Zaplet User Guide

Table of Contents

Introduction

Zaplet is a cross-platform command-line utility written in C++20, designed for testing REST APIs and creating/replaying test scenarios. Zaplet allows you to perform various types of HTTP requests, configure headers, send data, and analyze responses in a convenient format.

Key features of Zaplet:

- Executing HTTP requests (GET, POST, PUT, DELETE, PATCH, HEAD, OPTIONS)
- Replaying test scenarios from YAML files
- Support for variables in scenarios
- Conditional step execution
- Flexible response validation system
- Convenient output formatting (JSON, YAML, table view)
- Configurable logging system

Basic Usage

Zaplet provides a command-line interface for executing HTTP requests and running scenarios. The general command format:

```
zaplet <command> [options]
```

To get help on available commands:

```
zaplet --help
```

HTTP Requests

Zaplet supports all major HTTP methods. Below are examples of using each of them.

GET Request

```
zaplet get https://api.example.com/users
```

Additional parameters:

- `-H, --header` - add an HTTP header (can be specified multiple times)
- `-t, --timeout` - request timeout in seconds (default 30)

Example with an authorization header:

```
zaplet get https://api.example.com/users -H "Authorization: Bearer token123"
```

POST Request

```
zaplet post https://api.example.com/users -d '{"name": "John", "email":
"john@example.com"}'
```

Additional parameters:

- `-H, --header` - add an HTTP header
- `-d, --data` - data to send in the request body
- `--content-type` - content type (default "application/json")
- `-t, --timeout` - request timeout in seconds

PUT Request

```
zaplet put https://api.example.com/users/1 -d '{"name": "John Updated",
"email": "john@example.com"}'
```

Parameters are similar to the POST request.

DELETE Request

```
zaplet delete https://api.example.com/users/1
```

Parameters:

- `-H, --header` - add an HTTP header
- `-t, --timeout` - request timeout in seconds

PATCH Request

```
zaplet patch https://api.example.com/users/1 -d '{"name": "John Patched"}'
```

Parameters are similar to the POST request.

HEAD Request

```
zaplet head https://api.example.com/users
```

Parameters:

- `-H, --header` - add an HTTP header
- `-t, --timeout` - request timeout in seconds

OPTIONS Request

```
zaplet options https://api.example.com/users
```

Parameters:

- `-H, --header` - add an HTTP header
- `-t, --timeout` - request timeout in seconds

Output Formatting

By default, Zaplet formats the output in YAML. You can change the output format using the global `--format` option as follows:

```
zaplet --format json get https://api.example.com/users
```

Supported formats:

- `json` - JSON
- `yaml` - YAML (default)
- `table` - table view

Working with Scenarios

Zaplet allows you to execute previously created API testing scenarios. Scenarios are stored in files with the `.zpl` extension and allow you to run a sequence of HTTP requests with a single command.

Detailed documentation on creating scenarios can be found in the separate file `scenario_writing_guide.md`.

Running a Scenario

To run a scenario, use the `play` command:

```
zaplet play my_scenario.zpl
```

With variables (overriding variables from the scenario file):

```
zaplet play my_scenario.zpl -v base_url=https://api.staging.example.com -v
auth_token=test_token
```

Variables passed through the command line take precedence over variables defined in the scenario file.

Advanced Features

Request Headers

To add HTTP headers to a request, use the `-H` or `--header` option:

```
zaplet get https://api.example.com/users -H "Authorization: Bearer token" -H
"Accept: application/json"
```

Timeouts

The request timeout is specified in seconds:

```
zaplet get https://api.example.com/users -t 60
```

Logging

Zaplet supports flexible logging with configuration options.

Logging Configuration

The logging configuration is located in the `config/logger.conf` file:

```
[general]
name = zaplet
level = info
pattern = [%Y-%m-%d %H:%M:%S.%e] [%n] [%l] [thread %t] %v
async = false
async_queue_size = 8192
async_thread_count = 1

[console]
enabled = true

[file]
enabled = true
path = logs/zaplet.log
rotating = true
max_size = 10485760
max_files = 10
daily = false
rotation_hour = 0
rotation_minute = 0

[tcp]
enabled = false
host = 127.0.0.1
port = 9000

[udp]
enabled = false
host = 127.0.0.1
port = 9001
```

Available logging levels:

- **trace**: the most detailed level
- **debug**: debug messages
- **info**: information messages (default)
- **warning**: warnings
- **error**: errors
- **fatal**: critical errors
- **off**: logging disabled

Troubleshooting

Common Problems and Solutions

1. **Problem**: "Invalid URL" error
   Solution: Make sure the URL starts with `http://` or `https://`

2. **Problem**: "Connection refused" error
   Solution: Check the server availability and URL correctness

3. **Problem**: "Failed to parse file" error
   Solution: Check the YAML syntax in the scenario file

4. **Problem**: "Scenario file does not exist" error
   Solution: Check the path to the scenario file and its existence

Debugging and Logging

For more detailed information about errors, change the logging level to `debug` or `trace` in the `config/logger.conf` file.

Note: This guide covers the basic capabilities of Zaplet. For additional information about using HTTP requests and running scenarios, use the `zaplet --help` command. Detailed documentation on creating scenarios can be found in the `scenario_writing_en.pdf` file.