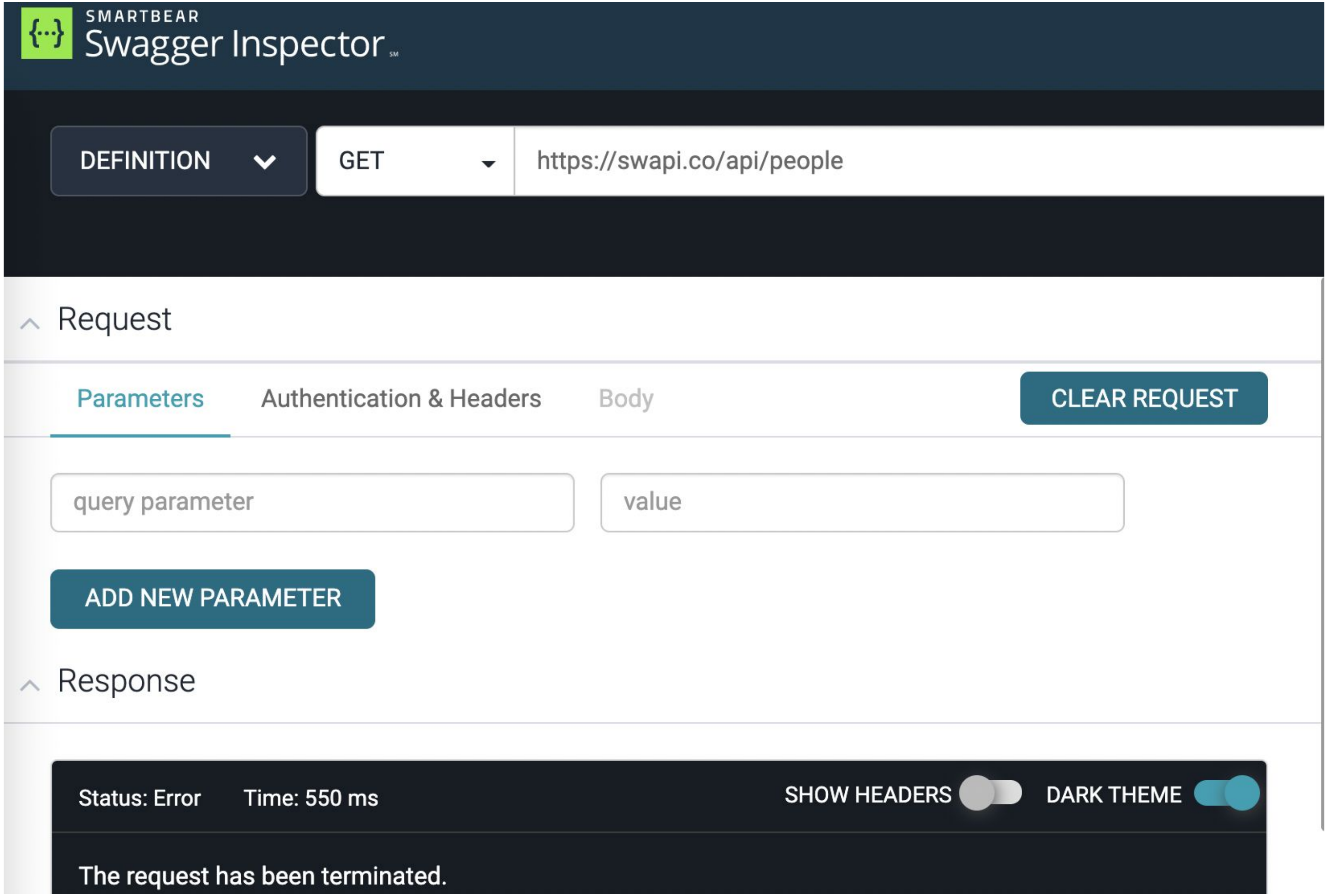
Test generation

API-based testing

- Application Programming Interface: abstraction for system design
- Standard representations for APIs
  - OpenAPI, Swagger etc.
- Can they also generate test cases?



GET https://swapi.co/api/people

Status: Error

Time: 550 ms

The request has been terminated.

Use cases

- Import API definition from standard like OpenAPI
- Generate tests for specific endpoints, scenarios
- Record API traffic
- Inject possible problem cases based on known techniques
- Data validation tests

Abstract Tests

- Semi-formal verbal description:
    - Make a request to ‘/’ endpoint
    - Ensure that result contains text “Hello world”

Abstract Tests

- Semi-formal verbal description:
  - Make a request to ‘/’ endpoint
  - Ensure that result contains text “Hello world”

```python
def test_hello(client):
    """Verify home page."""

    rv = client.get('/')
    assert b'Hello world' in rv.data
```

Model-based testing

Example: Authenticate user before showing information

- Scenarios:
  - User already logged in - page shown
  - User not yet logged in - redirect to login page
  - Forgot password - after resetting, come back to desired page
- Model:
  - Possible states (logged in, password reset, ...)
  - Possible transitions
  - Generate tests for the possible transitions

Models and Abstract Tests

- Abstract tests apply to generic models
- Create model for system-under-test
- Derive “executable” tests by combining abstract test information with model

(G)UI testing

- User interface: visual output
- Usually GUI - even for web-based systems
  - But specific details of graphical display may be different in web-based systems
- Tests:
  - Are specific elements present on page
  - Are navigation links present
  - What happens on random click on some part of the page

Browser automation

- Some tests cannot be directly run programmatically
  - Browser is **required**, just requests not sufficient
- Example:
  - IRCTC or SBI website - captcha protected
  - Some user input also required - cannot be completely automated
- Request generation:
  - Python requests library
  - Capybara (ruby), ...
- Direct browser automation:
  - Selenium framework - actually instantiate a browser

Security testing

- Generate invalid inputs to test app behaviour
- Try to crash server - overload, injection etc.
- Black-box or White-box approaches
- **Fuzzing** or Fuzz-testing:
  - Generate large number of random/semi-random inputs