The Power of Atomic Automated Tests: Our Experience of Improved Test Efficiency and Reliability

Sargis Sargsyan

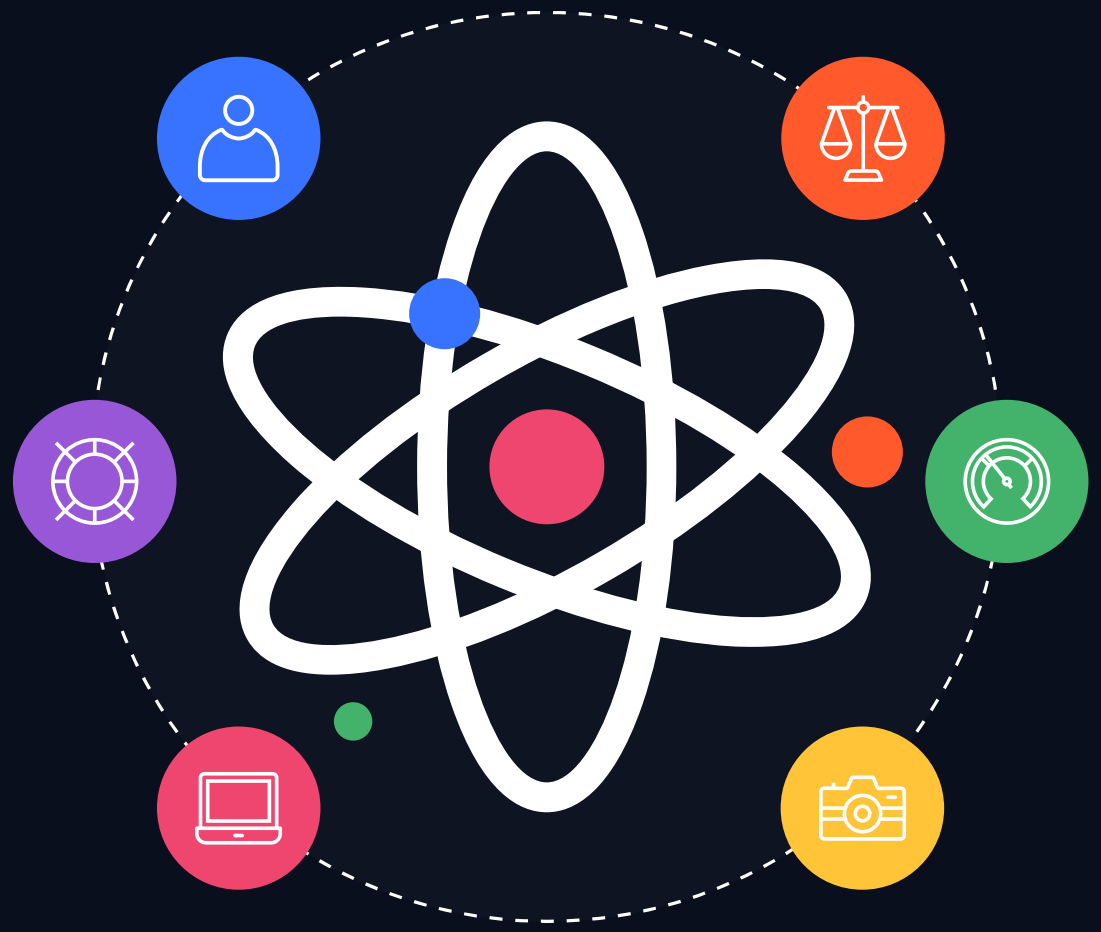
HUSTEF

HUNGARIAN SOFTWARE TESTING FORUM

WELCOME

Sargis Sargsyan

Independent Consultant, Public Speaker



INTRODUCTION TO ATOMIC TESTS

Atomic tests, in the context of test automation, are highly focused and small-scale tests that concentrate on testing a single specific feature or functionality. They aim to isolate the testing scope to ensure that each test verifies only one aspect of the application.

The Benefits of Atomic Tests

- Faster Execution
- Higher stability
- Easier maintenance
- Clearer test cases
- Enhanced reusability

Atomic tests play a pivotal role in ensuring high-quality software. Their focused nature, reusability, and speed make them essential in modern development workflows.

Atomic vs non-Atomic Tests

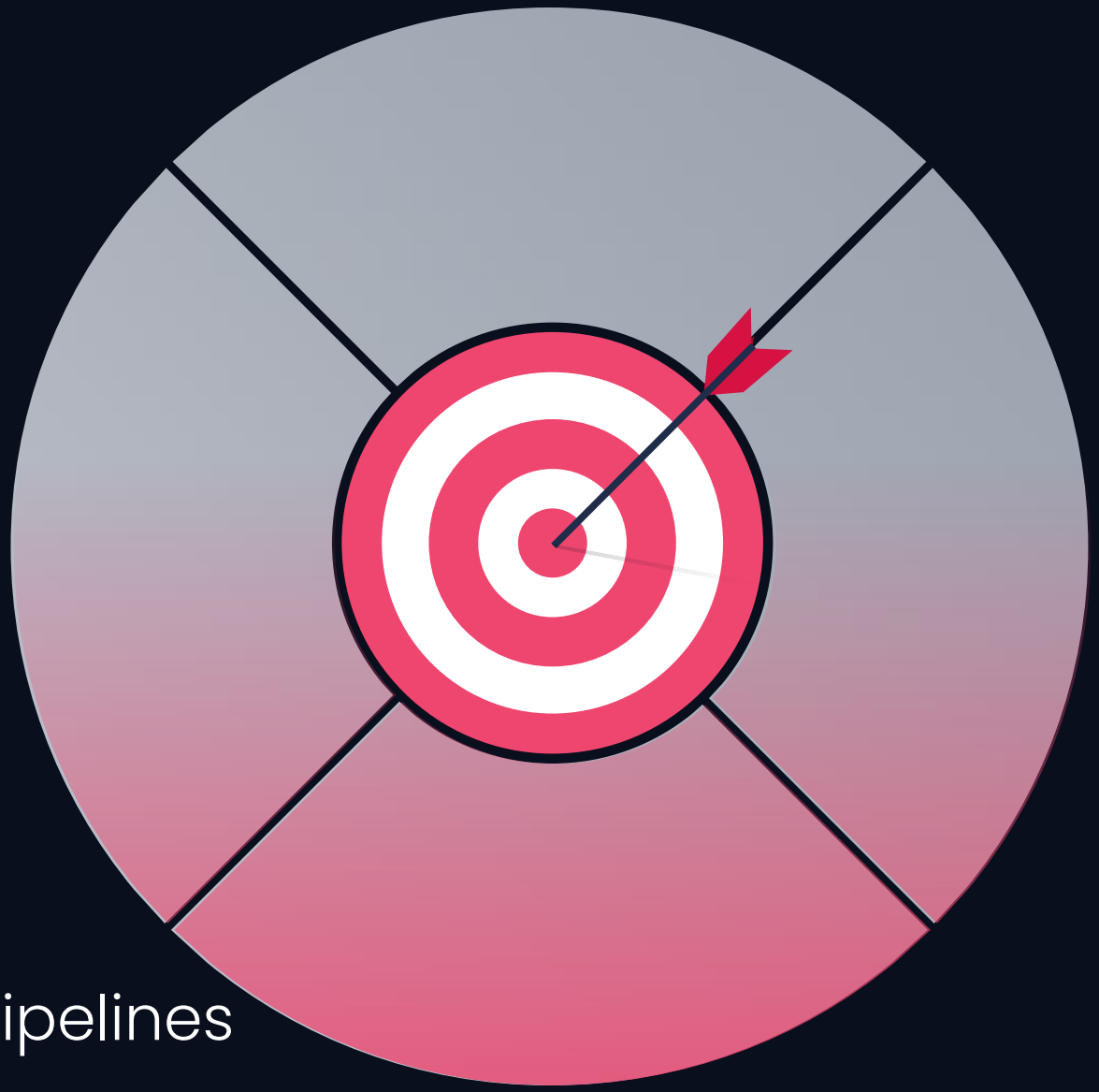
```
@Test
public void projectPage() {
    LoginPage loginPage = new LoginPage().open();
    loginPage.login( username: "test@user.test",  password: "TestPass2022");

    ProjectsPage projectsPage = new ProjectsPage().open();
    NewProjectPage newProjectPage = projectsPage.clickNewProjectButton();
    NewScrumProjectPage newScrumProjectPage = newProjectPage.selectScrum();
    newScrumProjectPage.typeProjectName("project name x");
    newScrumProjectPage.typeProjectDescription("project name x");
    ProjectBacklogPage projectBacklogPage = newScrumProjectPage.clickCreateProject();

    //do checks
}
```

Atomic vs non-Atomic Tests

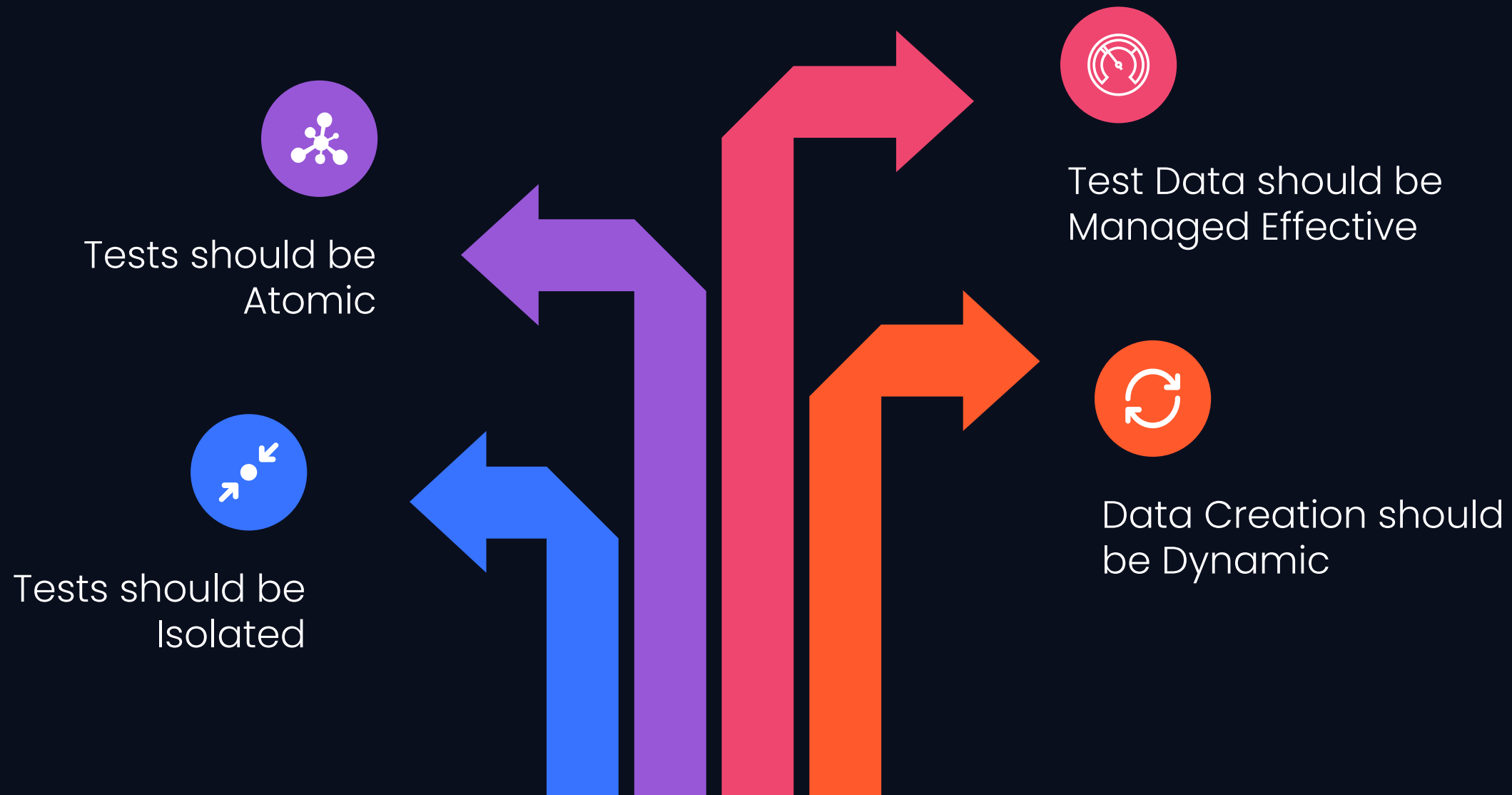
```
@Test
public void projectPageWithData() {
    JsonObject newProject = ProjectService.createProject();
    ProjectPage projectsPage = new ProjectPage(newProject).open();

    // do checks
}
```

Importance of Atomic Tests

- Isolates specific functionalities
- Enables PARALLEL EXECUTION
- Easier maintenance
- Reduces debugging time
- Provides quicker feedback in CI pipelines



Parallel Execution



How to Achieve Atomic Test Automation

Test Data Preparation

Database Seeding

Java Script Injection

Rest API Usage

Test Data Preparation

Test Data Preparation

Test Run

Clean up

Database Seeding

Database seeding complements atomic tests by providing a consistent foundation for test scenarios. By pre-populating the database with controlled data, you can achieve focused, reliable, and repeatable tests that contribute to the overall success of your test automation strategy.

Import Data into the Table

	id	name	ingredients	med_type	shape	colors	strength
2	2	PAREDRINE	{HYDROXYAMPHETAMINE …	<null>	<null>	<null>	{1%}
3	3	SULFAPYRIDINE	{SULFAPYRIDINE}	<null>	<null>	<null>	{500MG}
4	4	LIQUAEMIN SO…	{HEPARIN SODIUM}	<null>	<null>	<null>	{20,000 UNI…
5	5	LIQUAEMIN LO…	{HEPARIN SODIUM}	<null>	<null>	<null>	{100 UNITS/…
6	7	LIQUAEMIN SO…	{HEPARIN SODIUM}	<null>	<null>	<null>	{1,000 UNIT…
7	8	HISTAMINE PH…	{HISTAMINE PHOSPHATE}	<null>	<null>	<null>	{EQ 1MG BAS…
8	10	DOCA	{DESOXYCORTICOSTERON…	<null>	<null>	<null>	{5MG/ML}
9	11	VERARD	{VERARD}	<null>	<null>	<null>	{UNKNOWN}
10	12	GUANIDINE HY…	{GUANIDINE HYDROCHLO…	<null>	<null>	<null>	{125MG}

Import Data into the Table

```
InputStream csvStream = PostgresUtils.class.getResourceAsStream( name: "/medication.csv");
CopyManager copyManager = new CopyManager((BaseConnection) connection);
String sqlCopy = "COPY medication FROM STDIN WITH CSV HEADER";
long rs = copyManager.copyIn(sqlCopy, csvStream);
log.info("Result: " + rs + " Medications were imported to DB.");
```

Update Data in the Table

```
stmt = connection.createStatement();
String sql = "UPDATE user_data SET email_verified_at='2023-08-04 18:02:33.000000' " +
                    "WHERE email='oyg1hd9x@keeplabs-test.com'";
log.info("Running SQL Query: " + sql);
int rs = stmt.executeUpdate(sql);
log.info("Result: " + rs + " item(s) were updated");
stmt.close();
```

Select Data from the Table

```
stmt = connection.createStatement();
String sql = "SELECT TOKEN FROM OTP WHERE EMAIL='ap3j4@keeplabs-test.com'";
log.info("Running SQL Query: " + sql);
ResultSet rs = stmt.executeQuery(sql);
while (rs.next()) {
    result.add(rs.getString( columnLabel: "otp"));
}
log.info("Result: " + result);
rs.close();
stmt.close();
```

Power of JavaScript

JavaScript usage empowers testers to set up intricate scenarios quickly and efficiently. By directly manipulating the application's behavior, you can streamline data preparation for atomic tests and achieve a more robust and comprehensive test suite.

JavaScript Use Cases for Automation Tests

- Add/Modify Cookies, Local/Session Storage
- Element Manipulation
- Event Simulation
- Asynchronous Operations

Local Storage Add/Modify

```
// Use JavaScript to modify local storage
String localStorageKey = "userData";
String newValue = "{ \"username\": \"john_doe\", \"email\": \"john@example.com\" }";

// Set new data in local storage
jsExecutor.executeScript( script: "localStorage.setItem(arguments[0], arguments[1]);",
        localStorageKey, newValue);
```

Element Manipulation

```
// WebDriver command to identify the element
WebElement element = driver.findElement(By.id("elementId"));

// JavaScript code to change the text content
String jsScript = "arguments[0].textContent = 'New Text';";

// Inject JavaScript to update element content
((JavascriptExecutor) driver).executeScript(jsScript, element);
```

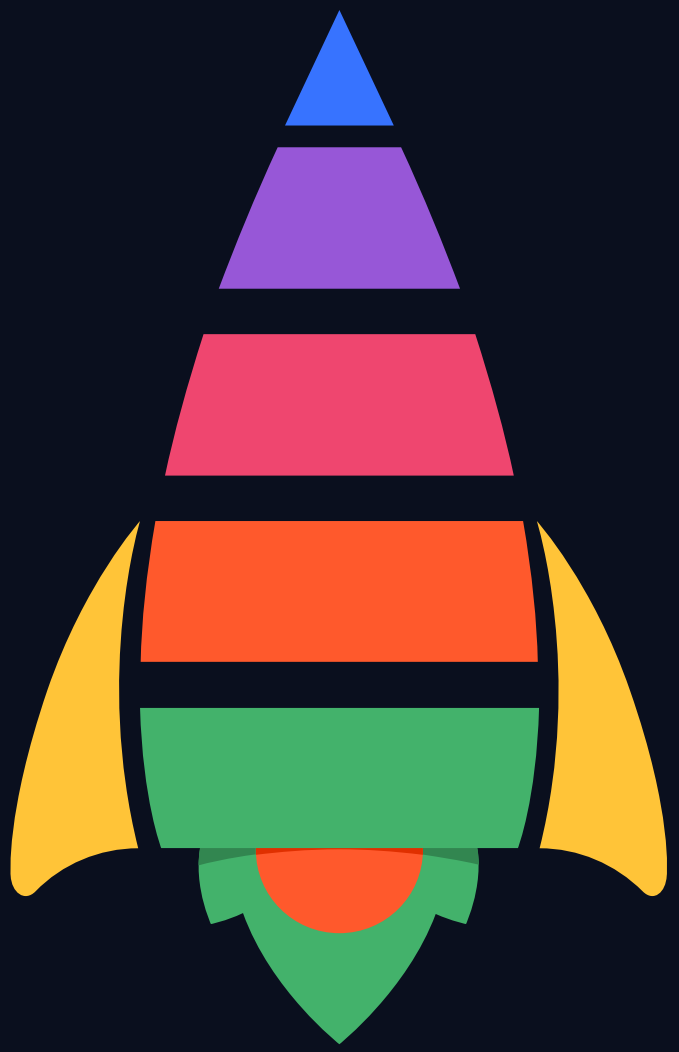
Event Manipulation

```
 // Create a JavascriptExecutor object
JavascriptExecutor jsExecutor = (JavascriptExecutor) driver;

// Scroll down the page by 500 pixels
jsExecutor.executeScript( s: "window.scrollBy(0, 500);");
```

Asynchronous Operations

```
jsExecutor.executeAsyncScript(
    s: "var callback = arguments[arguments.length - 1];" +
    "setTimeout(function() {" +
    "    var resultElement = document.getElementById('result');" +
    "    callback(resultElement.textContent);" +
    "}, 3000);"
);
```

Test Data Preparation with API



- Login without UI
- Project Creation
- Clean Up

Test Data Preparation with API

```
@Test
public void buttonsVisibility() {
    //login
    LoginPage loginPage = new LoginPage().open();
    loginPage.setUsername("testjanqwerty@gmail.com");
    loginPage.setPassword("TestPass!");
    HomePage homePage = loginPage.clickSubmit();
    //navigate
    homePage = homePage.init();
    ProjectsPage projectsPage = homePage.clickFolderIcon();
    //create project
    projectsPage = projectsPage.init();
    projectsPage.clickCreateProject();
    projectsPage.selectProjectType();
    projectsPage.setProjectName("Testing Project Name");
    projectsPage.setProjectDescription("Testing Project Description");
    projectsPage.submit();
    //navigate to Project page
    ProjectPage projectPage = new ProjectPage().init();
    projectPage.clickTimeline();
    //make an assertions
    assertTrue(projectPage.isLikeButtonVisible(), s: "Like button should be visible!");
    assertTrue(projectPage.istrackButtonVisible(), s: "Track button should be visible!");
}
```

Login without UI

DISCOVER PROJECTS

52911 public projects to discover

Type something...

MOST LIKED — Last week

Avans Portal — 5 5 7

Avans portal project scrum board

MOST ACTIVE — Last week

Valoo — 12 73 27

Regroupe tous les services de la plateforme : - Site web principal (www.valoo.com) - API principale - Serveur d'indexation - Serveur

Elements | Console | Sources | Network | Performance | Memory | Application | Security | Audits

Clear storage

Storage

- Local Storage
  - https://tree.taiga.io
  - ://
- Session Storage
- IndexedDB

Filter

Key	Value
intercom-state	{"app":{"color":"#5b8200","secondaryColor":"#5b8200","name":"Taiga","features":{"inboundMes...
token	"eyJ1c2VyX2F1dGhlbnRpY2F0aW9uX2lkIjozMDA5NTJ9:1fKfkT:_hpWFZuoSZZ-NtYUJAhEaJClrOk"
userInfo	{"email":"sqa.days@yandex.ru","id":300952,"date_joined":"2018-04-22T09:15:21.191Z","max_m...

1 "eyJ1c2VyX2F1dGhlbnRpY2F0aW9uX2lkIjozMDA5NTJ9:1fKfkT:_hpWFZuoSZZ-NtYUJAhEaJClrOk"

Login without UI

Help

DISCOVER PROJECTS

52911 public projects to discover

Elements Console Sources Network Performance Memory Application Security Audits

View: Group by frame Preserve log Disable cache Offline Online

Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other

10000 ms 20000 ms 30000 ms 40000 ms 50000 ms 60000 ms 70000 ms 80000 ms 90000 ms 100000 ms 110000 ms 120000 ms 130000 ms 140000 ms 150

Name

- auth
- projects?member=300952&order_by...
- joyride
- discover
- projects?discover_mode=true&order_...
- projects?discover_mode=true&order_...
- projects?discover_mode=true&is_fea...
- collect?v=1&_v=j67&a=530967766&t...

Headers Preview Response Timing

General

Request URL: `https://api.taiga.io/api/v1/auth`
Request Method: `POST`
Status Code: 200 OK
Remote Address: `5.57.231.21:443`
Referrer Policy: `no-referrer-when-downgrade`

Response Headers view source

Access-Control-Allow-Credentials: `true`
Access-Control-Allow-Headers: `content-type,x-requested-with,authorization,accept-encoding,x-disable-pagination,x-lazy-paginati`
ders

Login without UI

```
MediaType mediaType = MediaType.parse( string: "application/x-www-form-urlencoded");
RequestBody body = RequestBody.create(mediaType,  content: "username=" + username +
    "&password=" + password + "&type=normal");
Request request = new Request.Builder()
    .url(BASE_URL + "/auth")
    .post(body)
    .addHeader( name: "Content-Type",  value: "application/x-www-form-urlencoded")
    .build();
    response = client.newCall(request).execute();
    responseJson = response.body().string();
return responseJson;
```

Login without UI

```
1 usage   sargis
public void login(String email, String password) {
    JsonObject userJson = UserService.login(email, password);
    new LoginPage().open();
    ((JavascriptExecutor) driver)
            .executeScript( s: "window.localStorage.setItem('token','" +
                    userJson.get("auth_token") + "');");
    ((JavascriptExecutor) driver)
            .executeScript( s: "window.localStorage.setItem('userInfo','" +
                    userJson + "');");
}
```

Login without UI

```
Cookie cookie = new Cookie( name: "token", token);
getDriver().manage().addCookie(cookie);
```

Test Data Preparation with API

```
public static JsonObject createProject() {
    HashMap<String, Object> projectMap = new HashMap<>();
    projectMap.put("is_private", false);
    projectMap.put("creation_template", 1);
    projectMap.put( "name", "Test Project Name " + randomString( len: 5));
    projectMap.put( "description", "Test Project Description" + randomString( len: 10));
    String jsonString = gson.toJson(projectMap);
    Response response = HttpClient.post( url: "/projects", jsonString);
    return getJsonObject(response);
}
```

Test Data Clean Up with API

```
public static void deleteProject(JsonObject project) {
    HttpClient.delete( url: "/projects/" + project.get("id").getAsString());
}
```

Test Data Preparation with API

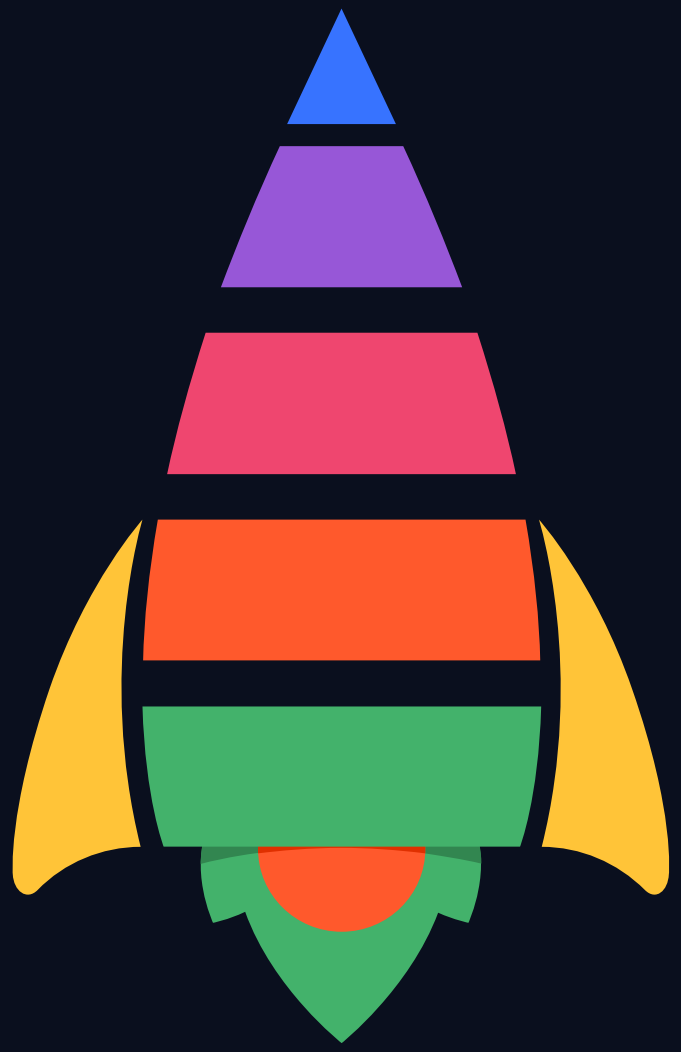
```
@Test
public void trackAndLikeButtonsVisibility() {
    ProjectPage projectPage = new ProjectPage(project);
    projectPage = projectPage.open();

    assertTrue(projectPage.isLikeButtonVisible(), s: "Like button should be visible!");
    assertTrue(projectPage.istrackButtonVisible(), s: "Track button should be visible!");
}
```

Advantages of Atomic Tests

If it’s Going to Fail, Fail Fast

- **Fail Fast, Fail Early:** Atomic tests offer the advantage of early failure detection. This means you can quickly identify issues and receive timely feedback.

- **Swift Feedback:** When assessing a feature's status, atomic tests provide rapid results. It takes no more than one minute to evaluate the feature's functionality.



Improved Test Workflow

- **Enhanced Test Isolation:** Atomic tests offer improved testing workflows. A failing test won't hinder the evaluation of other functionalities.

- **Risk of Reduced Coverage:** Long, monolithic tests can reduce overall test coverage, especially when dependencies are involved.

Swift Execution

- **Efficiency of Atomic Tests:** Atomic tests are characterized by their brevity and speed.
- **Parallel Execution:** When parallelized, atomic tests exhibit remarkable speed improvements.
- **Dramatic Time Reduction:** Parallel execution reduced the average test duration to a few seconds.

Thank You!

@sargiset

/sargissargsyan

https://sargissargsyan.com