

1.page.js

2.spec.js

PROTRACTOR

Page-object Pattern

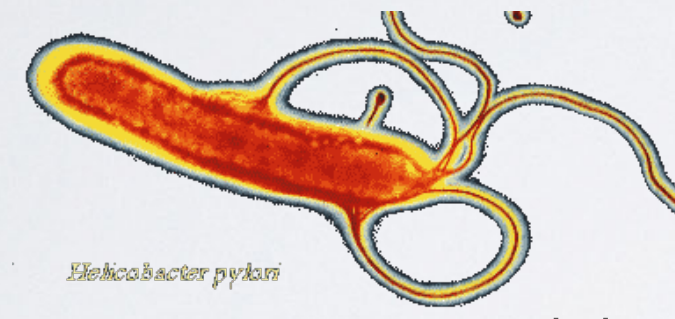
3.helper.js

4.conf.js

OBJECTIVE

1. less code duplication
2. readable code
3. maintainable code

ABSTRACTION:



Helicobacter pylori

test-object

NOT CROSS - DO NOT CROSS - DO NO

test



separation of the test-object from the test

	Test-object	Test	<u>Facebook</u>	Test-object	Test
I	single	single	home page	home.page.js	home.spec.js
2	single	multiple	profile page	profile.page.js	timeline.spec.js about.spec.js photos.spec.js
3	multiple	single	user registration	home.page.js security.check.page.js find.friends.page.js ...	registration.spec.js

FUNCTIONS

test

```
function clickSignUp() {...}

  enterFirstName: function(){ ...}
  getFirstName: function(){ ...}

  selectFemale: function(){ ...}
  isFemaleSelected: function(){ ...}

function selectMonth:(month) {...}
function getSelectedMonth() {...}

function isKeepMeLoggedInSelected() {...}
function clickKeepMeLoggedIn() {...}
```

DO NOT CROSS - DO NOT CROSS - DO NOT CROSS

```
it('should accept First Name', function(){
  HomePage.enterFirstName('Hanna');
  expect(HomePage.getFirstName()).toEqual('Hanna');
});

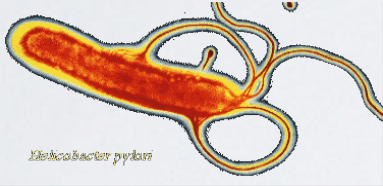


it('should select Female', function(){
  HomePage.selectFemale();
  expect(HomePage.isFemaleSelected()).toBe(true);
});

it('should check Keep me logged in', function(){
  HomePage.clickKeepMeLoggedIn();
  expect(HomePage.isKeepMeLoggedInSelected());
  .toBe(true)
});

  it('should select Female', function(){
    HomePage.selectMonth('May');
    expect(HomePage.getSelectedMonth());
    .toEqual('May')
  });

it('click Sign Up should navigate to profile page', function(){
  HomePage.clickSignUp();
  expect(browser.getTitle()).toEqual('Facebook');
  .toBe(true)
});
```


FRAMEWORK

 <p>1. Page Object</p>	<p>2. spec/test</p> 
 <p>3. config</p>	<p>4. helper</p>

I. PAGE OBJECT

```
PhonesDetails = {
  elements:{
    _name: function(){
      return element(by.binding('phone.name'));
    },

    _image: function(){
      return element(by.css('img.phone.active'));
    },

    _thumbnail: function(index){
      return element(by.css('.phone-thumbs li:nth-child(' + index + ') img'));
    }
  },

  _getName: function(){
    return this.elements._name().getText();
  },

  _getImage: function(){
    return this.elements._image().getAttribute('src');
  },

  clickThumbnail: function(index){
    this.elements._thumbnail(index).click();
  }
};

module.exports = PhonesDetails;
```

I. PAGE OBJECT

1. should contains elements and functions
2. should not contain assertions or expects
3. may represent a single page or a section in a page
4. should have a “.page” postfix (e.g. home.page.js)

I. PAGE OBJECT

do not include expects in page object, e.g

page-object

```
SomethingPage = {  
  elements: {  
    _something: function() {  
      return element(by.css('.month'));  
    },  
  },  
  
  _getSomething: function() {  
    return expect(this.elements._something().isDisplayed()).toBe(true);  
  },  
};  
  
module.exports = SomethingPage;
```

spec

```
describe('Phone list view', function() {  
  
  var somethingPage = require('../page_objects/something.page.js');  
  
  beforeEach(function() {  
    browser.get('app/index.html#/something');  
  })  
  
  it('should display something', function() {  
    somethingPage._getSomething();  
  });  
});
```



I. PAGE OBJECT

do not include expects in page object, e.g

page-object

```
SomethingPage = {  
  elements: {  
    _something: function() {  
      return element(by.css('.month'));  
    },  
  
    clickSomething: function() {  
      this.elements._something().click();  
    },  
  
    _isSomethingDisplayed: function() {  
      return this.elements._something().isDisplayed();  
    },  
  },  
};  
  
module.exports = SomethingPage;
```

spec

```
describe('Phone list view', function() {  
  
  var somethingPage = require('../page_objects/something.page.js');  
  
  beforeEach(function() {  
    browser.get('app/index.html#/something');  
  })  
  
  it('should not display something', function() {  
    expect(somethingPage._isSomethingDisplayed()).toBe(false);  
  });  
  
  it('should display something', function() {  
    somethingPage.clickSomething();  
    expect(somethingPage._isSomethingDisplayed()).toBe(false);  
  });  
});
```


I. PAGE OBJECT : SECTION

e.g. Facebook's **profile page** is a single page with multiple sections.
Even if it is a single page, dividing it into multiple page-objects is very important, else it will be one gigantic page-object and maintenance would be a nightmare.
For instance, it could be divided as follows:

profile.timeline. page.js	profile.about. page.js	profile.firnds. page.js
profile.photos. page.js	profile.places. page.js	profile.groups. page.js

2. SPEC

```
describe('Phone detail view', function(){

  var phones = require('../page_objects/phones.page.js'),
      phoneDetails;

  beforeEach(function() {
    browser.get('app/index.html#/phnes');
    phones.searchFor('nexus');
    phoneDetails = phones.selectFirstPhone();
  });

  it('should display nexus-s page', function() {
    expect(phoneDetails._getName()).toBe('Nexus S');
  });

  it('should display the first phone image as the main phone image', function() {
    expect(phoneDetails._getImage()).toMatch(/img\\/phones\\/nexus-s.0.jpg/);
  });

  it('should swap main image if a thumbnail image is clicked on', function() {
    phoneDetails.clickThumbnail(3);
    expect(phoneDetails._getImage()).toMatch(/img\\/phones\\/nexus-s.2.jpg/);

    phoneDetails.clickThumbnail(1);
    expect(phoneDetails._getImage()).toMatch(/img\\/phones\\/nexus-s.0.jpg/);
  });
});
```

2. SPEC

1. should only contains assertions and expect
2. should not contain function or element declaration
3. should only access web elements using page-object functions
4. should have a “.spec” postfix (e.g. home.spec.js)
5. reference to a page-object should have a “Page” postfix (e.g var Home**Page** = require('../page_objects/home.page.js))

2. SPEC

Spec should only access web elements using page-object functions



3. HELPER

1. should contain test data
2. should contain common scripts
3. should contain constants
4. should contain anything that is repeated or consumed multiple times (regex, login/logout, requires, ...)

3. HELPER

when something change you only have to change one thing, simplifies code maintenance.

e.g. test data changes, page added/removed, date format changed, ...

3. CONFIG

1. should contain framework configuration
2. should contain device/platform/browser/hosted testing service specific configurations
3. should contain CI configuration
4. should contain configuration parameters

END

Conclusion: this pattern is not Protractor specific. Can be used in any UI automation frameworks.

References: <https://github.com/xgirma>

TODO: researching actionAgent , more on helpers and config

Questions, comments and suggestions