How to Integrate Winston Logger into Playwright Cucumber Framework

---

1. Create Logger Utility

- **Create folder:** `src/helper/utils/`
- **Create file:** `logger.ts`
- **Add this code:**

```
import { transports, format } from 'winston';

export function options(scenarioName: string) {
  return {
    transports: [
      new transports.File({
        filename: `test-results/logs/${scenarioName}/log.log`,
        level: 'info',
        format: format.combine(
          format.timestamp({ format: 'MMM-DD-YYYY HH:mm:ss' }),
          format.align(),
          format.printf(info => `${info.level}: ${info.timestamp}:
${info.message}`)
        )
      })
    ]
  };
}
```

Explanation:

- Creates a separate log file **per scenario** at `test-results/logs/{scenarioName}/log.log`
- Logs include **timestamp** and are formatted for better readability

---

2. Update `pageFixture.ts`

- Add a **logger property** alongside the page object:

```
import { Page } from "@playwright/test";
import { Logger } from "winston";

export const pageFixture = {
  //@ts-ignore
  page: undefined as Page,
  logger: undefined as Logger
};
```

---

3. Install Winston

Run this command in your project terminal:

```
npm install winston --save-dev
```

---

4. Configure Logger in `hooks.ts`

- Initialize the logger **before each scenario** with a unique scenario name:

```
import { createLogger } from 'winston';
import { options } from '../helper/utils/logger';
import { pageFixture } from '../fixture/pageFixture'; // Adjust path if
needed

Before(async function({ pickle }) {
  const scenarioName = `${pickle.name}_${pickle.id}`; // Unique per
scenario
  const context = await browser.newContext();
  const page = await context.newPage();

  pageFixture.page = page;
  pageFixture.logger = createLogger(options(scenarioName));
});
```

Note on `pickle`:

- `pickle.name` = Scenario name
- `pickle.id` = Unique ID to avoid duplicate log files for scenarios with the same name

---

5. Close Logger After All Scenarios

- Add this in `hooks.ts` to properly close logger and browser:

```
AfterAll(async function() {
  await browser.close();
  if (pageFixture.logger) {
    pageFixture.logger.close();
  }
});
```

- Closing the logger **prevents memory leaks** and dangling file handles

---

6. Using Logger in Step Definitions

- Use the logger anywhere in your step files like this:

```
Given("User navigates to the application", async function() {
  await pageFixture.page.goto(process.env.BASEURL as string);
  pageFixture.logger.info("User navigated to the application");
});

Given("User searches for a {string}", async function(book) {
  pageFixture.logger.info(`Searching for a book: ${book}`);
  await pageFixture.page.locator("input[type='search']").fill(book);
});
```

Recap: Key Files and Responsibilities

File	Responsibility
`utils/logger.ts`	Configures Winston logger per scenario
`pageFixture.ts`	Holds shared `page` and `logger` objects
`hooks.ts`	Creates logger before each scenario and closes it after all tests
Step definitions	Use `pageFixture.logger.info()` to log messages
