

FRAMEWORK ANGULAR

Utilisation de Framework

Notion de composant web

Concept d'Angular

Tests

ANGULAR : TEST

Possibilité de créer et exécuter :

- des tests unitaires grâce à **Karma** et **Jasmine**.
- des tests end-to-end avec **Protractor** (anciennement) ou **Cypress**.

ANGULAR : TEST

Test unitaire :

→ Pour les exécuter avec Angular CLI : `ng test`.

Framework **Jasmine** :

- Ils sont rédigés en TypeScript,
- Utilisation du *task-runner* Karma
- Configuration au niveau des fichiers *karma.conf.js* et *app/test.ts*

ANGULAR : TEST

→ Récent problème lors de l'exécution de la commande `ng test` (début 2022):

Présence d'une boucle infinie dans un des module npm Colors (version 1.4.2)

→ Boucle infinie insérée par le développeur

```
undefined
undefined
undefined
```

$$\text{H}|\text{H}|\text{H}|\text{H}|\text{H}$$


ANGULAR : TEST

Framework **Jasmine** :

→ Basé sur le *BDD* (*Behaviour Driven Development*).

Concepts en commun :

- **Describe** : définit le titre du test et donne la fonction contenant les diverses spécifications.
- **It** : possède le titre d'une spécification et contient les tests validant cette spécification (assertions).
- **Expect** : assertion entre deux valeurs, se basant sur un matcher (ex: la valeur A DOIT être égale à la valeur B).
- **Matcher** : aides mise à disposition pour rédiger les assertions (toEqual, contains, etc.).

ANGULAR : TEST

Couple expect – matcher :

→ Large panel de matcher mis à disposition :

- toBe(), toBeNull(), toEqual(), toBeUndefined(), toBeFalsy(), etc.
- toThrow(), toThrowError(), etc.
- toContain()
- toBeGreaterThan()

→ Négation : expect(value).not.toEqual(otherValue)

→ Possibilité de créer des matchers personnalisés

ANGULAR : TEST

→ Test sur une fonction :

```
function additionner(a: number, b: number) {  
  return a+b;  
}
```

→ Créer un fichier `.spec.ts` avec le test suivant :

```
describe('Tests des méthodes manipulant des nombres', () => {  
  it("La méthode devrait additionner deux valeurs", () => {  
    let a = 1;  
    let b = 2;  
    let result = additionner(a, b);  
    expect(result).toEqual(3);  
  })  
});
```

Karma v 6.3.10 - connected; test: complete;

 Jasmine 3.8.0

.....

Ran 1 of 5 specs - run all

1 spec, 0 failures, randomized with seed 85775


Tests des méthodes manipulant des nombres
• La méthode devrait additionner deux valeurs

ANGULAR : TEST

→ Si le test est incorrect :

Karma v 6.3.10 - connected; test: complete; DEBUG

Chrome 96.0.4664.110 (Windows 10) is idle

 **Jasmine** 3.8.0 Options

5 specs, 1 failure, randomized with seed 54715

finished in 0.138s

Spec List | Failures

Tests des méthodes manipulant des nombres > La méthode devrait additionner deux valeurs

Expected -1 to equal 3.

Error: Expected -1 to equal 3.
at <Jasmine>
at UserContext.<anonymous> (http://localhost:9876/_karma_webpack_/webpack:/src/app/first/first.co
at ZoneDelegate.invoke (http://localhost:9876/_karma_webpack_/webpack:/node_modules/zone.js/fesm2
at ProxyZoneSpec.onInvoke (http://localhost:9876/_karma_webpack_/webpack:/node_modules/zone.js/fe

ANGULAR : TEST

- Dans une méthode `describe`, plusieurs méthodes `it` peuvent être déclarée.
- Attention à la redondance de code lors de l'initialisation de chaque test.
- Solution : exécuter du code avant et après chaque test avec `beforeEach` et `afterEach`.

ANGULAR : TEST

```
describe('Tests des méthodes manipulant des nombres', () => {
  let value:number;
  beforeEach( () => {
    value = 1;
    //autres initialisations de données
  })
  beforeEach(async () => {
    //mot clé async à importer de @angular/core/testing
  })
  it("La méthode devrait additionner deux valeurs", () => {
    let a = value;
    let b = 2;
    let result = additionner(a, b);
    expect(result).toEqual(3);
  })
  afterEach(()=>{
  })
});
```

ANGULAR : TEST

Test sur les composants :

- Besoin d'utiliser TestBed pour tester les composants Angular :
Point d'entrée des interfaces de tests
- Configuration des tests réalisée via la méthode `configureTestingModule` de `TestBed`
- Utilisation de la méthode `createComponent` pour créer le composant (pour les tests) qui renvoi un objet de type `ComponentFixture`.

Permet de récupérer l'instance du composant, les éléments HTML du composant, etc.

ANGULAR : TEST

Récupération des données dans les méthodes beforeEach()

Récupération de l'instance du composant à partir de fixture

```
test > src > app > first > TS first.component.spec.ts > ...
1  import { ComponentFixture, TestBed } from '@angular/core/testing';
2  import { FirstComponent } from './first.component';
3
4  describe('FirstComponent', () => {
5      let component: FirstComponent;
6      let fixture: ComponentFixture<FirstComponent>;
7
8      beforeEach(async () => {
9          await TestBed.configureTestingModule({
10             declarations: [ FirstComponent ]
11         })
12             .compileComponents();
13     });
14
15     beforeEach(() => {
16         fixture = TestBed.createComponent(FirstComponent);
17         component = fixture.componentInstance;
18         fixture.detectChanges();
19     });
20 }
```

ANGULAR : TEST

→ Premier test : instantiation du composant

Dans la même fonction describe :

```
it('should create', () => {  
  expect(component).toBeTruthy();  
});
```

→ Deuxième test : utilisation des propriétés du composant

```
it('la propriété doit avoir la valeur Hello', async () => {  
  expect(component.variable).toEqual('Hello');  
})
```

ANGULAR : TEST

→ Vérifier la cohérence du rendu :

```
it('la balise div myId doit contenir la valeur de la propriété variable', async () => {  
  fixture.detectChanges();  
  expect(componentHTMLElement.querySelector('#myId')?.textContent)  
    .toContain(component.variable)  
})
```

```
beforeEach(() => {  
  fixture = TestBed.createComponent(FirstComponent);  
  component = fixture.componentInstance;  
  componentHTMLElement = fixture.debugElement.nativeElement;  
})
```

7 specs, 0 failures, randomized with seed 60542

AppComponent

- should create the app
- should have as title 'test'
- should render title

FirstComponent

- la balise div myId doit contenir la valeur de la propriété variable
- la propriété doit avoir la valeur Hello
- should create

Tests des méthodes manipulant des nombres

- La méthode devrait additionner deux valeurs

```
test > src > app > first > <> first.component.html > ...  
1  ∨ <div id="myId">  
2    |    {{variable}}  
3    </div>
```

ANGULAR : TEST

Test e2e : Tests complémentaires aux tests unitaires.

Possibilité de tester de bout en bout une fonctionnalité en définissant une suite d'interaction à réaliser.

- Jusqu'à la version **12** d'Angular :
Framework **Jasmine** pour écrire les tests,
Protractor pour exécuter les scripts (programme node.js utilisant Selenium pour permettre aux tests d'interagir avec le navigateur).
- Maintenant : utilisation d'autres librairies comme **Cypress** (basée sur **Mocha**) ou encore **WebdriverIO** ou **TestCafe**.

ANGULAR : TEST

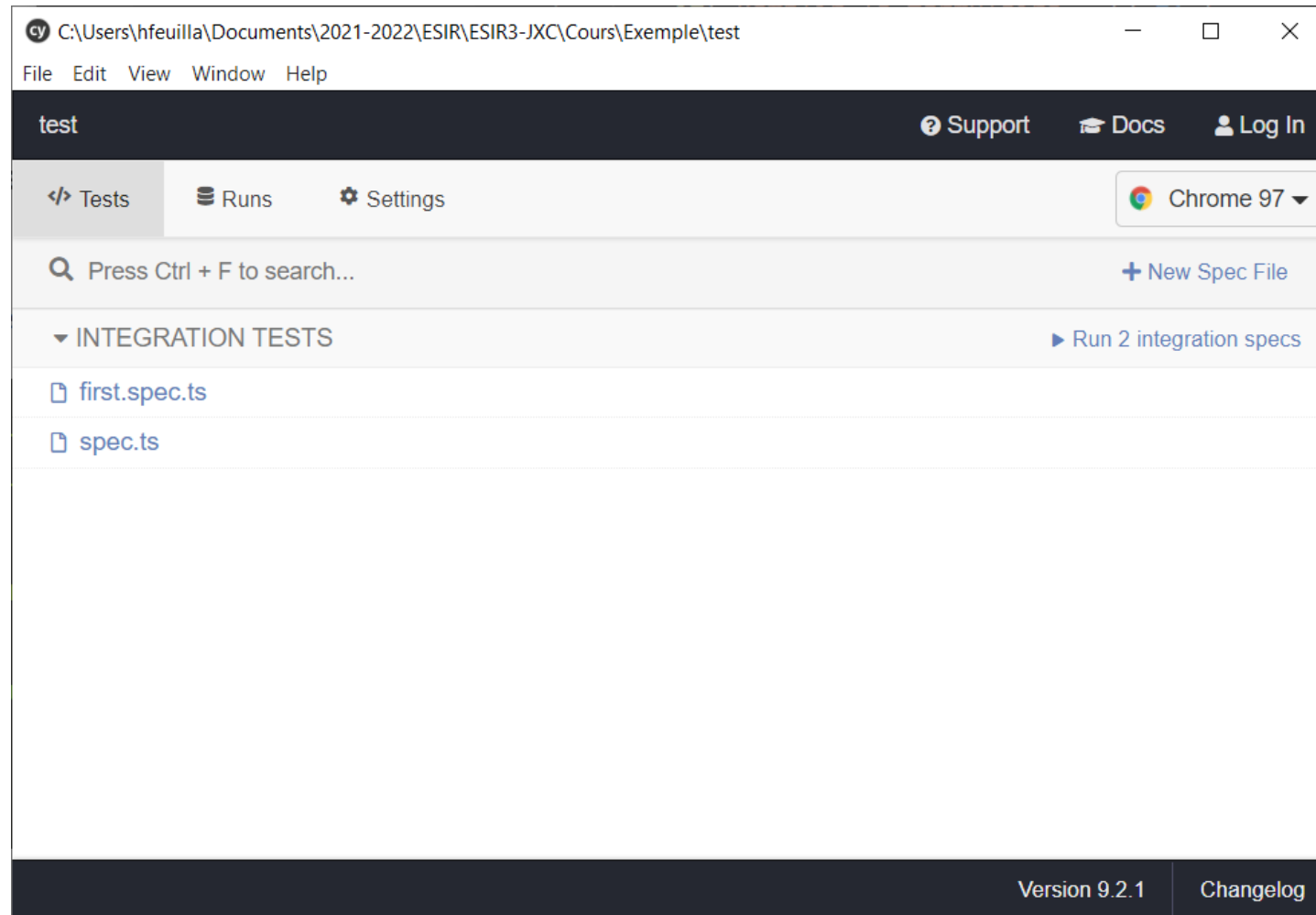
Cypress :

Installation de Cypress dans le projet Angular : `ng add @cypress/schematic`

- Tests sont structurés avec le framework **Mocha** couplé avec **Chai** pour les assertions.
- Similaire à la syntaxe de Jasmine.

- Pour exécuter les tests e2e avec CLI : `ng e2e`

ANGULAR : TEST

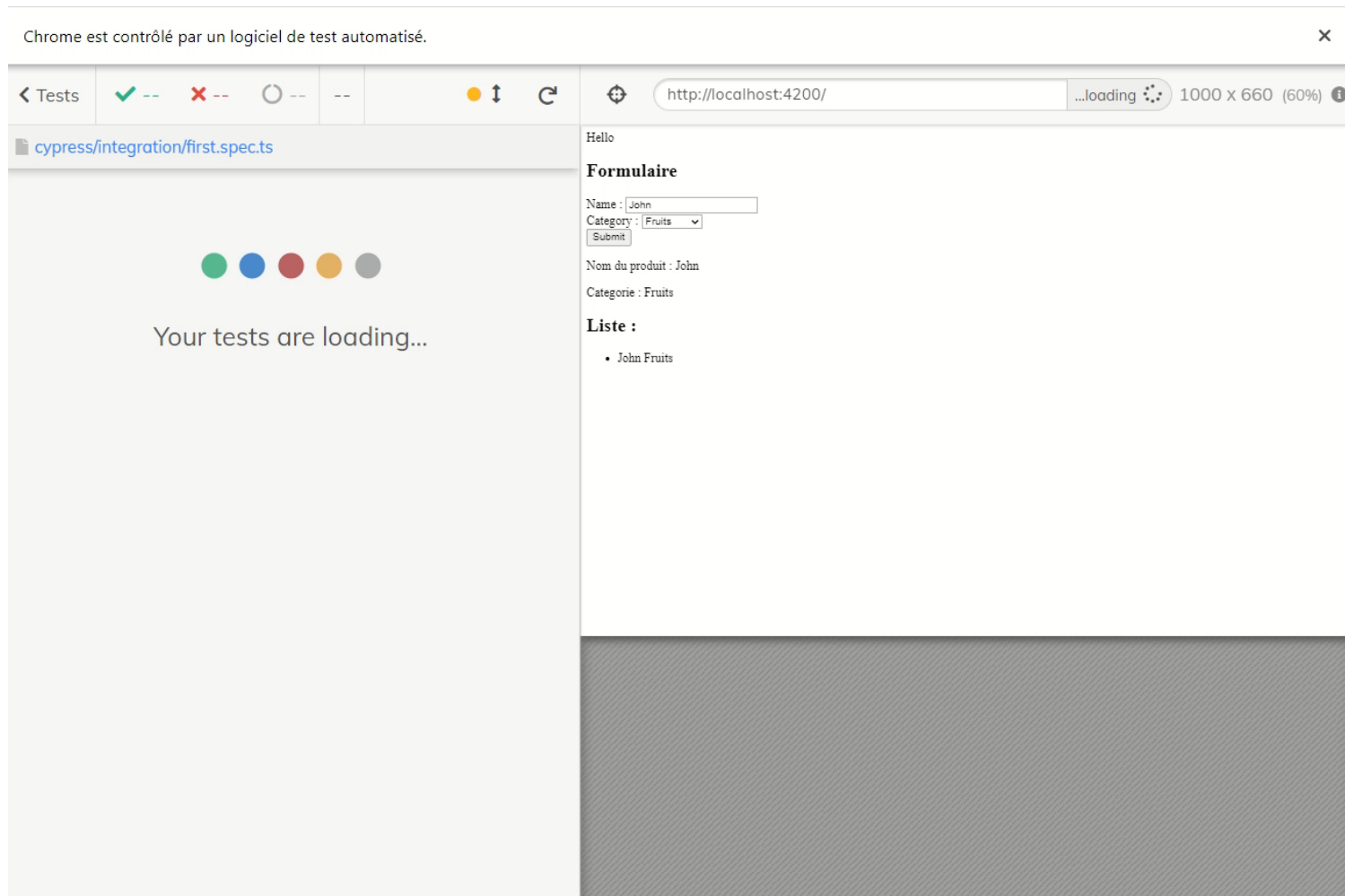


ANGULAR : TEST

The screenshot shows the Cypress test runner interface. The top bar indicates the test is running on localhost:4200. The left sidebar shows the test suite 'Premier test du formulaire' with a sub-test 'Remplir le formulaire' that has passed. The test body shows a sequence of commands: 'visit' to the root path, a GET request to a specific endpoint, and a 'get' command for the 'form' element. The right pane displays the application's state, including a 'Hello' message, a 'Formulaire' section with input fields for 'Name', 'Category', and a 'Submit' button, and a 'Liste' section. A 'DOM Snapshot' button is visible at the bottom right.

```
test > cypress > integration > TS first.spec.ts > ...  
1 describe('Premier test du formulaire', () => {  
2   it('Remplir le formulaire', () => {  
3     cy.visit('/')  
4     cy.get("form")  
5   })  
6 })
```

ANGULAR : TEST



ANGULAR : TEST

→ Code du test précédent :

```
test > cypress > integration > TS first.spec.ts > ...
 1  describe('Premier test du formulaire', () => {
 2      it('Remplir le formulaire', () => {
 3          cy.visit('/')
 4          cy.get("form")
 5          cy.get('input[name="name"]').type("John")
 6              .should("have.value", "John");
 7          cy.get('select[name="category"]').select('Fruits')
 8              .should("have.not.value", "Légumes");
 9          cy.get('button').should('be.visible')
10          cy.get('form').find('button').should('not.have.class', 'disabled')
11          cy.get('#btnSubmit').click()
12      })
13  })
```

ANGULAR : TEST

Avec :


```
test > src > app > first > <> first.component.html > form > div.group > select#category
4   <h2>Formulaire</h2>
5   <form #productForm="ngForm" (ngSubmit)="ajouterProduit()">
6     <div class="group">
7       <label for="name">Name : </label>
8       <input type="text" name="name" id="name" [formControl]="nameControl" required>
9     </div>
10    <div class="group">
11      <label for="category">Category : </label>
12      <select id="category" name="category" [formControl]="categoryControl" >
13        <option *ngFor="let cat of categories" [value]="cat">{{cat}}</option>
14      </select>
15    </div>
16    <button id="btnSubmit" type="submit" [disabled]="!productForm.valid">Submit</button>
17    <div>
18      <p>Nom du produit : {{name}}</p>
19      <p>Categorie : {{category}}</p>
20    </div>
21  </form>
22  <h2>Liste :</h2>
23  <div id="Liste">
24    <ul>
25      <li *ngFor="let prod of produits">
26        {{ prod.name }} {{ prod.category }}
27      </li>
28    </ul>
```

ANGULAR : TEST

Possibilité de faire des tests visuels avec le plugin **snapshot** :

- Installation du plugin dans le projet Angular
- Enregistrer le plugin dans le fichier *cypress/plugins/index.js*
- Penser à fixer le viewport
- Fonction `cy.matchImageSnapshot()` à appeler pour réaliser le snapshot

ANGULAR :

test > cypress > snapshots > first.spec.ts >  Preview

Hello

Formulaire

Name :

Category :

Nom du produit : John

Categorie : Fruits

Liste :

- John Fruits

cy test

localhost:4200/_/#/tests/integration/first.spec.ts

Chrome est contrôlé par un logiciel de test automatisé.

Tests

✓ 1

✗ --

○ --

02.44

● ↑

↺

cypress/integration/first.spec.ts

300 x 600 (97%)

1 viewport 300, 600

▼ TEST BODY

1 visit /

(xhr) GET /sockjs-node/info?t=1641977283...

2 get form

3 get input[name="name"]

4 - type John

5 - assert expected <input#name.ng-untouched.ng-dirty.ng-valid> to have value John

6 get select[name="category"]

7 - select Fruits

8 - assert expected <select#category.ng-untouched.ng-valid.ng-dirty> not to have value Légumes

9 get button

10 - assert expected <button#btnSubmit> to be visible

11 get form

12 - find button

13 - assert expected <button#btnSubmit> not to have class disabled

14 get #btnSubmit

15 - click

16 task Matching image snapshot, Object{3}

17 - screenshot {}

18 task Recording snapshot result

http://localhost:4200/

300 x 600 (97%)

Hello

Formulaire

Name :

Category :

Nom du produit : John

Categorie : Fruits

Liste :

- John Fruits

338

ANGULAR :

The screenshot shows a Cypress test runner interface. The top bar indicates the test is running on `localhost:4200/_/#/tests/integration/first.spec.ts`. The test results show a failure at line 19 of `cypress/integration/first.spec.ts`. The error message states: "Image was 0.9806222222222222% different from saved snapshot with 2758 different pixels. See diff for details: C:\Users\hfeuilla\Documents\2021-2022\ESIR\ESIR3-JXC\Cours\Exemple\test\cypress\snapshots\first.spec.ts_diff_output_\Premier test du formulaire -- Remplir le formulaire.diff.png". The code snippet shows a task for matching image snapshots, and the error occurs in the `- then` block. The web application on the right shows a form titled "Formulaire" with fields for Name (John), Category (Fruits), and a Submit button. Below the form, it displays "Inversion", "Categorie : Fruits", and "Nom du produit : John". The page title is "Changement de titre :".

Chrome est contrôlé par un logiciel de test automatisé.

Tests: ✓ -- ✗ 1 ○ -- 01.74 ● ↑ ↺ 🔍

`cypress/integration/first.spec.ts` 300 x 600 (97%) ⓘ

15 - CLICK

```
16 task Matching image snapshot, Object{3}
17 - screenshot {}
18 task Recording snapshot result
19 - then function(){}

❌ Error

Image was 0.9806222222222222% different from saved
snapshot with 2758 different pixels.
See diff for details:
C:\Users\hfeuilla\Documents\2021-2022\ESIR\ESIR3-
JXC\Cours\Exemple\test\cypress\snapshots\first.spec.
ts_diff_output_\Premier test du formulaire --
Remplir le formulaire.diff.png

node_modules/cypress-image-snapshot/command.j
s:51:1

49 |
50 |     if (failOnSnapshotDiff) {
> 51 |         throw new Error(message);
    |         ^
52 |     } else {
53 |         Cypress.log({ message });
54 |     }

▶ View stack trace 📄 Print to console
```

Hello

Formulaire

Name :

Category :

Inversion


Categorie : Fruits

Nom du produit : John

Changement de titre :

- John Fruits

ANGULAR : TEST

> cypress > snapshots > first.spec.ts > __diff_output__ >  Premier test du formulaire -- Remplir le formul

Hello	Hello	Hello
Formulaire	Formulaire	Formulaire
Name : <input type="text" value="John"/>	Name : <input type="text" value="John"/>	Name : <input type="text" value="John"/>
Category : <input type="text" value="Fruits"/>	Category : <input type="text" value="Fruits"/>	Category : <input type="text" value="Fruits"/>
<input type="button" value="Submit"/>	<input type="button" value="Submit"/>	<input type="button" value="Submit"/>
Nom du produit : John	Nom du produit : John	Inversion
Categorie : Fruits	Categorie : Fruits	Categorie : Fruits
Liste :	Liste :	Changement de titre :
<ul style="list-style-type: none">John Fruits	<ul style="list-style-type: none">John Fruits	<ul style="list-style-type: none">John Fruits