LAMBDA IT

Concept for less maintenance effort

E2E Tests with Cypress

Matthias Baldi

10.09.2020

Bärner JS Talks

LAMBDA IT

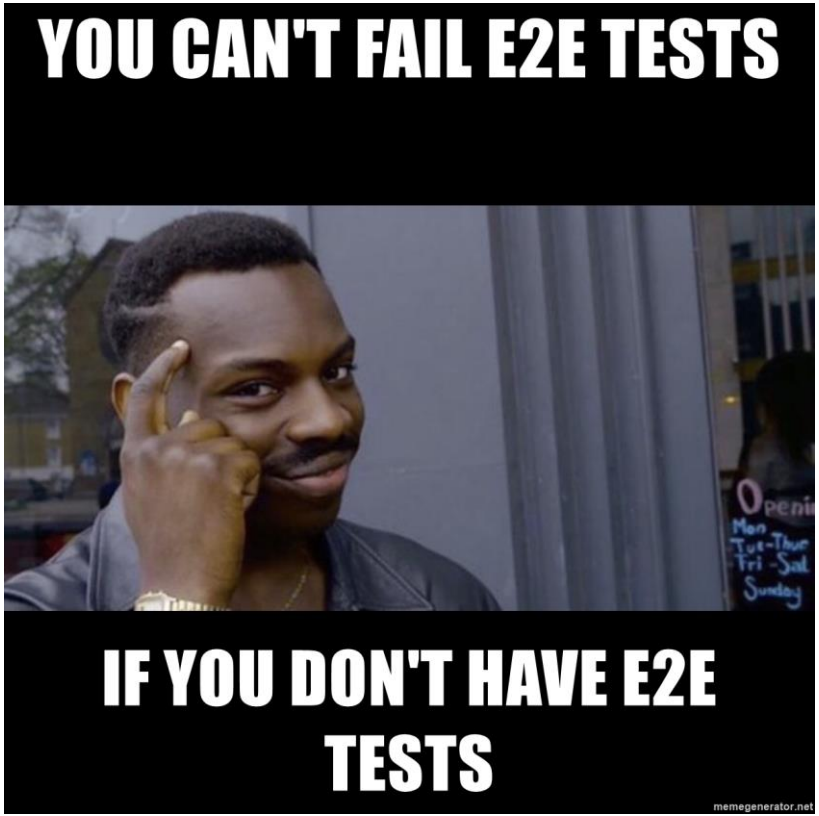
Agenda

- E2E Test Pain Points
- Used Framework
- Simple Test Case
- Concept
- Conclusion

LAMBDA IT

E2E Test Pain Points

- Bad maintainability during development
  - Selectors are changing
  - Sequence of steps is changing
- Not reproduceable on CI/CD
- Nobody knows why they run or fail
- They take too much time



Used Framework

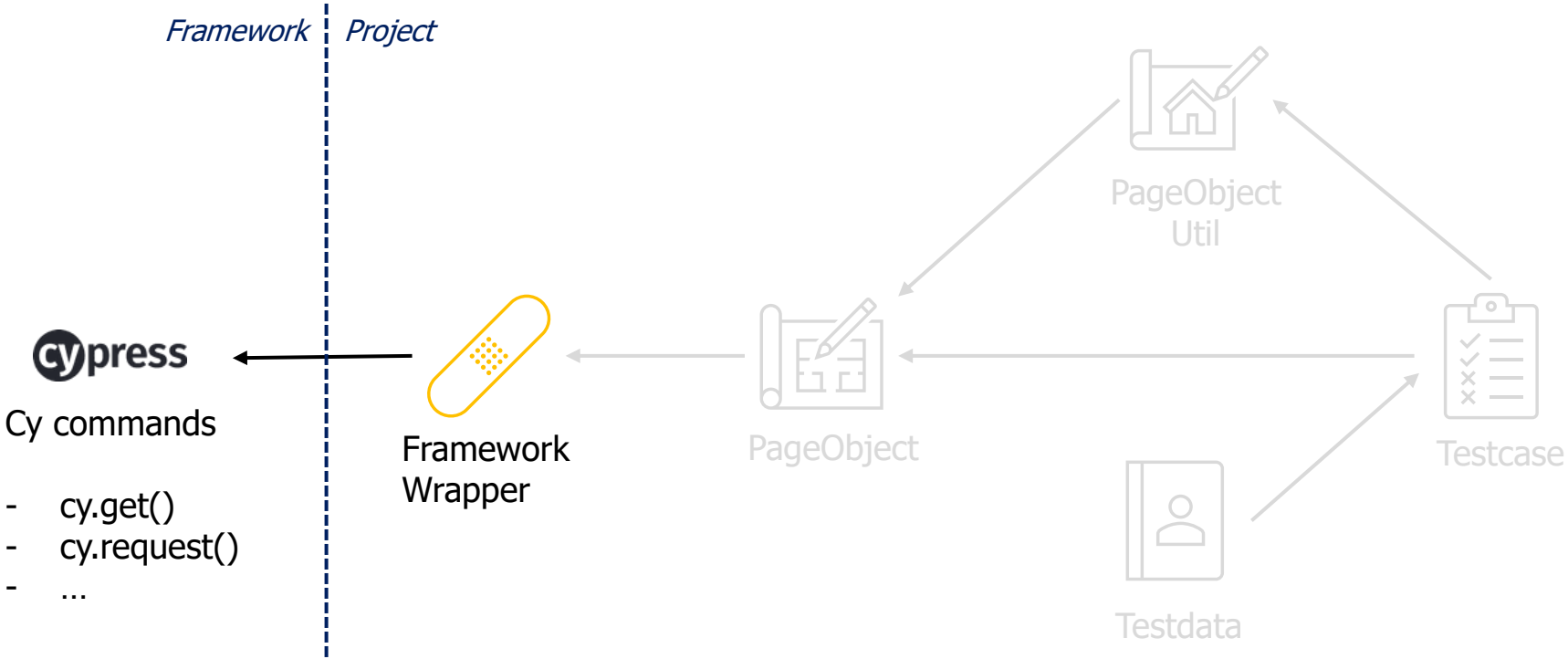
- We use Cypress as E2E framework
- Shown concepts also usable with other frameworks like Protractor

Simple Testcase

```
<form>
    <input type="text" name="firstname">
    <button data-action="submit" type="submit">
        Submit
    </button>
</form>
```


```
describe('My TestCase 01', () => {
    it('Fill form and submit', () => {
        cy.get('name["firstname"]').clear().type(person.firstname);
        cy.get('button[data-action="submit"]').click();
        ...
    });
});
```

Overview

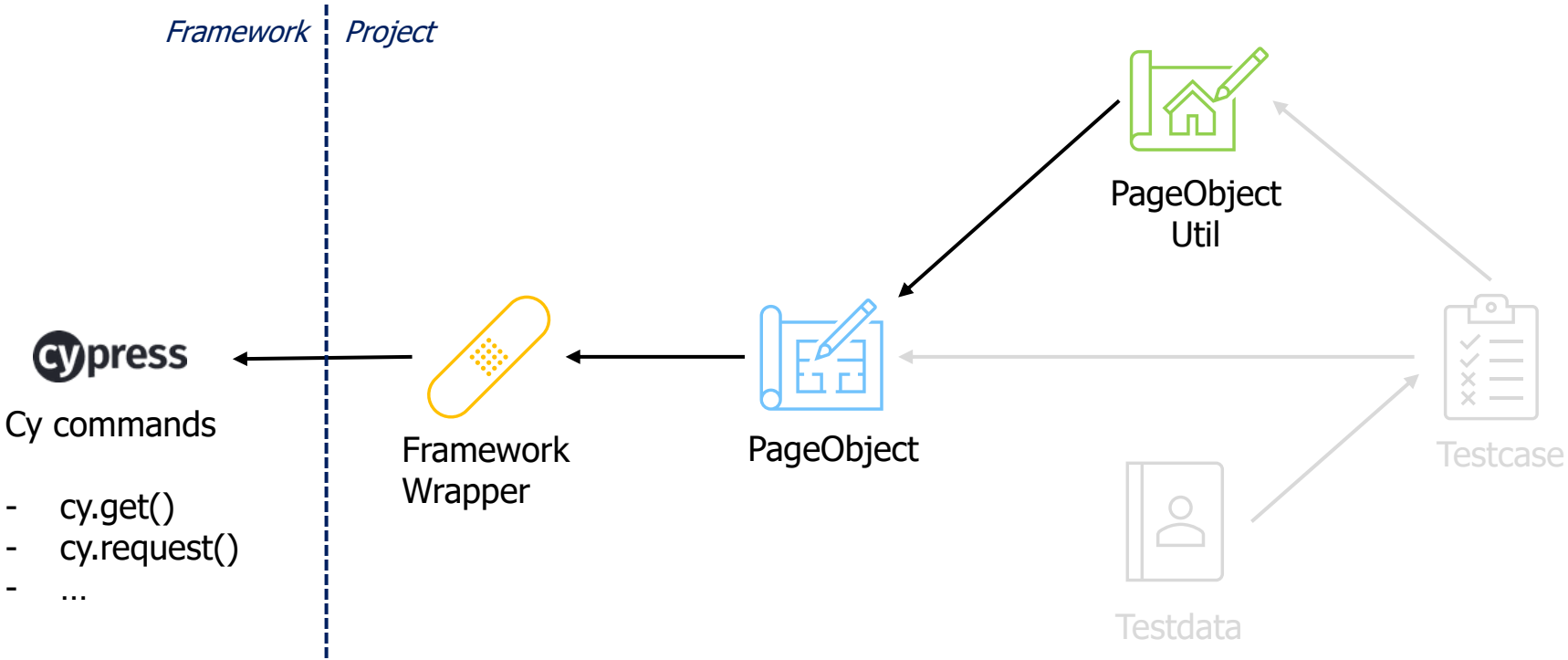

Framework
Project
cypress
Cy commands
- cy.get()
- cy.request()
- …
Framework
Wrapper
PageObject
Util
PageObject
Testcase
Testdata

Framework Wrapper

- Browser startup
- Central logging and configuration
- Fix breaking framework changes here
- May a sharable component for multiple projects

```
export function request(url: string, options) {
    cy.log(url);
    return cy.request(url, options);
}
```

Overview


Framework
Project
PageObject
Util
cypress
Cy commands
- cy.get()
- cy.request()
- …
Framework
Wrapper
PageObject
Testcase
Testdata

Page Object - Selectors

- Selectors are stored in constants
- Use when ever possible selectors like ‘name’ or ‘data-*’ attributes

```
// Component Selectors
export const REGISTER_FORM_FIRSTNAME = 'input[name="firstname"]';
export const REGISTER_BUTTON_PROCEED = 'button[data-action="proceed"]';
```

Page Object

- PO is not a Cypress concept we took this from Protractor
- Don’t represent components, represent views
- Methods for DOM operations

```
export function fillInputField(name: string, value: string) {
    // wrapped cy.get(), for framework
    Common.get(`input[name="${name}"]`).clear().type(value);
}

export function submitRegisterForm() {
    // wrapped cy.get().click()
    Common.clickButton(Selectors.BUTTONS_SUBMIT);
}
```

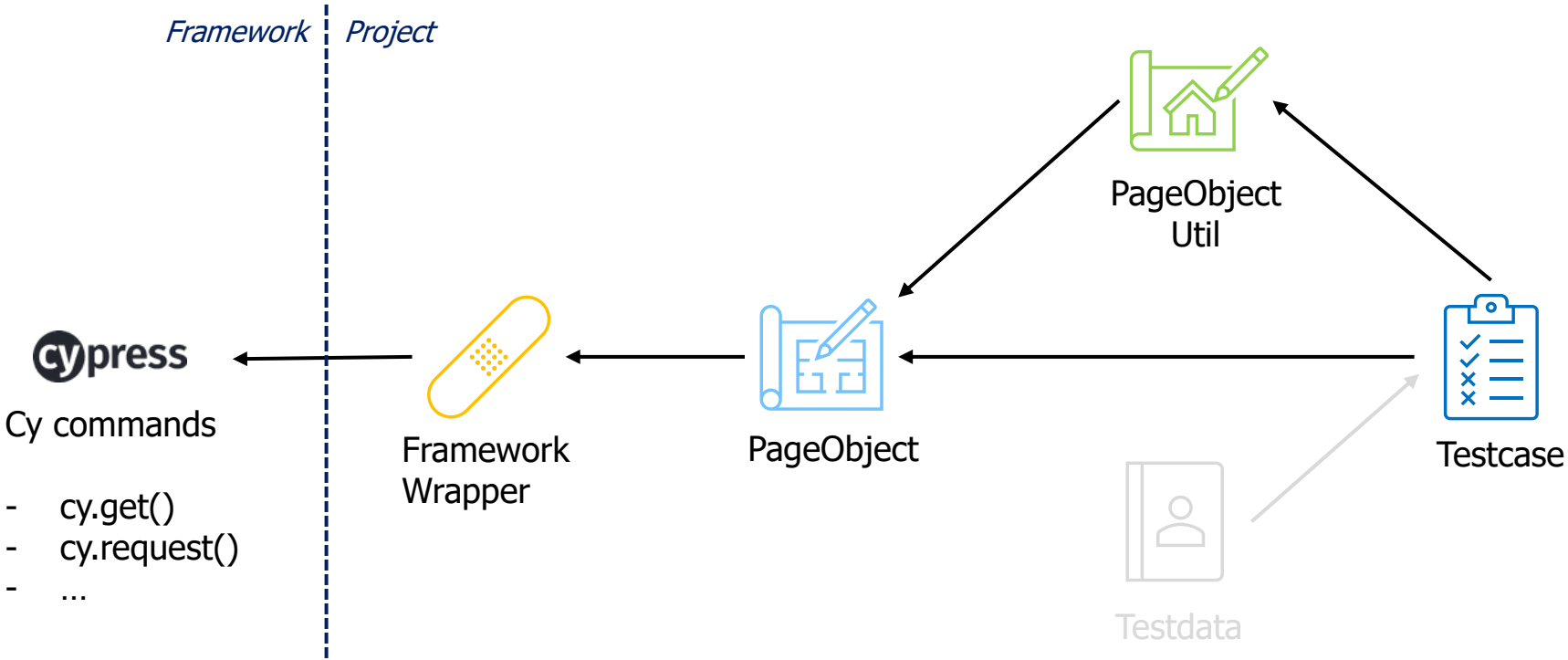
Page Object - Util

- Simplifying some corresponding steps, like filling a form and submitting it

```
// Summarize multiple redundant steps into one method
export function fillFormAndSubmit(person: Person) {
    RegisterPo.fillInputField('firstname', person.firstname);
    RegisterPo.fillInputField('lastname', person.lastname);
    RegisterPo.fillInputField('city', person.city);
    RegisterPo.fillInputField('email', person.email);

    RegisterPo.submitRegisterForm();
}
```

Overview


Framework
Project
PageObject
Util
cypress
Cy commands
- cy.get()
- cy.request()
- …
Framework
Wrapper
PageObject
Testcase
Testdata

Testcase

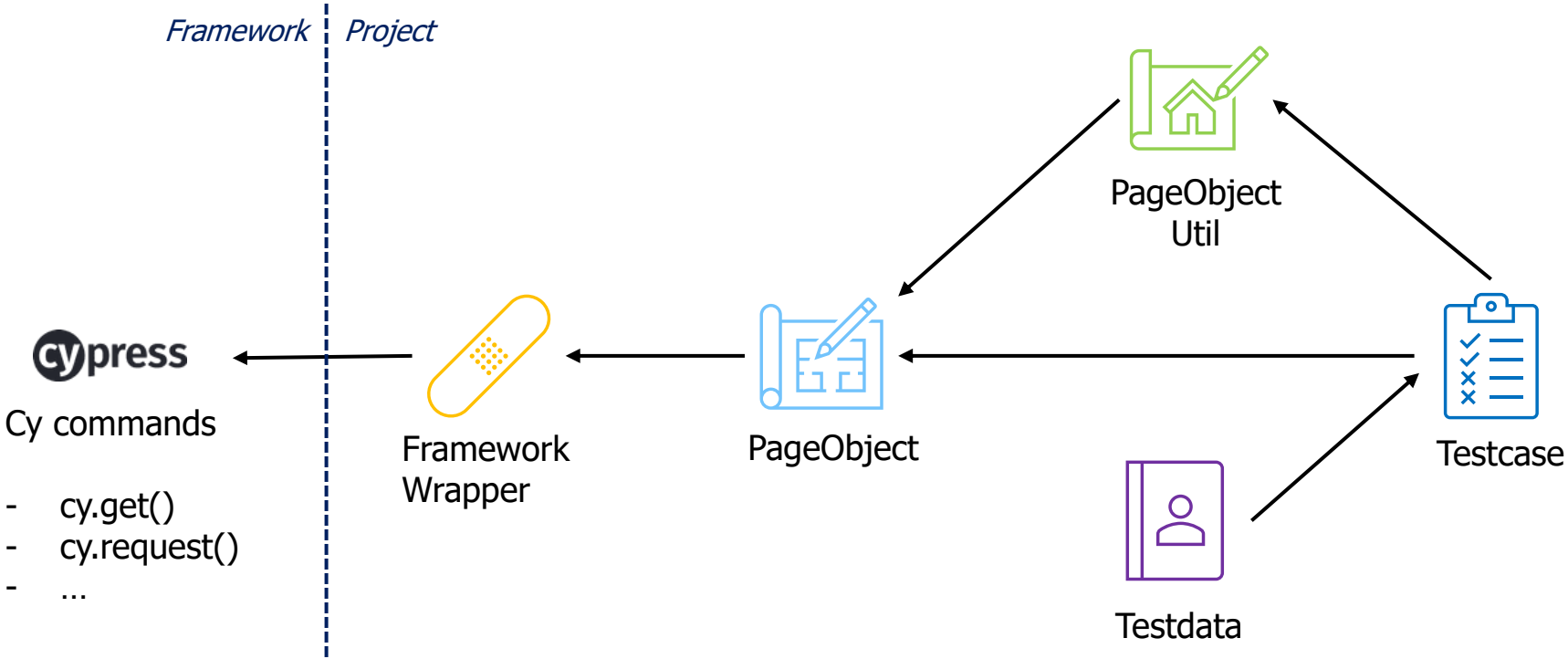
- Structure of spec files must match the project
- In our case Use-Cases or user flows
- Do not call the framework directly
- Should be easily readable (use meaningful method names)
- No framework know how needed

Testcase - Examples

```
describe('My TestCase 01', () => {
    before(() => {
       Common.open(URL, OPTIONS);
    });

    it('Step A', () => {
        DocumentRegisterUtil.fillRegisterFormAndSubmit();
        DocumentRegisterUtil.verifyResults();
        DocumentRegisterUtil.downloadAndVerifyPdf();
        ...
    });
    ...
});
```

Overview

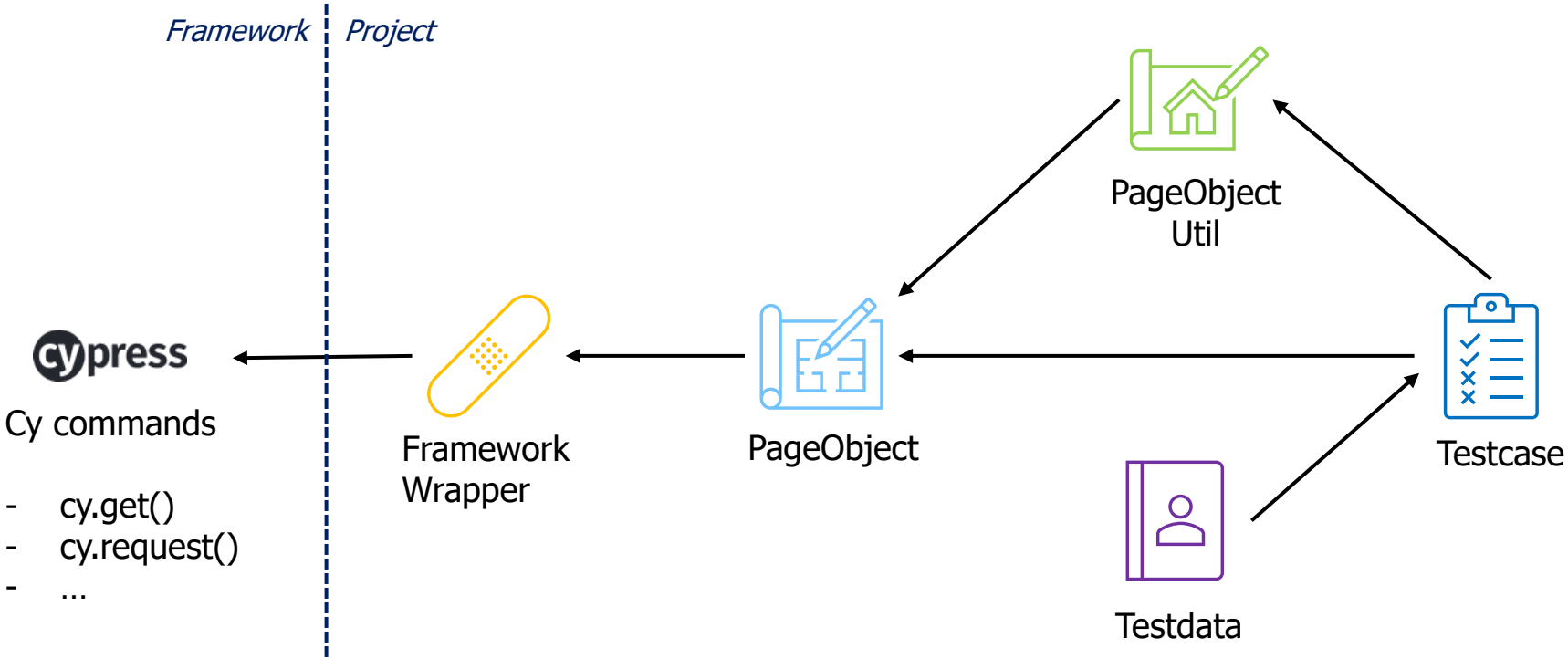

Framework
Project
PageObject Util
cypress
Cy commands
- cy.get()
- cy.request()
- …
Framework Wrapper
PageObject
Testcase
Testdata

Testdata

- Data to fill forms and input fields
- Media files to test uploads

```
// personen.json
{
    "person1": {
        "title": "Herr",
        "firstname": "Markus",
        "name": "Meier",
        "phone": "+4179xxxyyzz",
        "nationality": "Schweiz",
        "birthdate": "1975-08-12",
    },
    ...
}
```

Overview


Framework
Project
PageObject
Util
cypress
Cy commands
- cy.get()
- cy.request()
- …
Framework
Wrapper
PageObject
Testcase
Testdata

```
it('CH ID', () => {
    RegisterUtil.fillFormAndSubmit(PERSON);
    DocumentRegisterUtil.skipTipWidgetAndSetLang(Language.DE);
    DocumentRegisterUtil.authByDocument(AuthDocument.chid, PERSON);
    DocumentRegisterUtil.verifyResultExecuteZekAndFinish(PERSON, true);
    DocumentRegisterPo.downloadZipDocuments(Client.F1, AuthDocument.chid, F1_URL);
});

export function authByDocument(
    scenario: AuthDocument,
    formValues: string,
    additionalSelfieCheck: boolean = false,
) {
    DocumentRegisterPo.pickScenario(scenario);
    DocumentRegisterPo.getNameAndCompare(formValues);
    DocumentRegisterPo.clickProceedButtonOrRetry('authByDocument');
    DocumentRegisterPo.verifyScanningResults(IMAGE_QUALITY_TEXT);
    DocumentRegisterPo.clickProceedButtonOrRetry('authByDocument -> scanning results');
    if (additionalSelfieCheck) DocumentRegisterUtil.checkSelfie();
}
```

Conclusion

- App changes do not affect many test files
- Framework is wrapped
  - Less know how needed
  - Logging and configuration is centrally
- Self documenting, when you use meaningful names
- Use it like a toolbox, easy start for new team members


E2E
WOW

Links

- https://docs.cypress.io/guides/references/best-practices.html
- https://lambda-it.ch/blog/post/setup-e2e-cypress-tests-angular
- https://lambda-it.ch/blog/post/best-practices-with-cypress

LAMBDA IT

Thank you!