

RINGKASAN: Apa yang Harus QA Engineer Lakukan

Panduan Lengkap API Automation Framework

1. PERSIAPAN AWAL

Setup Environment

- **Java 11+** harus terinstall
- **IDE** (IntelliJ IDEA/Eclipse/VS Code)
- **Git** untuk version control
- **Gradle** (sudah included dalam project)

Clone dan Setup Project

```
# Clone repository
git clone <repository-url>
cd api-automation-framework

# Verifikasi setup
./gradlew build
```

2. STRUKTUR PROJECT YANG HARUS DIPAHAMI

Struktur Direktori

```
src/
├── main/java/com/api/automation/
│   ├── config/
│   │   └── ApiConfig.java           # Konfigurasi URL dan endpoints
│   ├── models/
│   │   ├── User.java               # Model data User
│   │   └── Post.java               # Model data Post
│   └── utils/
│       └── BaseApiUtils.java        # Utilities REST Assured
└── test/java/com/api/automation/
    ├── tests/
    │   ├── UserApiTest.java         # Test untuk User API
    │   ├── PostApiTest.java         # Test untuk Post API
    │   └── ReqResApiTest.java       # Test untuk ReqRes API
    └── utils/
        └── TestDataGenerator.java   # Generator data test
```

File Konfigurasi Penting

- **build.gradle** - Dependencies dan build configuration
 - **ApiConfig.java** - Base URLs dan endpoints
 - **BaseApiUtils.java** - REST Assured setup
-

3. CARA MENULIS TEST API

Template Dasar Test Class

```

@Epic("API Testing")
@Feature("User Management")
public class UserApiTest extends BaseApiUtils {

    @Test
    @Story("Get User by ID")
    @Severity(SeverityLevel.CRITICAL)
    @Description("Test untuk mengambil data user berdasarkan ID")
    public void testGetUser_ValidId() {
        // Given
        int userId = 1;

        // When
        Response response = given()
            .when()
            .get(JSONPLACEHOLDER_BASE_URL + USERS_ENDPOINT + "/" + userId);

        // Then
        response.then()
            .statusCode(OK)
            .body("id", equalTo(userId))
            .body("name", notNullValue());
    }
}

```

Naming Convention untuk Test

- **Positive Test:** testAction_ValidInput()
- **Negative Test:** testAction_InvalidInput()
- **Boundary Test:** testAction_BoundaryValue()

4. JENIS-JENIS TEST YANG WAJIB DIBUAT

Positive Testing

```

@Test
public void testCreateUser_ValidData() {
    User user = TestDataGenerator.generateValidUser();

    Response response = given()
        .contentType(ContentType.JSON)
        .body(user)
        .when()
        .post(JSONPLACEHOLDER_BASE_URL + USERS_ENDPOINT);

    response.then()
        .statusCode(CREATED)
        .body("name", equalTo(user.getName()));
}

```

Negative Testing

```

@Test
public void testCreateUser_EmptyName() {
    User user = new User();
    user.setName(""); // Invalid empty name

    Response response = given()
        .contentType(ContentType.JSON)
        .body(user)
        .when()
        .post(JSONPLACEHOLDER_BASE_URL + USERS_ENDPOINT);

    response.then()
        .statusCode(BAD_REQUEST);
}

```

Boundary Testing

```
@ParameterizedTest
@ValueSource(ints = {0, -1, 9999})
public void testGetUser_InvalidIds(int invalidId) {
    Response response = given()
        .when()
        .get(JSONPLACEHOLDER_BASE_URL + USERS_ENDPOINT + "/" + invalidId);

    response.then()
        .statusCode(NOT_FOUND);
}
```

5. BEST PRACTICES YANG HARUS DIIKUTI

Test Design

1. **Test Independence** - Setiap test harus berdiri sendiri
2. **Clear Assertions** - Validasi status code dan response body
3. **Descriptive Names** - Nama test harus jelas menggambarkan scenario
4. **Data Isolation** - Gunakan test data yang tidak saling bergantung

Allure Annotations

```
@Epic("API Testing")           // Level tertinggi (seluruh aplikasi)
@Feature("User Management")    // Fitur yang dites
@Story("Create New User")      // Story spesifik
@Severity(SeverityLevel.CRITICAL) // Tingkat kepentingan
>Description("Detailed test description")
```

REST Assured Best Practices

```
// Setup di BaseApiUtils
RequestSpecification requestSpec = new RequestSpecBuilder()
    .setBaseUri(baseUrl)
    .setContentType(ContentType.JSON)
    .addFilter(new AllureRestAssured())
    .build();

// Penggunaan dalam test
given(requestSpec)
    .when()
    .get("/endpoint")
    .then()
    .spec(responseSpec);
```

6. WORKFLOW TESTING HARIAN

Morning Routine

1. **Pull latest code** dari repository
2. **Run existing tests** untuk memastikan tidak ada regression
3. **Check test results** dan Allure reports

Development Cycle

1. **Analyze requirements** - Pahami API endpoints yang akan dites
2. **Write test cases** - Buat positive, negative, dan boundary tests
3. **Execute tests** - Jalankan dan validasi hasil
4. **Generate reports** - Create Allure reports untuk dokumentasi

Debugging Process

```
// Gunakan logging untuk debugging
log.info("Testing endpoint: {}", endpoint);
log.debug("Request body: {}", requestBody);
log.error("Test failed with response: {}", response.asString());

// Extract dan validasi response
String responseBody = response.asString();
System.out.println("Response: " + responseBody);

// Assertion bertahap
response.then()
    .statusCode(200)
    .body("size()", greaterThan(0))
    .body("[0].id", notNullValue());
```

7. COMMAND YANG WAJIB DIKUASAI

Build dan Test

```
# Build project
./gradlew build

# Run semua tests
./gradlew test

# Run test class tertentu
./gradlew test --tests UserApiTest

# Run test method tertentu
./gradlew test --tests UserApiTest.testGetUser_ValidId
```

Generate Reports

```
# Generate Allure report
./gradlew allureReport

# Serve Allure report (buka di browser)
./gradlew allureServe

# Clean dan rebuild
./gradlew clean build
```

8. TROUBLESHOOTING UMUM

Masalah Umum dan Solusi

Masalah	Penyebab	Solusi
Test timeout	API response lambat	Increase timeout di BaseApiUtils
401 Unauthorized	Missing authentication	Add auth headers atau token
JSON parsing error	Response format salah	Validate response structure
Connection refused	API server down	Check API availability

Debug Steps

1. **Check logs** - Lihat console output dan log files
2. **Validate request** - Pastikan request format benar
3. **Check response** - Print response body untuk debugging
4. **Verify endpoints** - Pastikan URL dan method benar

9. METRICS DAN KPI

Test Coverage Metrics

- **API Endpoints Coverage:** Minimal 90%

- **HTTP Methods Coverage:** GET, POST, PUT, DELETE
- **Status Codes Coverage:** 2xx, 4xx, 5xx
- **Test Types:** Positive (60%), Negative (30%), Boundary (10%)

Performance Metrics

- **Test Execution Time:** < 5 menit untuk full suite
- **Individual Test Time:** < 30 detik per test
- **Report Generation:** < 2 menit

10. LEARNING PATH UNTUK QAENGINEER

Beginner (Week 1-2)

1. Pahami REST API concepts
2. Setup dan run existing tests
3. Modify test data dan assertions
4. Generate dan baca Allure reports

Intermediate (Week 3-4)

1. Write new test cases
2. Implement parameterized tests
3. Add custom assertions
4. Debug failing tests

Advanced (Week 5+)

1. Framework customization
2. CI/CD integration
3. Custom reporting
4. Performance testing

RESOURCES DAN DOKUMENTASI

Dokumentasi Resmi

- [REST Assured Documentation \(https://rest-assured.io/\)](https://rest-assured.io/)
- [JUnit 5 User Guide \(https://junit.org/junit5/docs/current/user-guide/\)](https://junit.org/junit5/docs/current/user-guide/)
- [Allure Framework \(https://docs.gameta.io/allure/\)](https://docs.gameta.io/allure/)

Useful Links

- [JSONPlaceholder API \(https://jsonplaceholder.typicode.com/\)](https://jsonplaceholder.typicode.com/)
- [ReqRes API \(https://reqres.in/\)](https://reqres.in/)
- [HTTP Status Codes \(https://httpstatuses.com/\)](https://httpstatuses.com/)

Cheat Sheets

```
// Common REST Assured methods
given().when().get()      // GET request
given().when().post()     // POST request
given().when().put()      // PUT request
given().when().delete()   // DELETE request

// Common assertions
.statusCode(200)           // Check status code
.body("key", equalTo())   // Check JSON value
.time(lessThan())         // Check response time
.header("key", value)     // Check header
```

CHECKLIST HARIAN QAENGINEER

Daily Tasks

- [] Pull latest code changes

- ☐ Run existing test suite
- ☐ Check for failing tests
- ☐ Analyze new requirements
- ☐ Write new test cases
- ☐ Execute and validate tests
- ☐ Generate test reports
- ☐ Update documentation


Weekly Review

- ☐ Review test coverage
- ☐ Analyze test execution trends
- ☐ Update test data
- ☐ Refactor duplicate code
- ☐ Performance analysis
- ☐ Team knowledge sharing

KESIMPULAN

Sebagai QA Engineer yang menggunakan framework ini, fokus utama adalah:

1. **Memahami struktur project** dan file konfigurasi
2. **Menulis test yang komprehensif** (positive, negative, boundary)
3. **Mengikuti best practices** dalam naming dan coding
4. **Menggunakan Allure annotations** untuk reporting yang baik
5. **Melakukan debugging** yang efektif saat test gagal
6. **Generate reports** secara rutin untuk dokumentasi
7. **Continuous learning** untuk improve testing skills

Remember: Quality is not an act, it's a habit! 

*Generated by API Automation Framework - QA Engineer Guide
Last Updated: June 2025*