in28minutes

Unit Testing with Spring Boot, Mockito and JUnit

Learn to write awesome unit tests with Spring Boot and Mockito in 40 easy steps!

Table of Contents

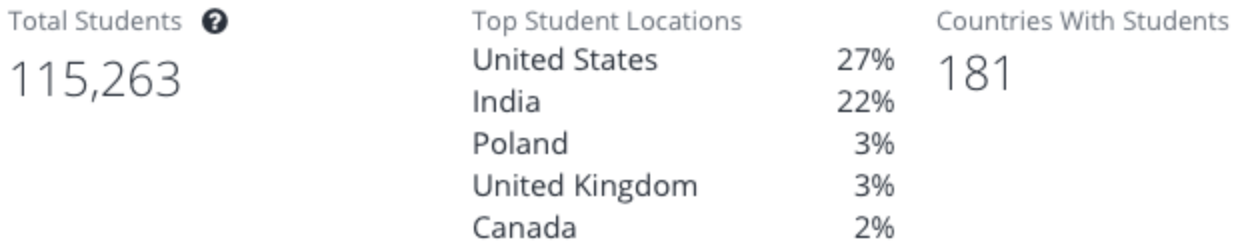
Congratulations

You have made a great choice in learning with in28Minutes. You are joining 150,000+ Learners learning everyday with us.

150,000+ Java beginners are learning from in28Minutes to become experts on APIs, Web Services and Microservices with Spring, Spring Boot and Spring Cloud.

About in28Minutes

How did in28Minutes get to 100,000 learners across the world?



We are focused on creating the awesome course (learning) experiences. Period.

An awesome learning experience?

What's that?

You need to get insight into the in28Minutes world to answer that.

You need to understand "***The in28Minutes Way***"

- What are our beliefs?
- What do we love?
- Why do we do what we do?
- How do we design our courses?

Let's get started on "***The in28Minutes Way***"!

Important Components of "The in28Minutes Way"

- Continuous Learning
- Hands-on
- We don't teach frameworks. We teach building applications!
- We want you to be strong on the fundamentals

- Step By Step

- Efficient and Effective

-
- Real Project Experiences
- Debugging and Troubleshooting skills
- Modules - Beginners and Experts!
- Focus on Unit Testing
- Code on Github

- Design and Architecture
- Modern Development Practices
- Interview Guides
- Bring the technology trends to you
- Building a connect
- Socially Conscious
- We care for our learners
- We love what we do

Troubleshooting Guide

We love all our 100,000 learners. We want to help you in every way possible.

We do not want you to get stuck because of a simple error.

This 50 page troubleshooting guide and faq is our way of thanking you for choosing to learn from in28Minutes.

.in28Minutes Trouble Shooting Guide

Getting Started

Recommended Versions

Tool/Framework/Language	Recommended Version	More Details
Java	Java 8	http://www.in28minutes.com/spr...
Eclipse	Eclipse Java EE Oxygen	Basics
Spring Boot	Spring Boot 2.0.0.RELEASE	

Installation

- Video : https://www.youtube.com/playlist?list=PLBBog2r6uMCSmMVTW_QmDLyASBvovyAO3
- PDF : https://github.com/in28minutes/SpringIn28Minutes/blob/master/InstallationGuide-JavaEclipseAndMaven_v2.pdf
- More Details : https://github.com/in28minutes/getting-started-in-5-steps

Troubleshooting

- A 50 page troubleshooting guide with more than 200 Errors and Questions answered

Course Overview

Github Repository :

https://github.com/in28minutes/spring-unit-testing-with-junit-and-mockito/

Course Overview

Title	Github	
Mocking and Spying with Mockito	Project Folder on Github	
Testing RESTful Services with Spring Boot & Mockito	Project Folder on Github	

Mocking and Spying with Mockito

Step By Step Details

- Step 01: Setting up the project using Spring Initializr
- Step 02: Writing Unit Test for a Simple Business Service
- Step 03: Setting up a Business Service to call a Data Service
- Step 04: Writing your first unit test with Stub
    - Exercise - Update Tests 2 & 3
- Step 05: Exercise Solution - Updating Tests 2 & 3 to use Stubs - Problem with Stubs.
- Step 06: Writing Unit Tests with Mocking using Mockito
    - Exercise - Updating Tests 2 & 3 to use Mockito
- Step 07: Exercise Solution - Updating Tests 2 & 3 to use Mockito
- Step 08: More Refactoring - @Mock, @InjectMocks and @RunWith(MockitoJUnitRunner.class)
- Step 09: Mockito Tips - Multiple Return Values and Specific Argument Matchers
- Step 10: Mockito Tips - Argument Matchers
- Step 11: Mockito Tips - Verify method calls
- Step 12: Mockito Tips - Argument Capture

- Step 13: Mockito Tips - Argument Capture on Multiple Calls
- Step 14: Introduction to Spy
- Step 15: Mockito FAQ

Testing RESTful Services with Spring Boot and Mockito

Step By Step Details

- Step 01: Creating a Hello World Controller
- Step 02: Using Mock Mvc to test Hello World Controller
- Step 03: Using Response Matchers to check status and content
- Step 04: Creating a Basic REST Service in Item Controller
- Step 05: Unit Testing Item Controller and Basic JSON Assertions
- Step 06: Digging deeper into JSON Assert
- Step 07: Writing a REST Service talking to Business Layer
- Step 08: Writing Unit Test for REST Service mocking Business Layer
- Step 09: Prepare Data Layers with JPA, Hibernate and H2
- Step 10: Create Item Entity and Populate data with data.sql
- Step 11: Create a RESTful Service talking to the database
- Step 12: Writing Unit Test for Web Layer - Controller - Using Mock MVC
- Step 13: Exercise & Solution - Writing Unit Test for Business Layer - Mocking
- Step 14: Writing Unit Test for Data Layer - Data JPA Test

- Step 15: Writing an Integration Test using @SpringBootTest
- - Exercise - Make Asserts Better
- Step 16: Tip : Using @MockBean to mock out dependencies you do not want to talk to!
- Step 17: Tip : Creating Different Test Configuration
- Step 18: Writing Unit Tests for Other Request Methods
- Step 19: Refactor SomeBusinessImpl to use Functional Programming
    - Exercise - Convert the second method to use Functional Approach
- Step 20: Better Assertions with Hamcrest - HamcrestMatcherTest
- Step 21: Better Assertions with AssertJ - AssertJTest
- Step 22: Better Assertions with JSONPath - JSONPathTest
- Step 23: Easier Static Imports
- Step 24: Tip : Measuring Test Coverage with Eclipse
- Step 25: Tip : Keep an eye on performance of unit tests!
- Step 26: Good Unit Tests

Complete Code

Complete Code Example

/src/main/java/com/in28minutes/unittesting/unittesting/business/ItemBusinessService.java

```
package com.in28minutes.unittesting.unittesting.business;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import
com.in28minutes.unittesting.unittesting.data.ItemRepository
;
import com.in28minutes.unittesting.unittesting.model.Item;

@Component
public class ItemBusinessService {

	@Autowired
	private ItemRepository repository;

	public Item retreiveHardcodedItem() {
		return new Item(1, "Ball", 10, 100);
	}

	public List<Item> retrieveAllItems()
```

```
 {
                List<Item> items = repository.findAll();

                for(Item item:items) {
                        item.setValue(item.getPrice() *
item.getQuantity());
                }

                return items;
        }

}
```

/src/main/java/com/in28minutes/unittesting/unittesting/business/SomeBusinessImpl.java

```
package com.in28minutes.unittesting.unittesting.business;

import
com.in28minutes.unittesting.unittesting.data.SomeDataServic
e;

public class SomeBusinessImpl {

        private SomeDataService someDataService;

        public void setSomeDataService(SomeDataService
someDataService) {
                this.someDataService = someDataService;
        }

        public int calculateSum(int[] data) {
                int sum = 0;
                for(int value:data) {
                        sum += value;
```

```
        }
                return sum;
                //Functional Style
                //return
Arrays.stream(data).reduce(Integer::sum).orElse(0);
        }

        public int calculateSumUsingDataService() {
                int sum = 0;
                int[] data =
someDataService.retrieveAllData();
                for(int value:data) {
                        sum += value;
                }

                //someDataService.storeSum(sum);
                return sum;
                //Functional Style
                //return
Arrays.stream(data).reduce(Integer::sum).orElse(0);
        }

}
```

/src/main/java/com/in28minutes/unittesting/unittesting/controller/HelloWorldController.java

```
package com.in28minutes.unittesting.unittesting.controller;

import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloWorldController {

```

```
        @GetMapping("/hello-world")
        public String helloWorld() {
                return "Hello World";
        }

}
```

/src/main/java/com/in28minutes/unittesting/unittesting/controller/ItemController.java

```
package com.in28minutes.unittesting.unittesting.controller;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RestController;

import
com.in28minutes.unittesting.unittesting.business.ItemBusine
ssService;
import com.in28minutes.unittesting.unittesting.model.Item;

@RestController
public class ItemController {

        @Autowired
        private ItemBusinessService businessService;

        @GetMapping("/dummy-item")
        public Item dummyItem() {
                return new Item(1, "Ball", 10, 100);
        }
```

```
        @GetMapping("/item-from-business-service")
        public Item itemFromBusinessService() {
                Item item =
businessService.retreiveHardcodedItem();

                return item;
        }

        @GetMapping("/all-items-from-database")
        public List<Item> retrieveAllItems() {
                return businessService.retrieveAllItems();
        }

}
```

/src/main/java/com/in28minutes/unittesting/unittesting/data/ItemRepository.java

```
package com.in28minutes.unittesting.unittesting.data;

import
org.springframework.data.jpa.repository.JpaRepository;

import com.in28minutes.unittesting.unittesting.model.Item;

public interface ItemRepository extends JpaRepository<Item,
Integer>{

}
```

/src/main/java/com/in28minutes/unittesting/unittesting/data/SomeDataService.java

```
package com.in28minutes.unittesting.unittesting.data;
```

```
public interface SomeDataService {

	int[] retrieveAllData();

	//int retrieveSpecificData();

}
```

/src/main/java/com/in28minutes/unittesting/unittesting/model/Item.java

```
package com.in28minutes.unittesting.unittesting.model;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Transient;

@Entity
public class Item {

	@Id
	private int id;
	private String name;
	private int price;
	private int quantity;

	@Transient
	private int value;

	protected Item() {

	}

	public Item(int id, String name, int price, int
quantity)
```

```
 {
                this.id = id;
                this.name = name;
                this.price = price;
                this.quantity = quantity;
        }

        public int getId() {
                return id;
        }

        public String getName() {
                return name;
        }

        public int getPrice() {
                return price;
        }

        public int getQuantity() {
                return quantity;
        }

        public int getValue() {
                return value;
        }

        public void setValue(int value) {
                this.value = value;
        }

        public String toString() {
                return String.format("Item[%d, %s, %d,
%d]", id, name, price, quantity);
        }
}
```

/src/main/java/com/in28minutes/unittesting/unittesting/UnitTestingApplication.java

```
package com.in28minutes.unittesting.unittesting;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplicatio
n;

@SpringBootApplication
public class UnitTestingApplication {

        public static void main(String[] args) {

SpringApplication.run(UnitTestingApplication.class, args);
        }
}
```

/src/main/resources/application.properties

```
spring.jpa.show-sql=true
spring.h2.console.enabled=true
```

/src/main/resources/data.sql

```
insert into item(id, name, price, quantity)
values(10001,'Item1',10,20);
insert into item(id, name, price, quantity)
values(10002,'Item2',5,10);
insert into item(id, name, price, quantity)
values(10003,'Item3',15,2);
```

/src/test/java/com/in28minutes/unittesting/unittesting/business/ItemBusinessServiceTest.java

```
package com.in28minutes.unittesting.unittesting.business;

import static org.junit.Assert.assertEquals;
import static org.mockito.Mockito.when;

import java.util.Arrays;
import java.util.List;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.junit.MockitoJUnitRunner;

import
com.in28minutes.unittesting.unittesting.data.ItemRepository
;
import com.in28minutes.unittesting.unittesting.model.Item;

@RunWith(MockitoJUnitRunner.class)
public class ItemBusinessServiceTest {

        @InjectMocks
        private ItemBusinessService business;

        @Mock
        private ItemRepository repository;

        @Test
        public void retrieveAllItems_basic() {

when(repository.findAll()).thenReturn(Arrays.asList(new
Item(2,"Item2",10,10),
                                new
```

```
 Item(3,"Item3",20,20)));
                List<Item> items =
business.retrieveAllItems();

                assertEquals(100, items.get(0).getValue());
                assertEquals(400, items.get(1).getValue());
        }
}
```

/src/test/java/com/in28minutes/unittesting/unittesting/business/ListMockTest.java

```
package com.in28minutes.unittesting.unittesting.business;

import static org.junit.Assert.assertEquals;
import static org.mockito.ArgumentMatchers.anyInt;
import static org.mockito.Mockito.atLeast;
import static org.mockito.Mockito.atLeastOnce;
import static org.mockito.Mockito.atMost;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.never;
import static org.mockito.Mockito.spy;
import static org.mockito.Mockito.times;
import static org.mockito.Mockito.verify;
import static org.mockito.Mockito.when;

import java.util.ArrayList;
import java.util.List;

import org.junit.Ignore;
import org.junit.Test;
import org.mockito.ArgumentCaptor;

public class ListMockTest {

```

```
        List<String> mock = mock(List.class);

        @Test
        public void size_basic() {
                when(mock.size()).thenReturn(5);
                assertEquals(5, mock.size());
        }

        @Test
        public void returnDifferentValues() {

when(mock.size()).thenReturn(5).thenReturn(10);
                assertEquals(5, mock.size());
                assertEquals(10, mock.size());
        }

        @Test
        @Ignore
        public void returnWithParameters() {

when(mock.get(0)).thenReturn("in28Minutes");
                assertEquals("in28Minutes", mock.get(0));
                assertEquals(null, mock.get(1));
        }

        @Test
        public void returnWithGenericParameters() {

when(mock.get(anyInt())).thenReturn("in28Minutes");

                assertEquals("in28Minutes", mock.get(0));
                assertEquals("in28Minutes", mock.get(1));
        }

        @Test
        public void verificationBasics()
```

```
 {
                // SUT
                String value1 = mock.get(0);
                String value2 = mock.get(1);

                // Verify
                verify(mock).get(0);
                verify(mock, times(2)).get(anyInt());
                verify(mock, atLeast(1)).get(anyInt());
                verify(mock, atLeastOnce()).get(anyInt());
                verify(mock, atMost(2)).get(anyInt());
                verify(mock, never()).get(2);
        }

        @Test
        public void argumentCapturing() {

                //SUT
                mock.add("SomeString");

                //Verification
                ArgumentCaptor<String> captor =
ArgumentCaptor.forClass(String.class);
                verify(mock).add(captor.capture());

                assertEquals("SomeString",
captor.getValue());

        }

        @Test
        public void multipleArgumentCapturing() {

                //SUT
                mock.add("SomeString1");
```

```
        mock.add("SomeString2");

                //Verification
                ArgumentCaptor<String> captor =
ArgumentCaptor.forClass(String.class);

                verify(mock,
times(2)).add(captor.capture());

                List<String> allValues =
captor.getAllValues();

                assertEquals("SomeString1",
allValues.get(0));
                assertEquals("SomeString2",
allValues.get(1));

        }

        @Test
        public void mocking() {
                ArrayList arrayListMock =
mock(ArrayList.class);

System.out.println(arrayListMock.get(0));//null

System.out.println(arrayListMock.size());//0
                arrayListMock.add("Test");
                arrayListMock.add("Test2");

System.out.println(arrayListMock.size());//0
                when(arrayListMock.size()).thenReturn(5);

System.out.println(arrayListMock.size());//5
        }
```

```
        @Test
        public void spying() {
                ArrayList arrayListSpy =
spy(ArrayList.class);
                arrayListSpy.add("Test0");

System.out.println(arrayListSpy.get(0));//Test0
                System.out.println(arrayListSpy.size());//1
                arrayListSpy.add("Test");
                arrayListSpy.add("Test2");
                System.out.println(arrayListSpy.size());//3

                when(arrayListSpy.size()).thenReturn(5);
                System.out.println(arrayListSpy.size());//5

                arrayListSpy.add("Test4");
                System.out.println(arrayListSpy.size());//5

                verify(arrayListSpy).add("Test4");
        }


}
```

/src/test/java/com/in28minutes/unittesting/unittesting/business/SomeBusinessMockTest.java

```
package com.in28minutes.unittesting.unittesting.business;

import static org.junit.Assert.assertEquals;
import static org.mockito.Mockito.when;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.InjectMocks;
```

```
import org.mockito.Mock;
import org.mockito.junit.MockitoJUnitRunner;

import
com.in28minutes.unittesting.unittesting.data.SomeDataServic
e;

@RunWith(MockitoJUnitRunner.class)
public class SomeBusinessMockTest {

        @InjectMocks
        SomeBusinessImpl business;

        @Mock
        SomeDataService dataServiceMock;

        @Test
        public void calculateSumUsingDataService_basic() {

when(dataServiceMock.retrieveAllData()).thenReturn(new
int[] { 1, 2, 3 });
                assertEquals(6,
business.calculateSumUsingDataService());
        }

        @Test
        public void calculateSumUsingDataService_empty() {

when(dataServiceMock.retrieveAllData()).thenReturn(new
int[] {});
                assertEquals(0,
business.calculateSumUsingDataService());
        }

        @Test
        public void calculateSumUsingDataService_oneValue()
{
```

```
when(dataServiceMock.retrieveAllData()).thenReturn(new
int[] { 5
```

```
 });
                assertEquals(5,
business.calculateSumUsingDataService());
        }
}
```

/src/test/java/com/in28minutes/unittesting/unittesting/business/SomeBusinessStubTest.java

```
package com.in28minutes.unittesting.unittesting.business;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

import
com.in28minutes.unittesting.unittesting.data.SomeDataServic
e;

class SomeDataServiceStub implements SomeDataService {
        @Override
        public int[] retrieveAllData() {
                return new int[] { 1, 2, 3 };
        }
}

class SomeDataServiceEmptyStub implements SomeDataService {
        @Override
        public int[] retrieveAllData() {
                return new int[] { };
        }
}

class SomeDataServiceOneElementStub implements
SomeDataService {
        @Override
```

```
        public int[] retrieveAllData() {
                return new int[] { 5 };
        }
}

public class SomeBusinessStubTest {

        @Test
        public void calculateSumUsingDataService_basic() {
                SomeBusinessImpl business = new
SomeBusinessImpl();
                business.setSomeDataService(new
SomeDataServiceStub());
                int actualResult =
business.calculateSumUsingDataService();
                int expectedResult = 6;
                assertEquals(expectedResult, actualResult);
        }

        @Test
        public void calculateSumUsingDataService_empty() {
                SomeBusinessImpl business = new
SomeBusinessImpl();
                business.setSomeDataService(new
SomeDataServiceEmptyStub());
                int actualResult =
business.calculateSumUsingDataService();//new int[] {}
                int expectedResult = 0;
                assertEquals(expectedResult, actualResult);
        }

        @Test
        public void calculateSumUsingDataService_oneValue()
{
                SomeBusinessImpl business = new
```

```
 SomeBusinessImpl();
                business.setSomeDataService(new
SomeDataServiceOneElementStub());
                int actualResult =
business.calculateSumUsingDataService();//new int[] { 5 }
                int expectedResult = 5;
                assertEquals(expectedResult, actualResult);
        }
}
```

/src/test/java/com/in28minutes/unittesting/unittesting/business/SomeBusinessTest.java

```
package com.in28minutes.unittesting.unittesting.business;

import static org.junit.Assert.*;

import org.junit.Test;

public class SomeBusinessTest {

        @Test
        public void calculateSum_basic() {
                SomeBusinessImpl business = new
SomeBusinessImpl();
                int actualResult =
business.calculateSum(new int[] { 1,2,3});
                int expectedResult = 6;
                assertEquals(expectedResult, actualResult);
        }

        @Test
        public void calculateSum_empty() {
                SomeBusinessImpl business = new
```

```
 SomeBusinessImpl();
                int actualResult =
business.calculateSum(new int[] { });
                int expectedResult = 0;
                assertEquals(expectedResult, actualResult);
        }

        @Test
        public void calculateSum_oneValue() {
                SomeBusinessImpl business = new
SomeBusinessImpl();
                int actualResult =
business.calculateSum(new int[] { 5});
                int expectedResult = 5;
                assertEquals(expectedResult, actualResult);
        }
}
```

/src/test/java/com/in28minutes/unittesting/unittesting/controller/HelloWorldControllerTest.java

```
package com.in28minutes.unittesting.unittesting.controller;

import static
org.springframework.test.web.servlet.result.MockMvcResultMatchers.content;
import static
org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

import org.junit.Test;
import org.junit.runner.RunWith;
import
org.springframework.beans.factory.annotation.Autowired;
import
```

```
 org.springframework.boot.test.autoconfigure.web.servlet.We
bMvcTest;
import org.springframework.http.MediaType;
import
org.springframework.test.context.junit4.SpringRunner;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.MvcResult;
import org.springframework.test.web.servlet.RequestBuilder;
import
org.springframework.test.web.servlet.request.MockMvcRequest
Builders;

@RunWith(SpringRunner.class)
@WebMvcTest(HelloWorldController.class)
public class HelloWorldControllerTest {

        @Autowired
        private MockMvc mockMvc;

        @Test
        public void helloWorld_basic() throws Exception {
                //call GET "/hello-world"  application/json

                RequestBuilder request =
MockMvcRequestBuilders
                                .get("/hello-world")

.accept(MediaType.APPLICATION_JSON);

                MvcResult result = mockMvc.perform(request)
                                .andExpect(status().isOk())

.andExpect(content().string("Hello World"))
                                .andReturn();

                //verify "Hello World"
                //assertEquals("Hello World",
```

```
 result.getResponse().getContentAsString());
        }

}
```

/src/test/java/com/in28minutes/unittesting/unittesting/controller/ItemControllerIT.java

```
package com.in28minutes.unittesting.unittesting.controller;

import org.json.JSONException;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.skyscreamer.jsonassert.JSONAssert;
import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.boot.test.context.SpringBootTest;
import
org.springframework.boot.test.context.SpringBootTest.WebEnv
ironment;
import
org.springframework.boot.test.web.client.TestRestTemplate;
import
org.springframework.test.context.junit4.SpringRunner;

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

        @Autowired
        private TestRestTemplate restTemplate;

        @Test
        public void contextLoads() throws JSONException
```

```
 {

                String response =
this.restTemplate.getForObject("/all-items-from-database",
String.class);

                JSONAssert.assertEquals("[{id:10001},
{id:10002},{id:10003}]",
                                        response, false);
        }

}
```

/src/test/java/com/in28minutes/unittesting/unittesting/controller/ItemControllerTest.java

```
package com.in28minutes.unittesting.unittesting.controller;

import static org.mockito.Mockito.when;
import static
org.springframework.test.web.servlet.result.MockMvcResultMa
tchers.content;
import static
org.springframework.test.web.servlet.result.MockMvcResultMa
tchers.status;

import java.util.Arrays;

import org.junit.Test;
import org.junit.runner.RunWith;
import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.boot.test.autoconfigure.web.servlet.Web
MvcTest;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.http.MediaType;
```

```
import
org.springframework.test.context.junit4.SpringRunner;
```

```
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.MvcResult;
import org.springframework.test.web.servlet.RequestBuilder;
import
org.springframework.test.web.servlet.request.MockMvcRequest
Builders;

import
com.in28minutes.unittesting.unittesting.business.ItemBusine
ssService;
import com.in28minutes.unittesting.unittesting.model.Item;

@RunWith(SpringRunner.class)
@WebMvcTest(ItemController.class)
public class ItemControllerTest {

        @Autowired
        private MockMvc mockMvc;

        @MockBean
        private ItemBusinessService businessService;

        @Test
        public void dummyItem_basic() throws Exception {

                RequestBuilder request =
MockMvcRequestBuilders
                                .get("/dummy-item")

.accept(MediaType.APPLICATION_JSON);

                MvcResult result = mockMvc.perform(request)
                                .andExpect(status().isOk())
                                .andExpect(content().json("
{\"id\":
1,\"name\":\"Ball\",\"price\":10,\"quantity\":100}"))
                                .andReturn();
                //JSONAssert.assertEquals(expected,
```

```
 result.getResponse().getContentAsString(), false);

        }

        @Test
        public void itemFromBusinessService_basic() throws
Exception {

when(businessService.retreiveHardcodedItem()).thenReturn(
                                new Item(2,"Item2",10,10));

                RequestBuilder request =
MockMvcRequestBuilders
                                .get("/item-from-business-
service")

.accept(MediaType.APPLICATION_JSON);

                MvcResult result = mockMvc.perform(request)
                                .andExpect(status().isOk())
                                .andExpect(content().json("
{id:2,name:Item2,price:10}"))
                                .andReturn();
                //JSONAssert.assertEquals(expected,
result.getResponse().getContentAsString(), false);

        }

        @Test
        public void retrieveAllItems_basic() throws
Exception {

when(businessService.retrieveAllItems()).thenReturn(
                                Arrays.asList(new
Item(2,"Item2",10,10),
                                                new
```

```
 Item(3,"Item3",20,20))
                                );

                RequestBuilder request =
MockMvcRequestBuilders
                                .get("/all-items-from-
database")

.accept(MediaType.APPLICATION_JSON);

                MvcResult result = mockMvc.perform(request)
                                .andExpect(status().isOk())
                                .andExpect(content().json("
[{id:3,name:Item3,price:20}, {id:2,name:Item2,price:10}]"))
                                .andReturn();
                //JSONAssert.assertEquals(expected,
result.getResponse().getContentAsString(), false);

        }

        @Test
        public void retrieveAllItems_noitems() throws
Exception {

when(businessService.retrieveAllItems()).thenReturn(
                                Arrays.asList()
                                );

                RequestBuilder request =
MockMvcRequestBuilders
                                .get("/all-items-from-
database")

.accept(MediaType.APPLICATION_JSON);

                MvcResult result =
```

```
 mockMvc.perform(request)
                                    .andExpect(status().isOk())
                                    .andExpect(content().json("
[]"))
                                    .andReturn();
                //JSONAssert.assertEquals(expected,
result.getResponse().getContentAsString(), false);

        }

}
```

/src/test/java/com/in28minutes/unittesting/unittesting/data/ItemRepositoryTest.java

```
package com.in28minutes.unittesting.unittesting.data;

import static org.junit.Assert.assertEquals;

import java.util.List;
import java.util.Optional;

import org.junit.Test;
import org.junit.runner.RunWith;
import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.boot.test.autoconfigure.orm.jpa.DataJpa
Test;
import
org.springframework.test.context.junit4.SpringRunner;

import com.in28minutes.unittesting.unittesting.model.Item;

@RunWith(SpringRunner.class)
@DataJpaTest
```

```
public class ItemRepositoryTest {

	@Autowired
	private ItemRepository repository;

	@Test
	public void testFindAll() {
		List<Item> items = repository.findAll();
		assertEquals(3,items.size());
	}

	@Test
	public void testFindOne() {
		Item item =
repository.findById(10001).get();

		assertEquals("Item1",item.getName());
	}

}
```

/src/test/java/com/in28minutes/unittesting/unittesting/spike/AssertJTest.java

```
package com.in28minutes.unittesting.unittesting.spike;

import static org.assertj.core.api.Assertions.assertThat;

import java.util.Arrays;
import java.util.List;

import org.junit.Test;

public class AssertJTest {

```

```
        @Test
        public void learning() {
                List<Integer> numbers =
Arrays.asList(12,15,45);

                //assertThat(numbers, hasSize(3));
                assertThat(numbers).hasSize(3)

.contains(12,15)
                                                .allMatch(x
-> x > 10)
                                                .allMatch(x
-> x < 100)

.noneMatch(x -> x < 0);

                assertThat("").isEmpty();
                assertThat("ABCDE").contains("BCD")

.startsWith("ABC")

.endsWith("CDE");
        }

}
```

/src/test/java/com/in28minutes/unittesting/unittesting/spike/HamcrestMatchersTest.java

```
package com.in28minutes.unittesting.unittesting.spike;

import static org.hamcrest.CoreMatchers.everyItem;
import static org.hamcrest.CoreMatchers.hasItems;
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.containsString;
```

```
import static org.hamcrest.Matchers.endsWith;
import static org.hamcrest.Matchers.greaterThan;
import static org.hamcrest.Matchers.hasSize;
import static org.hamcrest.Matchers.isEmptyString;
import static org.hamcrest.Matchers.lessThan;
import static org.hamcrest.Matchers.startsWith;

import java.util.Arrays;
import java.util.List;

import org.junit.Test;

public class HamcrestMatchersTest {

	@Test
	public void learning() {
		List<Integer> numbers =
Arrays.asList(12,15,45);

		assertThat(numbers, hasSize(3));
		assertThat(numbers, hasItems(12,45));
		assertThat(numbers,
everyItem(greaterThan(10)));
		assertThat(numbers,
everyItem(lessThan(100)));

		assertThat("", isEmptyString());
		assertThat("ABCDE", containsString("BCD"));
		assertThat("ABCDE", startsWith("ABC"));
		assertThat("ABCDE", endsWith("CDE"));

	}

}
```

/src/test/java/com/in28minutes/unittesting/unittesting/spike/JsonAssertTest.java

```
package com.in28minutes.unittesting.unittesting.spike;

import org.json.JSONException;
import org.junit.Test;
import org.skyscreamer.jsonassert.JSONAssert;

public class JsonAssertTest {

	String actualResponse = "
{\"id\":1,\"name\":\"Ball\",\"price\":10,\"quantity\":100}"
;

	@Test
	public void
jsonAssert_StrictTrue_ExactMatchExceptForSpaces() throws
JSONException {
		String expectedResponse = "{\"id\": 1,
\"name\":\"Ball\", \"price\":10, \"quantity\":100}";
		JSONAssert.assertEquals(expectedResponse,
actualResponse, true);
	}

	@Test
	public void jsonAssert_StrictFalse() throws
JSONException {
		String expectedResponse = "{\"id\": 1,
\"name\":\"Ball\", \"price\":10}";
		JSONAssert.assertEquals(expectedResponse,
actualResponse, false);
	}

	@Test
	public void jsonAssert_WithoutEscapeCharacters()
throws JSONException
```

```
 {
                String expectedResponse = "{id:1,
name:Ball, price:10}";
                JSONAssert.assertEquals(expectedResponse,
actualResponse, false);
        }
}
```

/src/test/java/com/in28minutes/unittesting/unittesting/spike/JsonPathTest.java

```
package com.in28minutes.unittesting.unittesting.spike;

import static org.assertj.core.api.Assertions.assertThat;

import java.util.List;

import org.junit.Test;

import com.jayway.jsonpath.DocumentContext;
import com.jayway.jsonpath.JsonPath;

public class JsonPathTest {

        @Test
        public void learning() {
                String responseFromService = "[" +
                                "{\"id\":10000,
\"name\":\"Pencil\", \"quantity\":5}," +
                                "{\"id\":10001,
\"name\":\"Pen\", \"quantity\":15}," +
                                "{\"id\":10002,
\"name\":\"Eraser\", \"quantity\":10}" +
```

```
        "]";

                DocumentContext context =
JsonPath.parse(responseFromService);

                int length = context.read("$.length()");
                assertThat(length).isEqualTo(3);

                List<Integer> ids = context.read("$..id");


assertThat(ids).containsExactly(10000,10001,10002);

                System.out.println(context.read("$.
[1]").toString());
                System.out.println(context.read("$.
[0:2]").toString());
                System.out.println(context.read("$.[?
(@.name=='Eraser')]").toString());
                System.out.println(context.read("$.[?
(@.quantity==5)]").toString());

        }

}
```

/src/test/java/com/in28minutes/unittesting/unittesting/UnitTestingApplicationTests.java

```
package com.in28minutes.unittesting.unittesting;

import org.junit.Test;
import org.junit.runner.RunWith;
import
org.springframework.boot.test.context.SpringBootTest;
```

```
import
org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)
@SpringBootTest
//@TestPropertySource(locations= {"classpath:test-
configuration.properties"})
public class UnitTestingApplicationTests {

        @Test
        public void contextLoads() {
        }

}
```

/src/test/resources/application.properties

```
spring.jpa.show-sql=false
spring.h2.console.enabled=false
```

References

References

- Intellij
  - https://www.jetbrains.com/help/idea/configuring-testing-libraries.html
  - https://blog.jetbrains.com/idea/2016/08/using-junit-5-in-intellij-idea/

- Spring & Spring Boot Framework - https://www.youtube.com/watch?v=PSP1-2cN7vM&t=893s
- Introduction to JPA and Hibernate using Spring Boot Data Jpa - http://www.springboottutorial.com/introduction-to-jpa-with-spring-boot-data-jpa
- Functional Programming - https://youtu.be/aFCNPHfvqEU
- JUnit - https://junit.org/junit5/docs/current/user-guide/
- AssertJ - https://joel-costigliola.github.io/assertj/
- Mockito - https://github.com/mockito/mockito/wiki/FAQ
- JsonPath - https://github.com/json-path/JsonPath
- Setting up JUnit 5 with Mockito and Spring Boot 2.0 - https://medium.com/@dSebastien/using-junit-5-with-spring-boot-2-kotlin-and-mockito-d5aea5b0c668
- Good Unit Testing
  - https://github.com/mockito/mockito/wiki/How-to-write-good-tests
  - FIRST. https://pragprog.com/magazines/2012-01/unit-tests-are-first
  - Patterns - http://xunitpatterns.com

- Mocking Static, Private Methods and Constructors
  - https://github.com/in28minutes/MockitoTutorialForBeginners/blob/master/Step15.md
  - https://github.com/in28minutes/MockitoTutorialForBeginners/tree/master/src/test/java/com/in28minutes/powermock

Become an expert on Spring Boot, APIs, Microservices and Full Stack Development

Checkout the Complete in28Minutes Course Guide