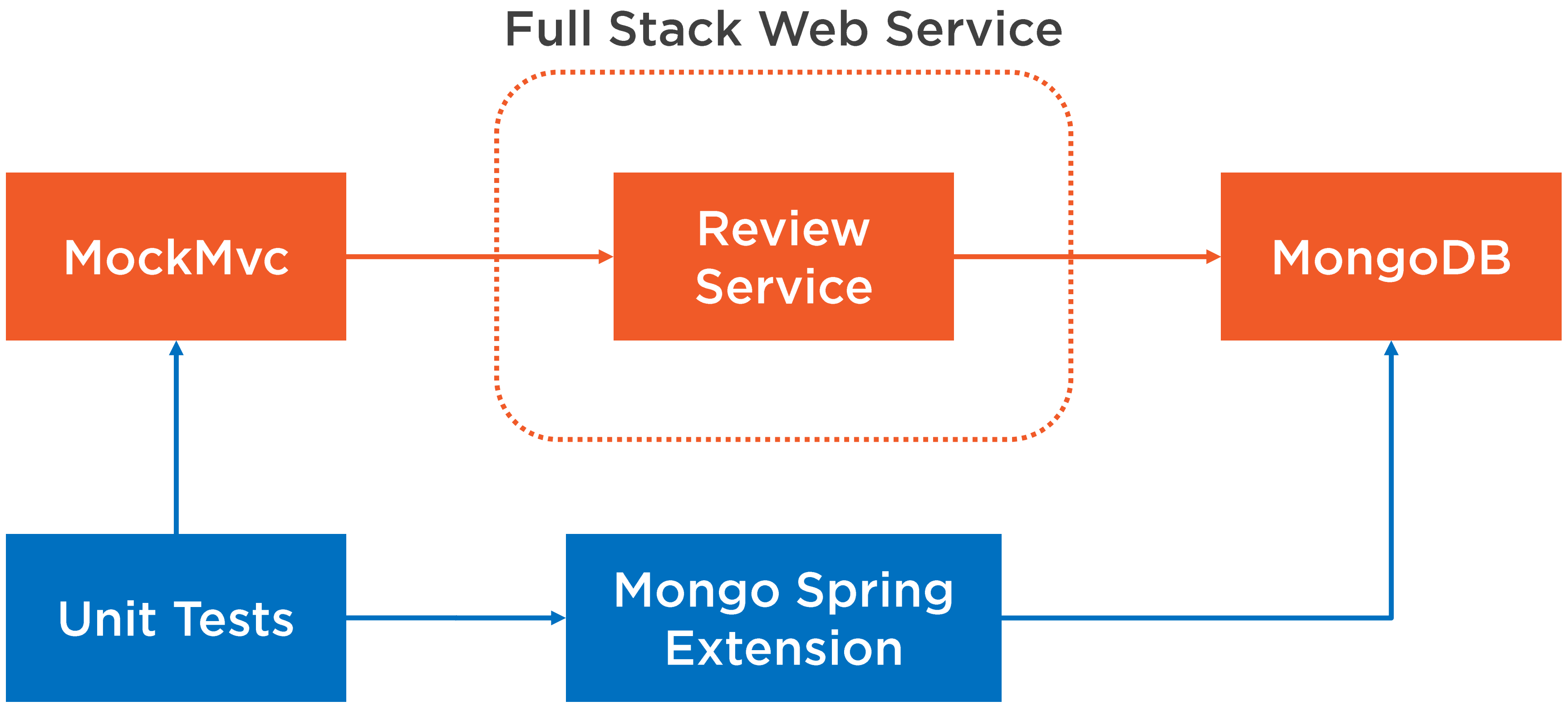
Integration Testing Strategy



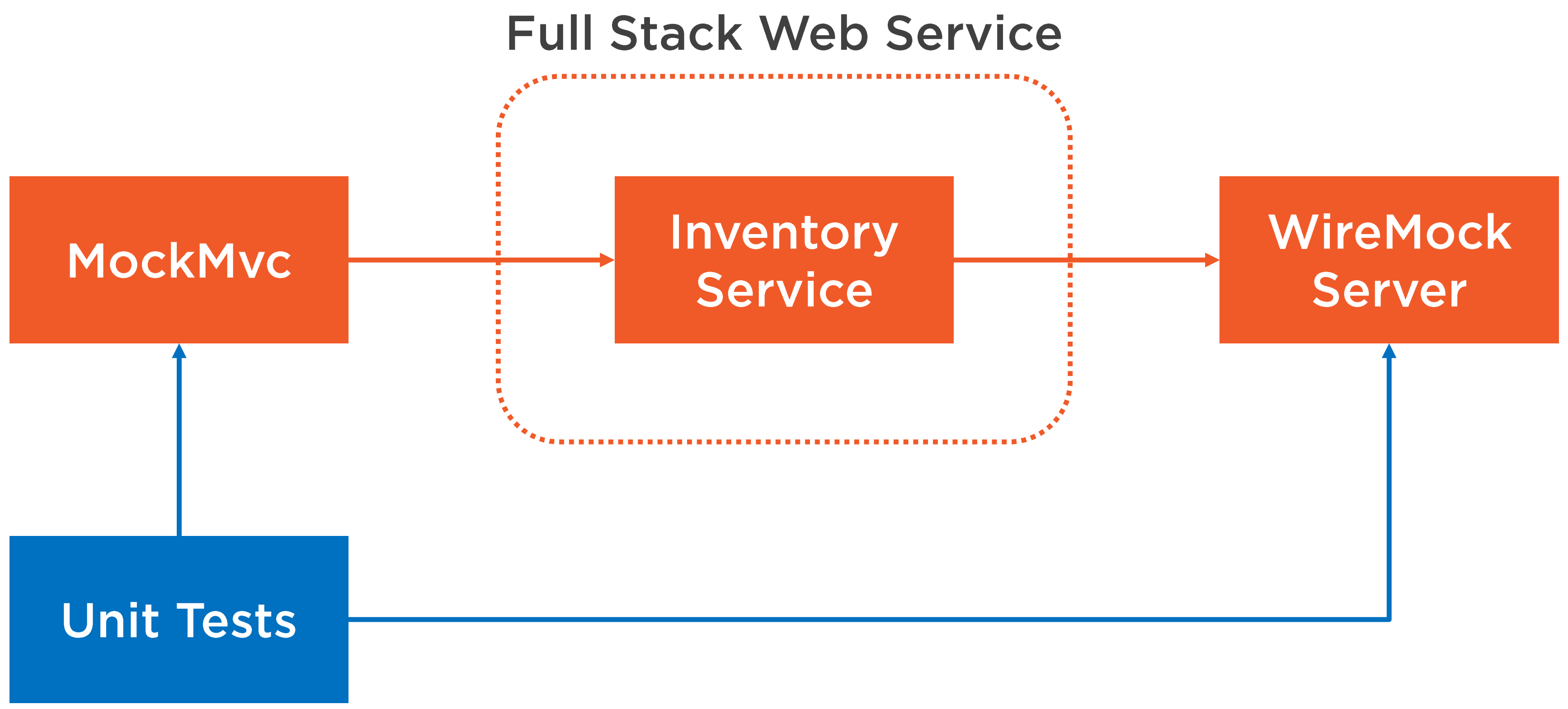
Product Service Integration Test



Review Service Integration Test



Inventory Service Integration Test



Summary

Writing Integration tests with JUnit 5

- SQL Back-end
- MongoDB Back-end
- Third-party API