Testing a Spring Service with a SQL Back-end

INTRODUCTION

Steven Haines
PRINCIPAL SOFTWARE ARCHITECT

@geekcap   www.geekcap.com

Overview

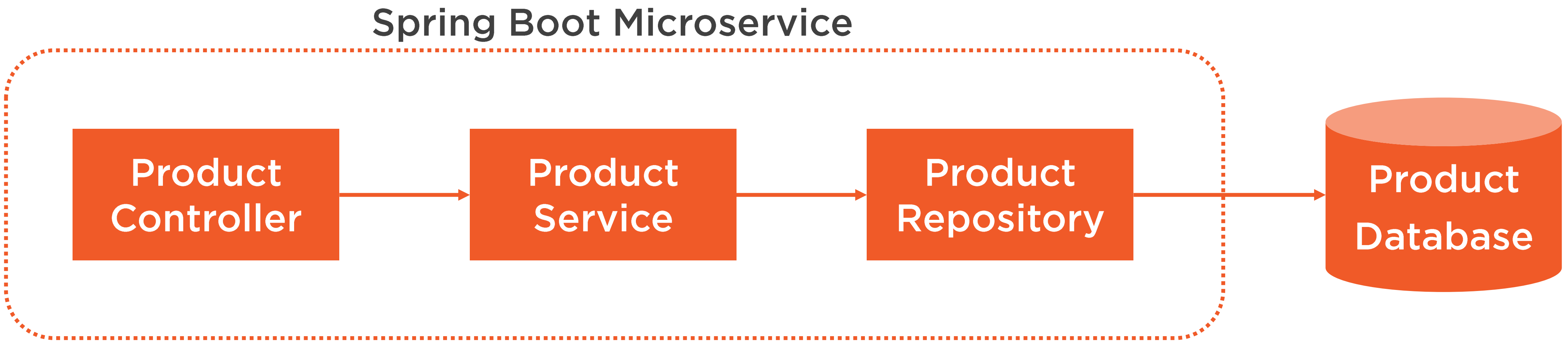
Overview of Globomatics Product Service

Testing the Controller Layer

Testing the Service Layer

Testing the Repository Layer

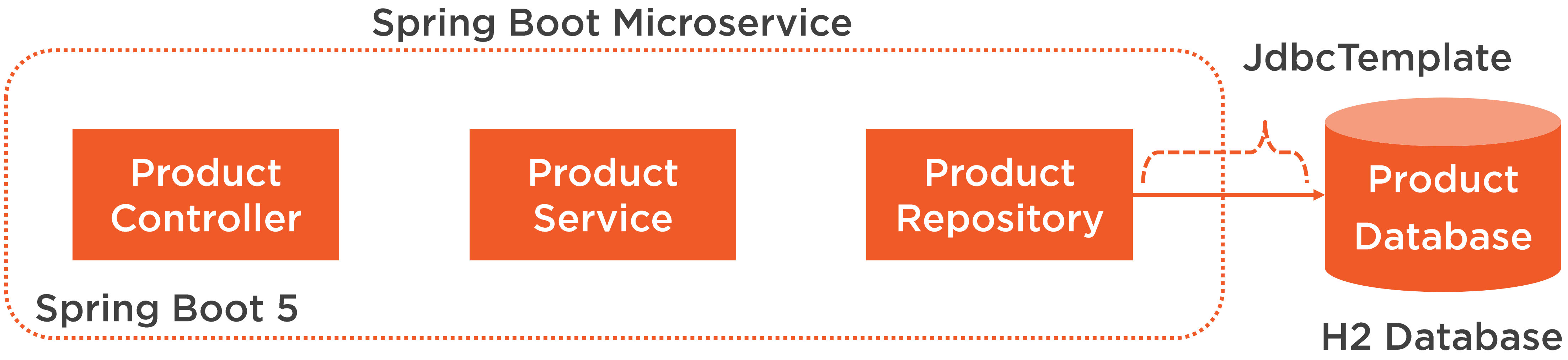
Globomatics Product Service



Product

ID	Primary Key
Name	Name of Product
Quantity	Inventory
Version	Product Version

Product Service Technology Stack



Spring Boot 5

RestController

Autowiring, Convention over Configuration

H2 Database

Embedded Database

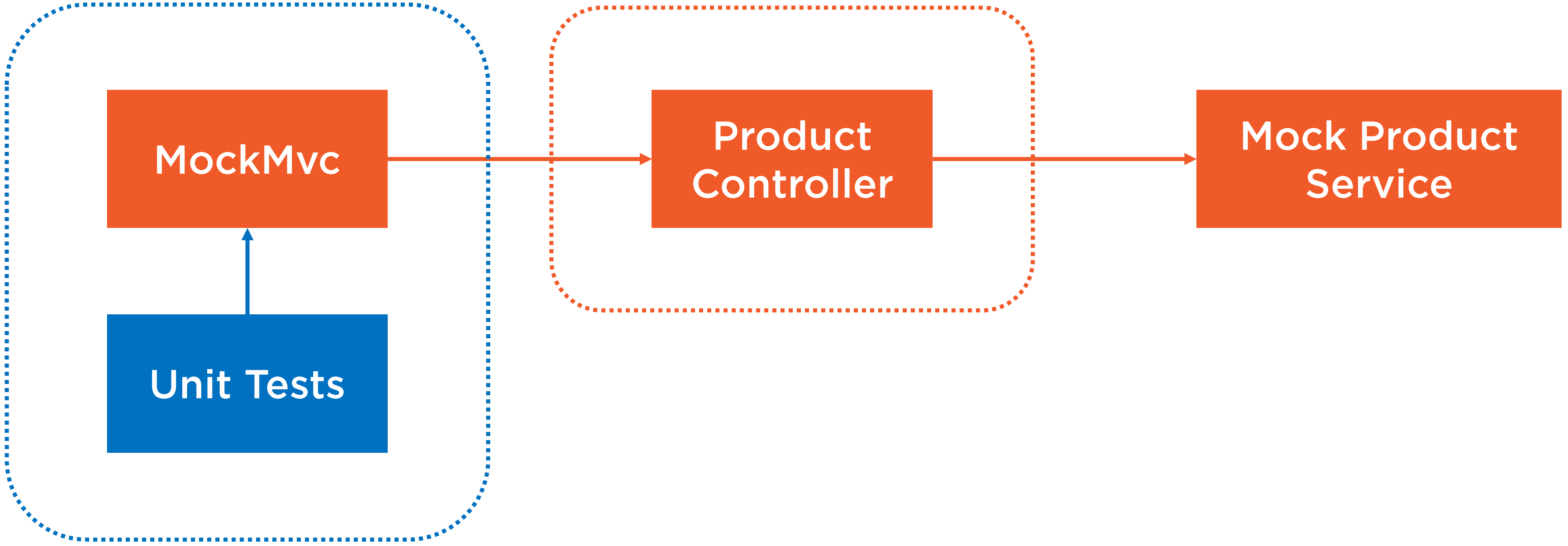
Use at runtime for simplicity

JdbcTemplate

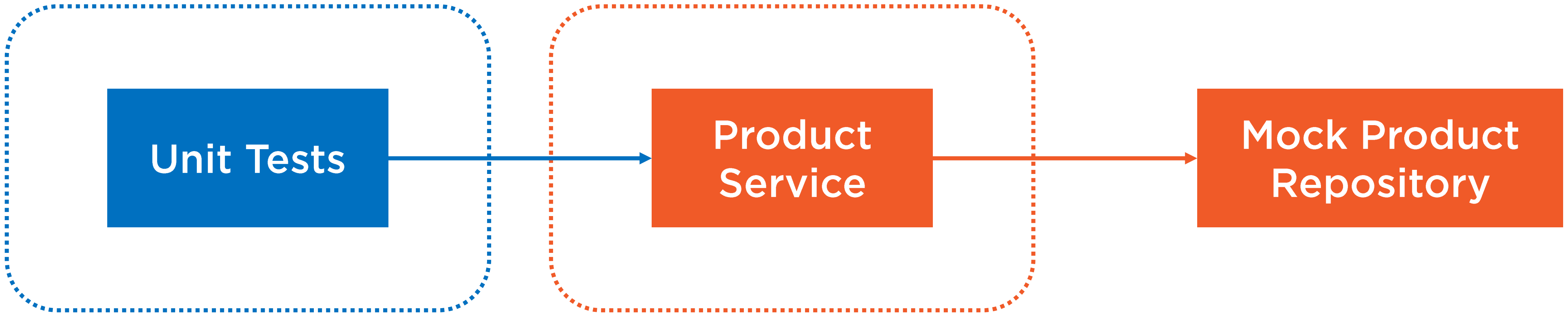
Hand-written SQL

Template Design Pattern

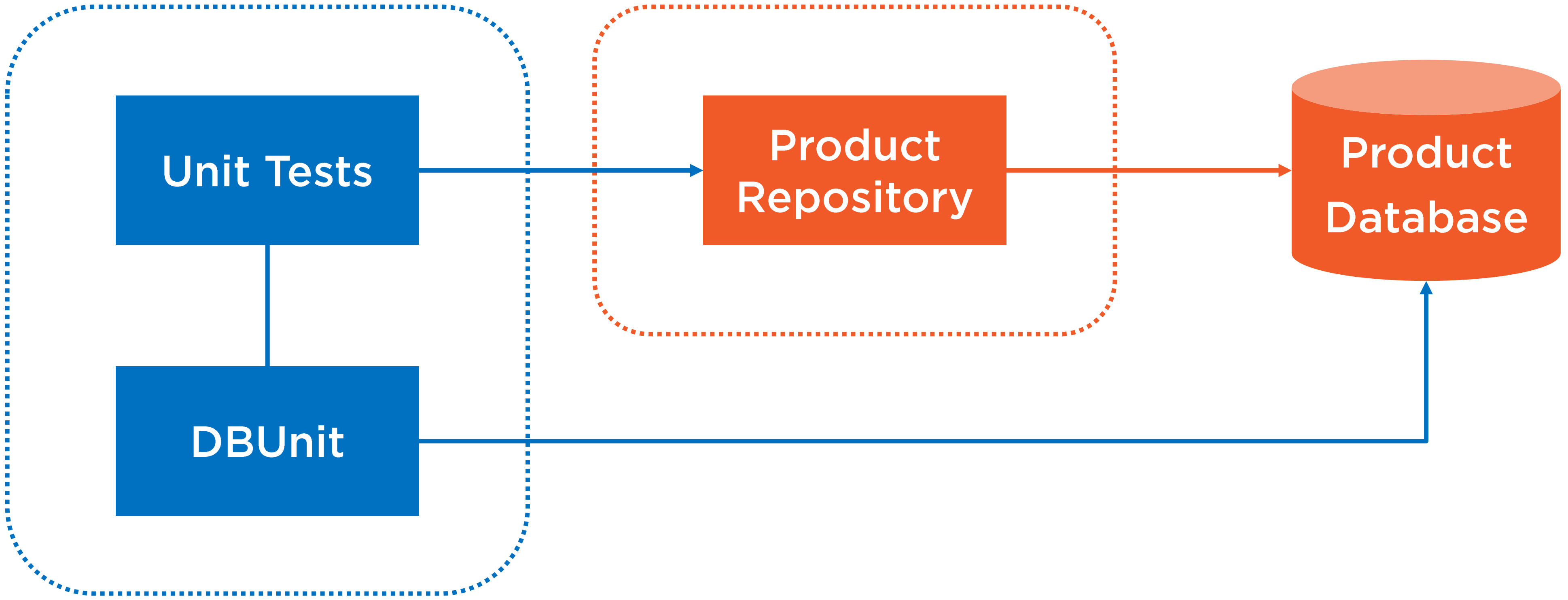
Strategy: Testing the Controller



Strategy: Testing the Service



Strategy: Testing the Repository



Testing a Spring Controller

Role of the Product Controller

Web Request

Receive a request from a web client

Product Service

Delegate to the product service for business logic

Web Response

Returns a web response: JSON and HTTP Headers

HTTP Methods

GET

Retrieve all products or a specific product

POST

Create a new product

PUT

Update an existing product

Demo

ProductController walkthrough

MockMvc

Execute Request

Validate HTTP Response

Validate HTTP Headers

Validate Response Body

```
@ExtendWith(SpringExtension.class)


@SpringBootTest


@AutoConfigureMockMvc
class ProductControllerTest {


    @MockBean
    private ProductService service;


    @Autowired
    private MockMvc mockMvc;

     ...

}
```

- SpringExtension (JUnit 5)
- Tell Spring to load the application context
- Create and configure MockMvc
- Create a mock version of the Product Service
- Autowire a MockMvc instance into the test class

Demo

ProductControllerTest walkthrough

Adding a Delete Method using TDD

Success

A valid product is deleted

Not Found

An attempt is made to delete a product that does not exist

Failure

The delete operation fails

Demo

Adding DELETE tests to the ProductControllerTest

Implementing the DELETE operation to the Product Controller

Validate that our tests pass

Testing a Spring Service

Role of a Spring Service

Business Logic

Implements business processes and logic

Abstraction Layer

Serves as an abstraction layer between the controller and back-end dependencies

Product Service Functionality

Find

Save

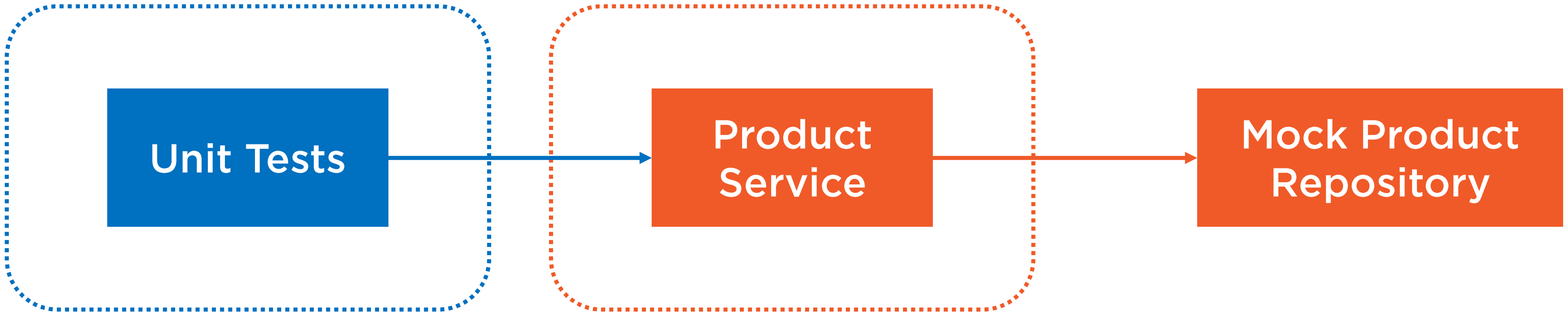
Update

Delete

Demo

ProductService walkthrough

Strategy: Testing the Service



Demo

ProductServiceTest walkthrough

Testing a Spring Repository

Spring Boot and JDBC

H2 & DataSource

Schema Creation

JdbcTemplate

Autowiring

Product Repository Functionality

Find

Save

Update

Delete

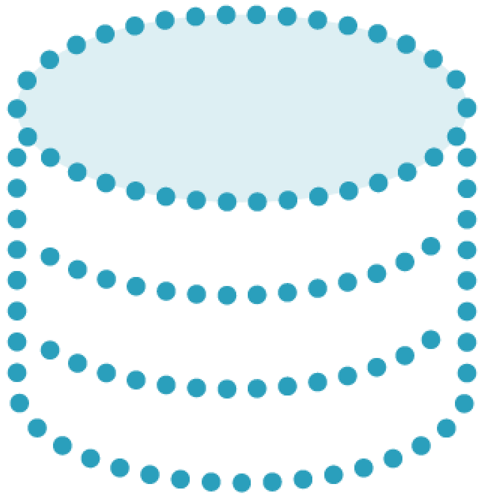
Demo

Maven POM file

Schema SQL file

ProductRepository walkthrough

Strategies for Testing Repositories



No Database

Mock DataSource

Database

Test queries against a real database

```
@Configuration

@Profile("test")

public class
ProductRepositoryTestConfiguration {
    @Primary
    @Bean
    public DataSource dataSource() {

        // Setup a data source for our tests
        DriverManagerDataSource dataSource =
               new DriverManagerDataSource();
        dataSource.setDriverClassName(
                       "org.h2.Driver");
        dataSource.setUrl(
          "jdbc:h2:mem:db;DB_CLOSE_DELAY=-1");
        dataSource.setUsername("sa");
        dataSource.setPassword("");
        return dataSource;
    }
}
```

◂ Spring Configuration Class

◂ Spring Profile: "test"

◂ Create a new DataSource Bean for testing

◂ Configure the data source

DBUnit and DBUnitExtension

Database Setup

Load test data into your database

Database Refresh

Delete all data and setup database before each unit test

products.yml

```
products:
  - id: 1
    name: "Product 1"
    quantity: 10
    version: 1
  - id: 2
    name: "Product 2"
    quantity: 5
    version: 2
```

```
/src/test/resources
         /datasets/products.yml
```

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

◀ Identifies the table (products)

◀ Create Product 1

◀ Create Product 2

◀ The YAML file is stored in your test resources

◀ Your test in annotated with @DataSet that references your YAML file

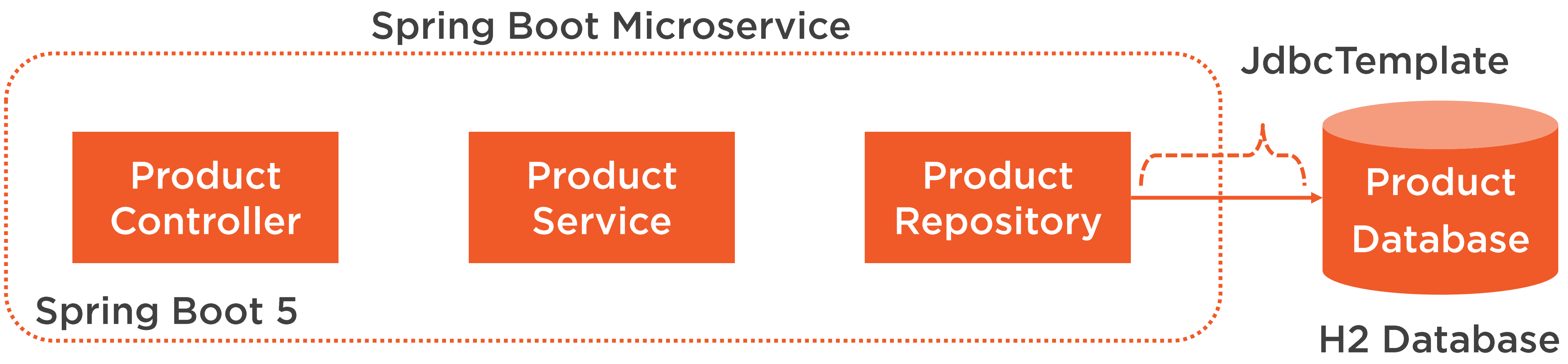
Demo

Review the setup code for our tests

Build unit tests for our repository

Summary

Product Service Technology Stack



Spring Boot 5

RestController

Autowiring, Convention over Configuration

H2 Database

Embedded Database

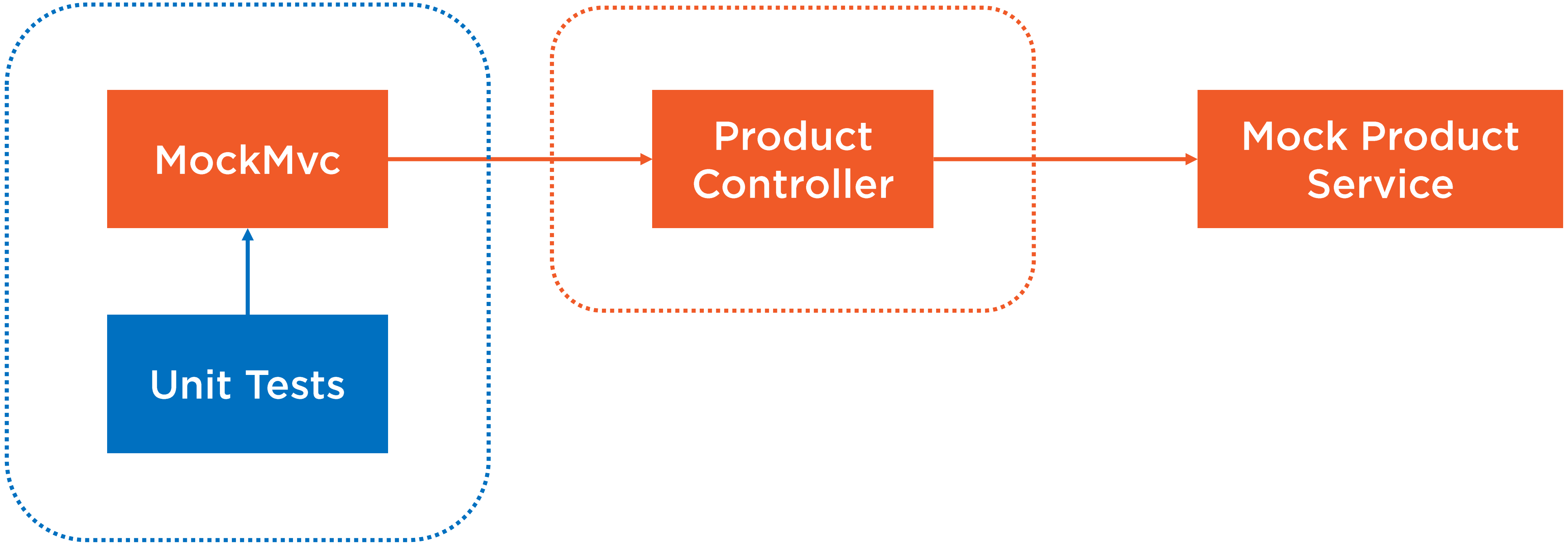
Use at runtime for simplicity

JdbcTemplate

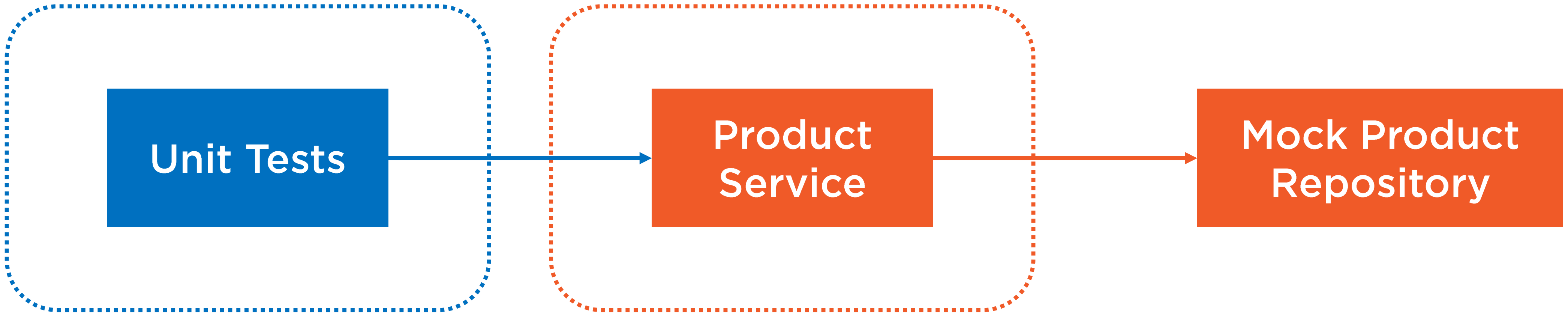
Hand-written SQL

Template Design Pattern

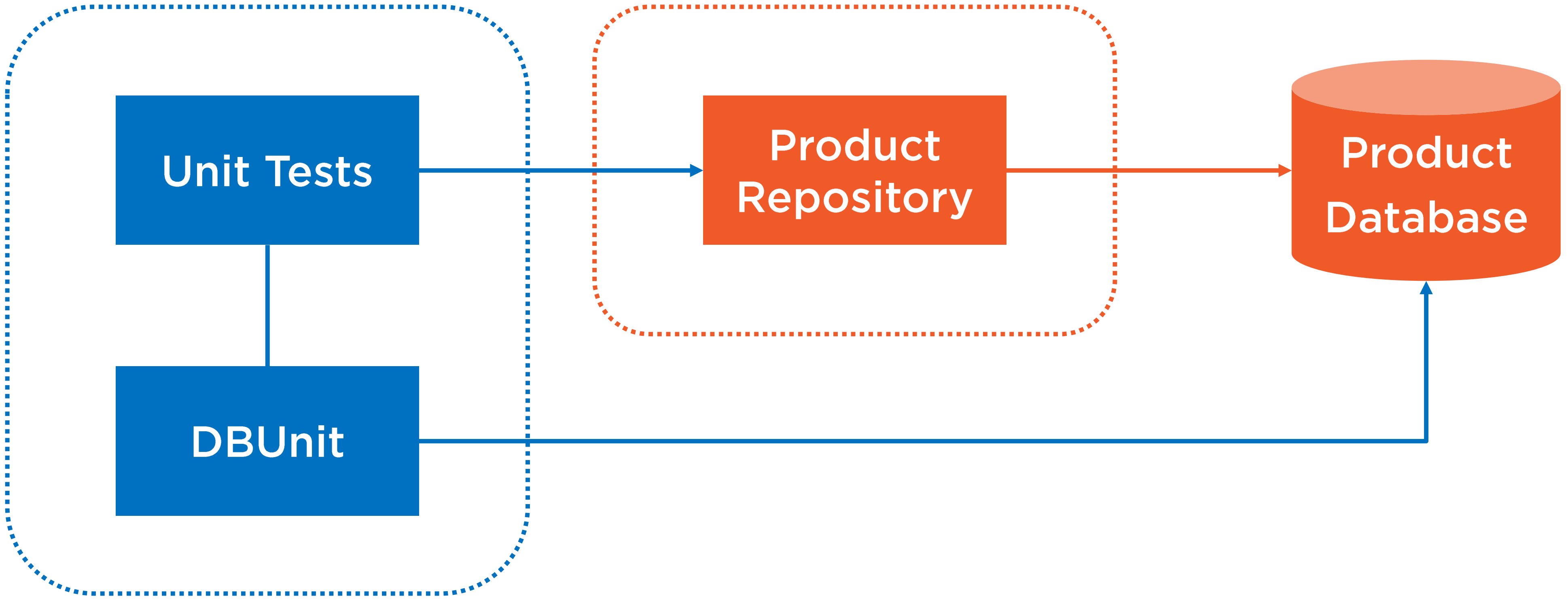
Strategy: Testing the Controller



Strategy: Testing the Service



Strategy: Testing the Repository



Summary

Testing Controllers with MockMvc

Testing Services

Testing Repositories that use a SQL database