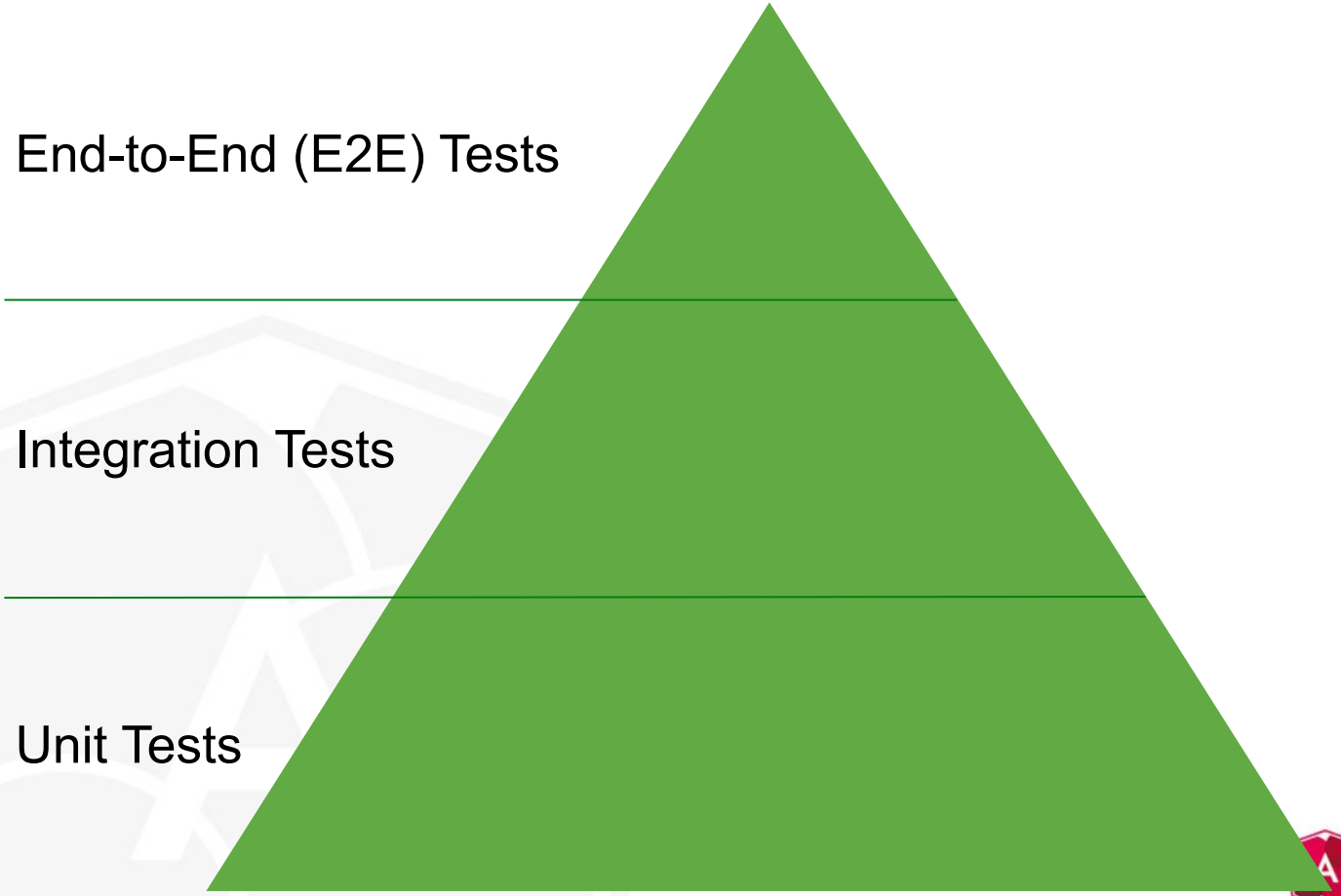
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Spring Workshop

2 - Testing



Agenda

- Unit Tests
  - JUnit
  - AssertJ
  - Mockito
- Integration Tests
  - @SpringBootTest

JUnit

- De-facto standard Java testing framework
- Ecosystem with lots of plugins/extensions
  - AssertJ
  - Mockito
  - Jacoco
- Shipped with Spring Boot

AssertJ

- Alternative to JUnits assertions
- Alternative to Hamcrest
- FluentAPI
- Type-Safe Matchers
- Special Assertions for
  - Property Matching
  - Thrown Exceptions
  - Type specific
- Has additional plugins

AssertJ

Fluent assertions for java

Basic Test

```
class CustomersRepositoryTest {


  @Test
  void testFullnameAlwaysStartsWithLastname() {
    this.franzMaier = new Customer(1L, "Franz", "Maier", true);
    this.repository = new CustomersRepository(true);
    assertThat(franzMaier.getFullname()).startsWith("Maier,");
  }
}
```

JUnit

AssertJ

Important Assertions

- `isTrue, isEmpty, isEqualTo`
- `isInstanceOf`
- Collections: `contains, hasSize`
- `extracting`
- `assertThatThrownBy`

https://assertj.github.io/doc/#assertj-core-assertions-guide

@BeforeEach, @BeforeAll, @AfterEach, @AfterAll

```
class CustomersRepositoryTest {

  Customer franzMaier;
  CustomersRepository repository;

  @BeforeEach
  void setupFranzMaier() {
    this.franzMaier = new Customer(1L, "Franz", "Maier", true);
    this.repository = new CustomersRepository(true);
  }

  @Test
  void testAddCustomer() {
    repository.add(franzMaier);
    assertThat(repository.findAll()).contains(franzMaier);
  }

  @Test
  void testEntryDateMustBeBeforeNow() {
    Instant entryDate = repository.add(franzMaier);
    assertThat(entryDate).isBefore(Instant.now());
  }

  @Test
  void testNonGdprSignedThrowsError() {
    franzMaier.setSignedGdpr(false);

    assertThatThrownBy(() -> repository.add(franzMaier));
  }
}
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

ObjectMother Pattern

- Builder Pattern
  - Customisable
  - Pre-Defined
  - Utility Methods (e.g. dealing with dates, nested objects)
- Can be based on Lombok's @Builder
  - Careful with initialised fields (@Builder.Default)

Example: ObjectMother

```
package com.softarc.eternal.domain;

public class CustomerMother {

  static Long id = 1L;

  public static Customer.CustomerBuilder franz() {
    return Customer
      .builder()
      .id(++id)
      .firstname("Franz")
      .lastname("Maier")
      .signedGdpr(true);
  }
}
```

Lombok

Example: ObjectMother in Action

```
class CustomersRepositoryTest {

  CustomersRepository repository;

  @BeforeEach
  void setup() {
    this.repository = new CustomersRepository(true);
  }

  @Test
  void testFullnameAlwaysStartsWithLastname() {
    var franz = CustomerMother.franz().build();
    assertThat(franz.getFullname()).startsWith("Maier,");
  }


  @Test
  void testNonGdprSignedThrowsError() {
    var franz = CustomerMother.franz().signedGdpr(false).build();
    assertThatThrownBy(() -> repository.add(franz));
  }
}
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Parameterized Tests

```
class CustomersRepositoryTest {

  @Test
  void testNoNumbersInLastname() {
    var franzMaier = CustomerMother.franz().lastname("Maier1").build();

    assertThatThrownBy(() -> repository.add(franzMaier))
      .hasMessage("Lastname is not valid");
  }

  @Test
  void testNoSpacesInLastname() {
    var franzMaier = CustomerMother.franz().lastname("Franz Maier").build();
    assertThatThrownBy(() -> repository.add(franzMaier))
      .hasMessage("Lastname is not valid");
  }

  @Test
  void testNoHahnekamp() {
    var franzMaier = CustomerMother.franz().lastname("Hahnekamp").build();
    assertThatThrownBy(() -> repository.add(franzMaier))
      .hasMessage("Lastname is not valid");
  }
}
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Parameterized Tests

```
class CustomersRepositoryTest {

  @Test
  void testNoNumbersInLastname() {
    var franzMaier = CustomerMother.franz().lastname("Maier1").build();

    assertThatThrownBy(() -> repository.add(franzMaier))
      .hasMessage("Lastname is not valid");
  }

  @Test
  void testNoSpacesInLastname() {
    var franzMaier = CustomerMother.franz().lastname("Franz Maier").build();
    assertThatThrownBy(() -> repository.add(franzMaier))
      .hasMessage("Lastname is not valid");
  }

  @Test
  void testNoHahnekamp() {
    var franzMaier = CustomerMother.franz().lastname("Hahnekamp").build();
    assertThatThrownBy(() -> repository.add(franzMaier))
      .hasMessage("Lastname is not valid");
  }
}
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Parameterized Tests

```
class CustomersRepositoryTest {

  @ParameterizedTest

  @ValueSource(strings = { "Maier1", "Franz Maier", "Hahnekamp" })

  void testNoNumbersInLastname(String lastname) {

    var franzMaier = CustomerMother.franz().lastname(lastname).build();


    assertThatThrownBy(() -> repository.add(franzMaier))

      .hasMessage("Lastname is not valid");

  }

}
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Parameterized Tests with own types

```
record LastnameCheck(String lastname, boolean isValid) {}

class CustomersRepositoryTest {

  @ParameterizedTest
  void testNoNumbersInLastname(LastnameCheck parameter) {
    var franzMaier = CustomerMother
      .franz()
      .lastname(parameter.lastname())
      .build();

    if (parameter.isValid()) {
      assertThatCode(() -> repository.add(franzMaier))
        .doesNotThrowAnyException();
    } else {
      assertThatThrownBy(() -> repository.add(franzMaier))
        .hasMessage("Lastname is not valid");
    }
  }
}
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Parameterized Tests with own types

```
class CustomersRepositoryTest {

  CustomersRepository repository;

  static ArrayList<LastnameCheck> provideLastnameChecks() {
    var checks = new ArrayList<LastnameCheck>();
    checks.add(new LastnameCheck("Maier1", false));
    // ...

    return checks;
  }

  @ParameterizedTest
  @MethodSource("provideLastnameChecks")
  void testNoNumbersInLastname(LastnameCheck parameter) {
    // ...
}
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Lab Time

Mockito

- `mock`: Mocking: Generates instance of any class
- `when`: define behaviour of a method
- `verify`: validate that methods have been called
- `ArgumentCaptor`

Mockito: mock

```
public class CustomersControllerTest {

  @Test
  void testRepo() {
    var repository = mock(CustomersRepository.class);
    var controller = new CustomersController(repository);
  }
}
```

Mockito: when

```
public class CustomersControllerTest {

  @Test
  void testRepo() {
    var franz = CustomerMother.franz().build();
    var repository = mock(CustomersRepository.class);
    var controller = new com.softarc.eternal.CustomersController(repository);
    when(repository.findAll()).thenReturn(Collections.singletonList(franz));
  }
}
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Mockito: mock

```
public class CustomersControllerTest {

  @Test
  void testRepo() {
    var repository = mock(CustomersRepository.class);
    var controller = new com.softarc.eternal.CustomersController(repository);
  }
}
```

Mockito: verify

```
public class CustomersControllerTest {

  @Test
  void testRepo() {
    var franz = CustomerMother.franz().build();
    var repository = mock(CustomersRepository.class);
    var controller = new com.softarc.eternal.CustomersController(repository);
    when(repository.findAll()).thenReturn(Collections.singletonList(franz));

    controller.index();

    verify(repository, times(1)).findAll();
  }
}
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Mockito: ArgumentCaptor

```
public class CustomersControllerTest {

  @Test
  void testAdd() {
    var franz = CustomerMother.franz().build();
    ArgumentCaptor<Customer> customerCaptor = ArgumentCaptor.forClass(
      Customer.class
    );
    var repository = mock(CustomersRepository.class);
    repository.add(franz);
    verify(repository).add(customerCaptor.capture());

    assertThat(customerCaptor.getValue()).isEqualTo(franz);
  }
}
```

Mockito: Modern Version

```
@ExtendWith(MockitoExtension.class)
public class CustomersControllerTest {

  @Mock CustomersRepository repository;

  @Captor ArgumentCaptor<Customer> customerCaptor;

  @Test
  void testAdd() {
    var franz = CustomerMother.franz().build();
    repository.add(franz);
    verify(repository).add(customerCaptor.capture());

    assertThat(customerCaptor.getValue()).isEqualTo(franz);
  }
}
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Integration Tests with `@SpringBootTest`

- Available Dependency Injection
- Overriding Properties
- Integrates Mockito
- Specialized Testing Types available
    - @WebMvcTest
    - @DataJpaTest
    - @RestClientTest
    - ...

Controller Test Rewritten (Unit Test Style)

```
@SpringBootTest
public class CustomersControllerUnitTest {

  @Autowired CustomersController controller;

  @MockBean CustomersRepository repository;

  @Captor ArgumentCaptor<Customer> customerCaptor;

  @Test
  void testAdd() {
    var franz = CustomerMother.franz().build();
    repository.add(franz);
    verify(repository).add(customerCaptor.capture());

    assertThat(customerCaptor.getValue()).isEqualTo(franz);
  }
}
```

Controller Test Rewritten (Integration Test Style)

```
@SpringBootTest
public class CustomersControllerIntegrationTest {

  @Autowired CustomersController controller;

  @Autowired CustomersRepository repository;

  @Test
  void testRepo() {
    var franz = CustomerMother.franz().build();
    controller.add(franz);
    assertThat(controller.index()).containsOnly(franz);
  }
}
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Unit vs. Integration Tests

Unit Tests

- ✅ Fast
- ✅ Precision & Control
- ✅ Explorative
- ✅ Good Code Design
- ⛔ Tightly Coupled
- ⛔ Don't support Refactoring
- ⛔ Low Coverage

Integration Tests

- ✅ Loosely Coupled
- ✅ Support Refactoring
- ✅ High Coverage
- ⛔ Slow
- ⛔ Hard to Write

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Lab Time