Synovus MuleSoft API Test Automation - Reusable Components

Table of Contents

Utils folder

ApiPropertiesContext

This class is a Spring Configuration class that defines beans for various API-related properties. This class leverages Spring's `@Value` annotation to inject property values from the application's configuration file into private fields. It then exposes these values as beans using methods annotated with `@Bean`, making them available for dependency injection throughout the Spring application. This setup ensures that critical API endpoint configurations are centrally managed and easily accessible.

BeanFactory

`BeanFactory` class is a Spring Configuration class responsible for creating and managing beans related to application functionality. It uses the `@Autowired` annotation to inject dependencies, the `@Value` annotation to inject configuration properties, and the `@Bean` annotation to define methods that return bean instances. This class configures and provides an `AdoManager` bean, which is initialized with organization, project, and API version properties, and a `Faker` bean for generating fake data. The use of the `@Lazy` annotation on the Hook dependency indicates that it will be initialized lazily, improving startup performance by delaying its instantiation until it is needed.

EndpointReader

This class is a Spring Component that reads endpoint properties from configuration files based on the environment. It loads these properties into a map for easy retrieval during runtime.

HeaderHelper

The `HeaderHelper` class manages headers for REST Assured requests. It handles token authentication and environment configuration, providing methods for JSON and SOAP requests.

JsonHelper

The `JsonHelper` class is a component that provides utility methods for working with JSON data. It includes functionalities for comparing JSON strings, searching for field values in JSON responses.

PayloadHelper

The `PayloadHelper` class is a component that assists in handling payloads for MuleSoft applications. It provides methods for reading payload files, creating JSON payloads from velocity templates and properties files, and converting XML payloads to strings. This utility class utilizes `JsonPayloadGenerator` for generating JSON payloads from velocity templates and properties files.

SchemaHelper

The `SchemaHelper` class provides utility methods for handling REST & SOAP schemas. It includes functions for retrieving schema file paths, parsing and modifying SOAP messages, and transforming them into strings.

TestDataConfig

The `TestDataConfig` class is a Spring component that manages test data properties. It loads properties from `.properties` files located in the `src/test/resources/testdata` directory during initialization. This class provides methods for adding and retrieving values from additional test data, as well as accessing the loaded properties.

Token

The `Token` class is a Spring component responsible for retrieving authentication tokens. It utilizes `RestAssured` for HTTP requests and `JSONPath` for parsing JSON responses. This class is configured with properties such as the token base URL, client ID, and client secret, which are injected via Spring's `@Value` annotation. Additionally, it depends on the `ApiPropertiesContext` class for environment configuration. The `getAuthToken` method sends a request to the token endpoint, authenticates with client credentials, and retrieves the access token for authorization.

Validations

The `Validations` class is a Spring component that provides various validation methods for testing RESTful and SOAP APIs. It includes functionalities for validating response codes, field values in responses, JSON and SOAP schemas, error messages, and date, integer, string, and long field values.

Dynamic payload

JsonPayloadGenerator

The `JsonPayloadGenerator` class is a component responsible for generating JSON payloads based on Velocity templates and test data properties. It utilizes the Apache Velocity Engine for template processing. This class includes methods for generating JSON payloads using Velocity templates and populating them with test data properties. Additionally, it handles the loading of properties through the `TestDataConfig` class and provides options for generating payloads with additional test data.

ADO Integration

AdoManager

The `AdoManager` class is responsible for managing interactions with Azure DevOps, such as retrieving test plan and test point IDs, creating test runs, and updating test results. It utilizes the `AzureDevOpsUrlBuilder` class to construct URLs for different Azure DevOps endpoints and the `HttpUtils` class to make HTTP requests.

AzureDevOpsUrlBuilder

The `AzureDevOpsUrlBuilder` class is responsible for constructing URLs and payloads used in Azure

DevOps API requests. It provides methods for building URLs for various Azure DevOps endpoints such as test plans, test points, test runs, and test results. Additionally, it offers functionality to generate payloads for creating test runs and updating test results.

Config

The `Config` class is a Spring configuration class responsible for defining beans and properties needed for the Azure DevOps integration module. It utilizes Spring's annotation-based configuration to define beans for various URL templates and property values required by the Azure DevOps API.

HttpUtils

The `HttpUtils` class provides utility methods for making HTTP requests using Apache HttpClient.

HttpRequest

The `HttpRequest` class is a simple Java bean that represents an HTTP request.

API Object

SRequest

`SRequest` is a Spring component responsible for making HTTP requests using RestAssured in a Mulesoft API environment. It supports POST and GET requests, along with SOAP requests. The class handles constructing requests, setting headers, and managing request payloads, providing a streamlined interface for interacting with the API.

ResponseHolder

`ResponseHolder` is a Spring component designed to store HTTP responses obtained from API requests. It provides methods to set and retrieve response objects, allowing for easy access to response data across different parts of the application. This class serves as a convenient way to manage and pass around response data within the Mulesoft API environment.

Runner

Hook

The `Hook` class contains methods annotated with `@Before` and `@After`, which serve as hooks for setup and teardown actions before and after each scenario in Cucumber tests. These hooks handle tasks such as logging scenario execution, updating Azure DevOps (ADO) test results, and managing resources. Additionally, the class initializes necessary components and registers a shutdown hook to perform cleanup actions upon application termination.

CommonSteps

CommonSteps

The `CommonSteps` class defines step definitions for Cucumber scenarios related to API testing. It includes methods for creating and sending REST and SOAP requests, validating response status codes and error messages, verifying REST and SOAP contracts, executing database queries, and validating query results. Additionally, it makes use of various utility classes for payload creation, response handling, validation, and JSON processing.