API Testing Examples – Documentation

The following documentation describes a set of practical files used to demonstrate RESTful API testing within a Playwright automation framework. These examples illustrate direct API validation, backend mocking, and test data management, techniques commonly employed when testing web applications that interface with backend services or embedded systems.

API tests are maintained separately from end-to-end browser tests to ensure faster execution, reduced flakiness, and improved isolation during continuous integration workflows. The files described below are in my /RESTful_API_Tests/ directory.

1. playwright-threat-status-test.ts

Purpose

This Playwright test script performs validation of a RESTful GET endpoint responsible for retrieving status information (detection or anomaly data from a controlled system). It leverages Playwright's APIRequestContext fixture to execute direct API calls, bypassing the browser for greater speed and reliability.

Key Features

- Constructs and sends an authenticated GET request including query parameters
- Validates HTTP status codes and overall response structure
- Performs assertions on critical response fields such as location and confidence score
- Designed for straightforward extension to cover negative scenarios, error conditions, or additional business rules

Usage

This approach is particularly effective for backend-focused regression testing and for verifying APIs that supply data to web-based dashboards or control interfaces. It integrates well into CI/CD pipelines as a fast-running preliminary check before broader end-to-end execution.

2. express-threat-api-server.js

Purpose

This lightweight Node.js/Express application serves as a local mock server implementing the /api/threat/status endpoint. It provides controlled responses during test development and execution when access to a production or development backend is unavailable or undesirable.

Key Features

- Implements a simple GET endpoint with basic query parameter validation
- Returns predefined JSON payloads that can be easily modified
- Launches quickly for local development and debugging sessions

Usage

The mock server is valuable during early test authoring, when the actual backend services are still under development, or when consistent, deterministic responses are required across test runs. It can also be utilized in controlled CI environments to eliminate external dependencies.

3. sample-threat-json-payloads.json

Purpose

This file contains representative JSON response payloads covering both positive (detections present) and negative (no detections) scenarios. These payloads are intended for direct use in test mocking or as reference examples.

Key Features

- Provides realistic data structure including identifiers, geographic coordinates, confidence levels, and timestamps
- Includes distinct cases to support comprehensive test coverage
- Facilitates rapid configuration of mocked responses within Playwright

Usage

These payloads are ideally suited for data-driven testing and response mocking via Playwright's route.fulfill() method. They enable precise control over test conditions, ensuring reliable validation of application behavior across various operational states.

These components collectively form a robust foundation for API-level automated testing. They can be readily expanded to accommodate additional endpoints, authentication strategies, or integration with larger test frameworks as project requirements evolve.

Jaan John