Testing Advanced Scenarios

Marko Vajs
Software Development Engineer in Test

Module Overview

Learn about commands and plugins and how to define custom commands

Examine Cypress's ability to make and intercept HTTP requests

See what lifecycle hooks are and discover common use cases

Use Cypress CLI to run tests

Commands and Plugins

Commands and Plugins

Commands

Commands are functions that perform specific tasks.

Plugins

Plugins enable you to tap into, modify, or extend Cypress's internal behavior.

Commands

support

commands.js

```
// ***********************************************
// This example commands.js shows you how to
// create various custom commands and overwrite
// existing commands.
// ***********************************************
```

index.js

```
import './commands'
```

Types of Commands

Parent

`.visit(), .get()`

Child

`.click(), .type(), .should()`

Dual

`.contains()`

Defining a Custom Command

```
// parent command
Cypress.Commands.add('addComment', (name: string, comment: string) => {
       cy.get('#name').type(name);
       cy.get('#comment-text').type(comment);
});

// child command
Cypress.Commands.add('addComment', {prevSubject: true}, (name: string, comment: string) => {
       cy.get('#name').type(name);
       cy.get('#comment-text').type(comment);
});

// dual command
Cypress.Commands.add('addComment', {prevSubject: 'optional'},
 (name: string, comment: string) => {
       cy.get('#name').type(name);
       cy.get('#comment-text').type(comment);
});
```

Commands

Defining and using a custom command

support/commands.js

```
Cypress.Commands.add('addComment', (name:
string, comment: string) => {
        cy.get('#name').type(name);
        cy.get('#comment-text').type(comment);
});
```

integration/adventure.spec.js

```
cy.addComment('Josh',
    'What a great adventure!');
```

Overwriting a Command

```
Cypress.Commands.overwrite('type', (originalFn, element, text, options) => {
    if (options && options.sensitive) {
        options.log = false;
    }

   return originalFn(element, text, options);
});
```

Plugins

plugins

index.js

```js
// Plugins enable you to tap into, modify, or extend the internal behavior of Cypress
// For more info, visit https://on.cypress.io/plugins-api

module.exports = (on, config) => {}
```

Plugins

Defining and using your own plugin

plugins/index.js

```
module.exports = (on, config) => {
    on('task', {
      log(message) {
        console.log(message);
        return null;
      },
    });
}
```

integration/adventure.spec.js

```
cy.task('log', 'This will be output.');
```

When to use this feature?

- Seeding a database
- Querying a database
- Overriding the default Cypress's configuration

Demo

Creating a custom command

Making HTTP Requests

With Cypress, the CORS check is bypassed.

```
cy.request('http://localhost:3000');

cy.visit('http://localhost:3000/adventures');
cy.request('adventures/1'); // URL is http://localhost:3000/adventures/1

cy.request('DELETE', 'http://localhost:3000/adventures/1');

cy.request('POST', 'http://localhost:3000/adventures', { title: 'New Adventure' });

cy.request({
    method: 'POST',
    url: 'http://localhost:3000/adventures',
    body: { title: 'New Adventure' },
    headers: {'Content-Type': 'application/json' }
});
```

```
cy.request('POST', 'http://localhost:3000/adventures', { name: 'New Adventure' })
 .then((response) => {
      expect(response.status).to.eq(201);
      expect(response.body).to.have.property('name', 'New Event');
  }
);
```

When to Use Requests?

Log in

Get data

Create data

Demo

Making HTTP requests

Using Hooks

```
describe('Hooks', () => {
   before(() => {
      // runs once before all tests in the block
   });

   beforeEach(() => {
      // runs before each test in the block
   });

   afterEach(() => {
      // runs after each test in the block
   });

   after(() => {
      // runs once after all tests in the block
   });
})
```

It is suggested to do the clean-up before and not after tests.

Demo

Using hooks

Intercepting HTTP Requests

Read Change Block

network requests

When to use this feature?

- If a server-side implementation is not ready
- If you intend only to test a presentation layer

Demo

Intercepting HTTP requests

Demo

Using Cypress CLI to run tests

Summary

Summary

Identify main business scenarios

Identify steps to automate through the UI

Use App Actions for supporting steps

Automate flows that behave consistently

Use precise and future-proof selectors

Keep your test project well-organized

Cypress Documentation

https://docs.cypress.io