

# UAA12 Flask TP2 - Méthodes GET et POST

## Transmettre des données par URL ou formulaires :

### 1 Test1 : Méthode POST

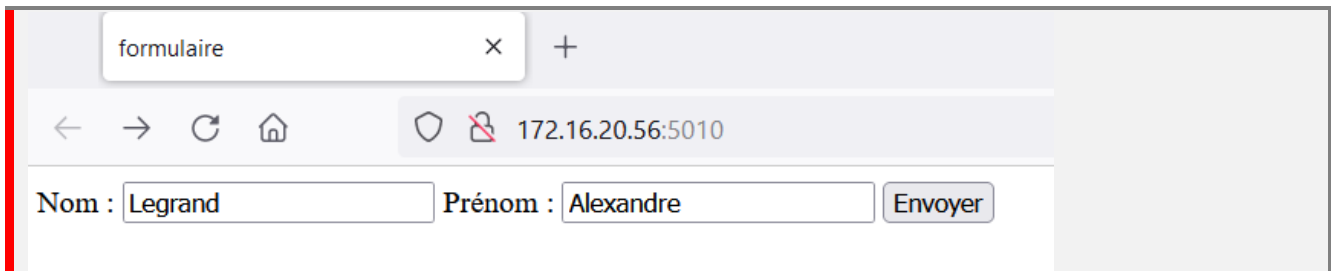
**Prérequis : savoir écrire un formulaire**

⇒ Voir le tp dans le dossier « 1\_uaa12\_html\_tp1\_form » pour un tutoriel sur la création d'un formulaire

1. Crée un dossier **tp2** à côté de tp1, copie dedans TOUT le dossier **flask\_template** et renomme-le **test1\_post**. ([12] A faire vous-même 8 & 9)

2. Crée un formulaire dans **index.html**

**index.html dans le navigateur (avec ou sans serveur)**



3. Pour cela, copie et essaie de comprendre le code suivant.

**code de index.html**

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8" />
    <title>formulaire</title>
  </head>
  <body>
    <form action="/cible" method="post">
      <p>
        <label>Nom</label> : <input type="text" name="nom_post" />
        <label>Prénom</label> : <input type="text" name="prenom_post " />
        <input type="submit" value="Envoyer" />
      </p>
    </form>
  </body>
</html>
```

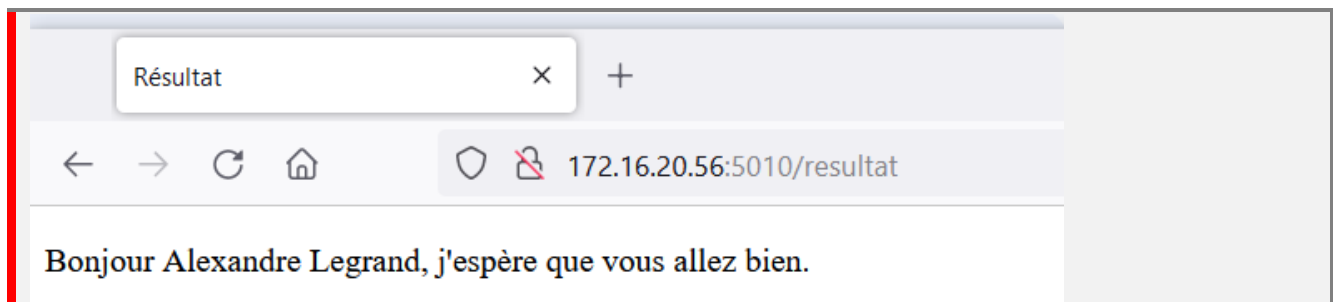
## Explications sur index.html

On remarque 2 attributs dans cette balise form : **action="/cible"** et **method="POST"**. Ces 2 attributs indiquent que le client devra effectuer une requête de type POST (méthode utilisée pour l'envoi de données par un formulaire) dès que l'utilisateur appuiera sur le bouton "Envoyer". Cette requête POST sera envoyée à l'URL "/cible" (voir l'attribut "action"). Les données saisies dans le formulaire seront envoyées au serveur par l'intermédiaire de cette requête.

4. Crée une **page cible.html** qui affiche les données entrées dans le formulaire.

Si tu saisis Legrand et Alexandre dans les champs Nom et Prénom du formulaire, tu devrais obtenir le résultat suivant après avoir appuyé sur le bouton "Envoyer"

### Cible.html dans le navigateur (avec le serveur)



Pour cela, tu peux utiliser 2 variables jinja qui ont les mêmes noms que ceux du formulaire (**nom**, **prenom**)

### Cible.html dans le navigateur (sans le serveur)

Bonjour {{prenom\_jin}} {{nom\_jin}}, j'espère que vous allez bien.

5. Complète le fichier **view.py** avec le code ci-dessous

### code de view.py

```
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/cible', methods = ['POST'])
def cible():
    result = request.form
    print("debug: ", result)
    n_py = result['nom_post']
    p_py = result['prenom_post']
    print("debug: ", n_py, p_py)
    return render_template("cible.html", nom_jin=n_py, prenom_jin =p_py)
```

## 6. Teste ton code :

- Assure-toi que tu accèdes bien aux pages index.html et cible.html par le navigateur et qu'elles sont donc bien routées (il suffit d'appuyer sur le bouton envoi même sans remplir le formulaire).
- Assure-toi que lorsque tu remplies le formulaire, son contenu (variables nom et prenom) est affiché dans le terminal (instruction "print("debug n / p:", n, p)") et que view.py a donc bien récupéré le contenu du formulaire
- Assure-toi que le contenu du formulaire s'affiche bien dans la page cible.html et donc qu'il a bien été transmis aux 2 variables jinja dans cible.html

### Explications sur view.py

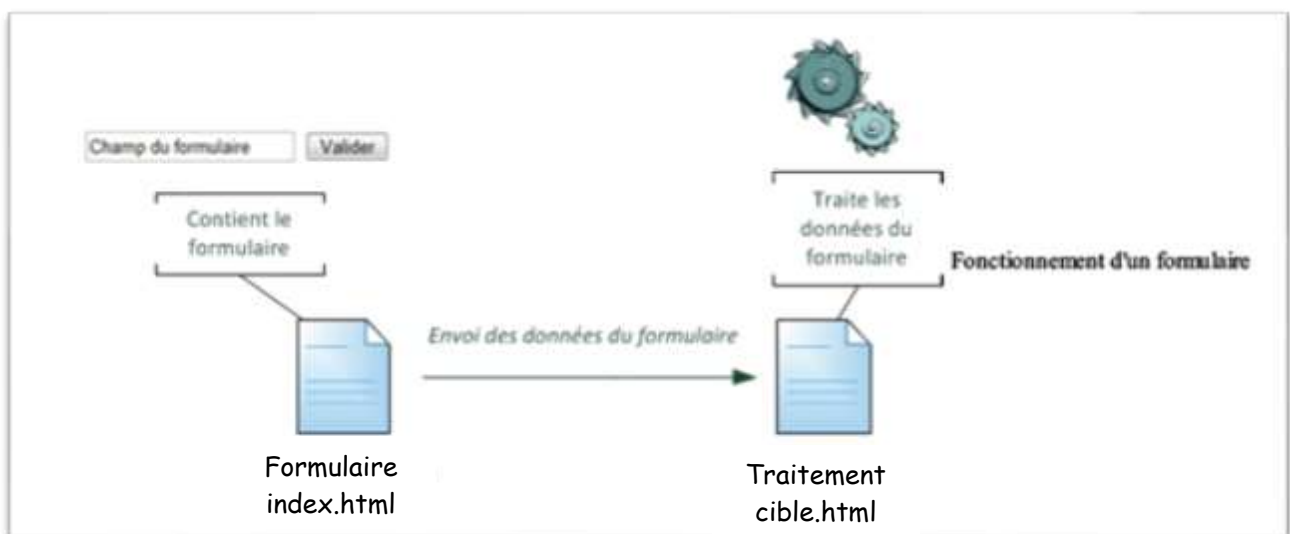
Intéressons-nous à la fonction cible(), puisque c'est cette fonction qui sera exécutée côté serveur pour traiter la requête POST :

**request.form** est un dictionnaire Python qui a pour clés les attributs name des balises input du formulaire (dans notre cas les clés sont donc "**nom\_post**" et "**prenom\_post**") et comme valeurs ce qui a été saisi par l'utilisateur. Si l'utilisateur saisit "Legrand" et "Alexandre", le dictionnaire "request.form" sera :

```
{'nom_post':Legrand, 'prenom_post':Alexandre}
```

En réponse à la requête POST, le serveur renvoie une page HTML créée à partir du template cible.html et des paramètres nom et prenom. Si l'utilisateur a saisi Legrand et Alexandre, le navigateur affichera "Bonjour Legrand Alexandre, j'espère que vous allez bien."

Les données issues du formulaire sont envoyées au serveur sans être directement visibles mais elles peuvent être lues en analysant la requête http. Seule l'utilisation du protocole sécurisé HTTPS garantit un transfert sécurisé des données entre le client et le serveur (les données sont chiffrées et donc illisibles pour une personne ne possédant pas la clé de déchiffrement).



## 2 Test2 : Méthode GET

### 1. Introduction

Fais une recherche sur le site de duckduckgo.com sur Alexandre Legrand et observe la barre d'adresse du navigateur.

```
https://duckduckgo.com/?q=alexandre+legrand ...
```

Ta requête est passée par l'URL après le " ? " et a été sauvegardée dans la variable q. Cette méthode s'appelle la **méthode GET**.

Fais une nouvelle recherche en modifiant directement la barre d'adresse, par exemple comme ceci :

```
https://duckduckgo.com/?q=annie+cordy ...
```

2. Copie TOUT le dossier `flask_template` et renomme-le `test2_get`. ([12] A faire vous-même 10, 11, 12)

3. Crée une page `index.html`

`index.html` dans le navigateur (avec ou sans serveur)



Il n'y a aucune variable jinja dans cette page.

Le lien permet de naviguer vers `naiss_age2050.html` en faisant passer les nom, prénom et âge dans l'URL par la méthode GET.

Code du lien dans `index.html`:

```
<a href="naiss_age2050?nom_get=Legrand&prenom_get=Alexandre&age_get=30" >
```

En résumé, on passe les variables par la méthode GET dans l'URL avec la syntaxe :

```
<a href="URL?nomvar1=valeur1&nomvar2=valeur2 ..." >
```

Résultat dans la barre d'adresse du navigateur lors du click sur le lien:

```
Localhost :5000/naiss_age2050?nom_get=Legrand&prenom_get=Alexandre&age_get=30
```

Contrairement à la méthodes POST, les données transmises avec la méthode GET sont directement visibles et accessibles par l'utilisateur.

4. Crée une page `naiss_age2050.html` qui affiche les données passées par la méthode GET sous formes de **variables jinja** comme vu précédemment

## naiss\_age.html dans le navigateur (avec le serveur)



5. Complète le fichier **view.py** avec le code ci-dessous

### code de view.py

```
@app.route('/')
def index():
    return render_template("index.html")
@app.route('/naiss_age2050', methods=['GET'])
def page1():
    result = request.args
    print("debug result: ", result)
    n_py= result ['nom_get ']
    p_py = result ['prenom_get ']
    a_py = result ['age_get ']
    a_py =int(a_py)
    print("debug n_py,p_py,a_py: ", n_py,p_py,a_py)
    return render_template("naiss_age2050.html",nom_jin=n_py,prenom_jin =p_py,
                           age_jin =a_py,annaiss_jin =2022-a_py,age2050_jin =2050-2022+a_py)
```

### Explications

- Le serveur s'attend à recevoir des données par la méthode GET ("methods=['GET']" précisée)
- Les données sont disponibles dans le dictionnaire request.
- Pense à afficher dans le terminal le contenu des variables reçues args :

```
print("debug result: ", result)
print("debug n_py,p_py,a_py: ", n_py,p_py,a_py)
```

- Pense à convertir la chaîne de caractères en entier pour effectuer des opérations dessus (comme pour `int(input())` en python)

6. Au lieu de fournir à la page `naiss_age2050.html` les 3 variables `age`, `annaiss`, `age2050`, tu peux ne fournir que l'âge et effectuer le calcul directement dans la page.

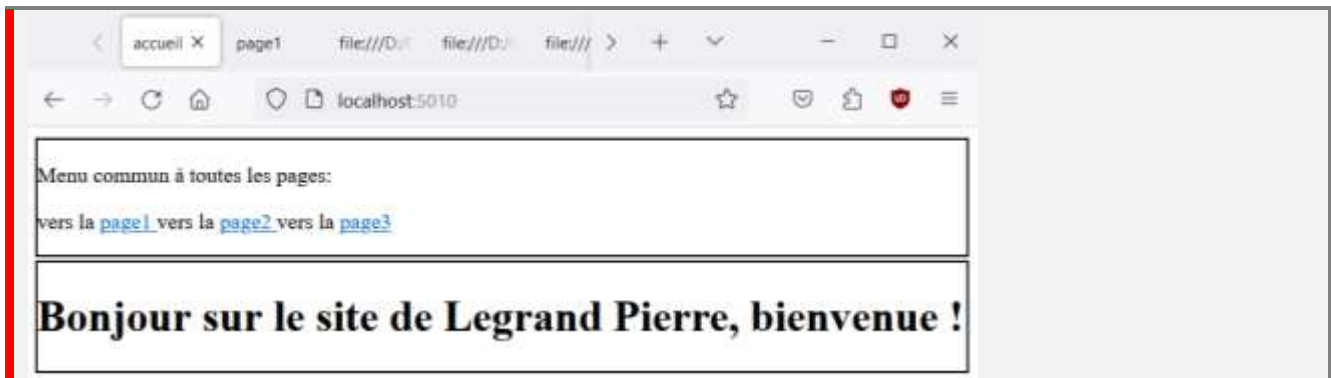
### code avec calculs dans `naiss_age2050.html`

```
Tu es né en {{ 2022 - age_jin }} et en 2050, tu auras {{ 2050 - 2022 + age_jin }} ans
```

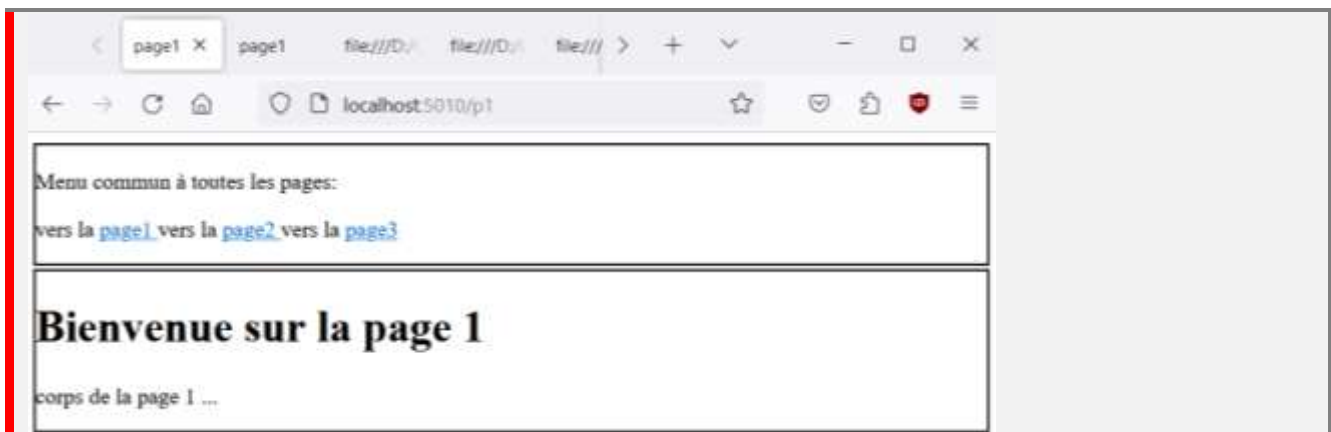
### 3 Test3 - Exercice GET: optimisation d'un site à 3 pages

1. Copie TOUT le dossier `flask_template` et renomme-le `test3_exo_get_3p`.
2. Le but est de créer comme au `tp1 / test8_site_3p` une page d'accueil et 3 pages contenant une partie commune (le menu permettant de naviguer d'une page à l'autre), et une partie propre à chacune (le message de bienvenue).
3. L'idée est de toujours afficher dans le serveur la page `index.html` (la page se rappelle elle-même), ce dernier créant la page d'accueil, les pages 1,2,3 en fonction de la variable GET fournie.

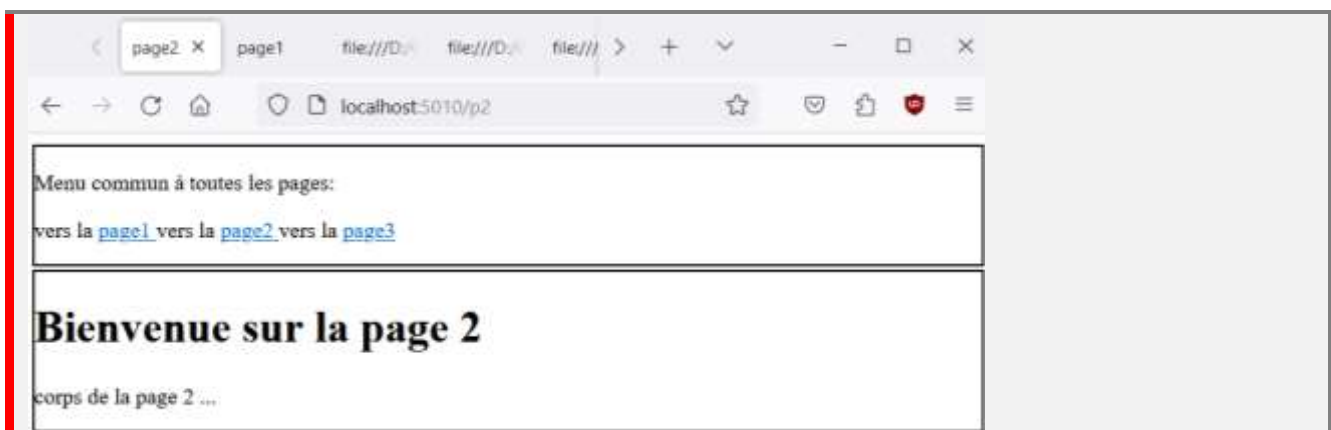
URL `Localhost:5000/` dans la barre d'adresse du navigateur



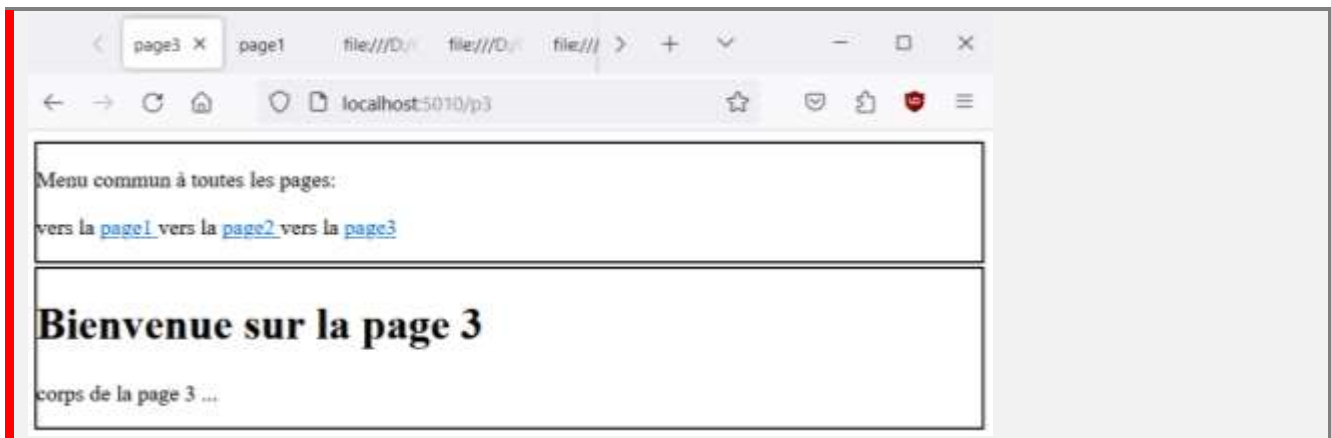
URL `Localhost:5000/?page=page1` dans la barre d'adresse du navigateur



URL `Localhost:5000/?page=page2` dans la barre d'adresse du navigateur



URL **localhost:5000/?page=page3** dans la barre d'adresse du navigateur



4. Pour cela, crée les fichiers `index.html`, `menu.html`, `accueil.html`, `page1.html`, `page2.html`, `page3.html`.
5. Le **menu** est défini dans un fichier à part. Il ne contient que le code ci-dessous (pas de `<html>`, ni de `<head>`, ni de `<body>`)

#### Code de menu.html

```
<div>
  <p> Menu commun à toutes les pages: </p>
  <p> vers la <a href="/?page=page1"> page1 </a> vers la <a href="/?page=page2"> page2 </a> vers la
    <a href="/?page=page3"> page3 </a> </p>
</div>
```

Il rappelle la page `index.html`, le numéro de la page est passée par la variable `page` par la méthode `GET`.

6. Le fichier `index.html` inclus le menu et le contenu de la page sélectionnée

#### Code d'index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8" />
  <title> {{page}} </title>
</head>
<body>
  {% include 'menu.html' %}
  {% include page+'.html' %}
</body>
</html>
```

7. Le fichier `view.py` ne génère plus qu'une page `index.html` en fonction de la variable Jinja `page` passée lors du click sur le lien.

## code de view.py

```
@app.route('/',methods=['GET','POST'])
def index():
    if request.method == 'GET':
        if 'page' in request.args:
            p=request.args['page']
        else:
            p='accueil'
        print("debug page: ",request.args, p)
        return render_template("index.html",page=p)
```

## Explications

- ✓ Attention, par défaut (tant qu'on n'a pas cliqué sur un lien, la méthode GET n'est pas encore utilisée), le serveur affiche le contenu de la page d'accueil. Il faut donc vérifier l'existence de page par la méthode GET (instruction if 'page' in request.args: ... else ...).
- ✓ Une fois que l'on click sur un lien, la variable page est fournie avec l'URL par la méthode GET et la page index.html est générée avec la variable jinja page affectée.
- ✓ Pense à afficher le contenu des variables dans le terminal.



## 4 Test4 : Exercice Post - Fiche de renseignements

Reprends le test2 dans uaa12\_html\_tp1\_form sur la fiche de renseignement et crée un fichier cible.html afin d'afficher son contenu ainsi qu'un lien pour retourner au formulaire.

Page index.html dans le navigateur:



The screenshot shows a web form titled "Fiche de renseignements" with a light blue header. The form is divided into two main sections: "Signalétique" (left, yellow background) and "Renseignements divers" (right, yellow background). In the "Signalétique" section, there are radio buttons for "Monsieur" (selected), "Madame", and "Mademoiselle". Below these are input fields for "Nom" (containing "reymondie") and "Né le" (containing "03 / 09 / 1996" with a clear button). In the "Renseignements divers" section, there is a "Mail" field (containing "reymondie.yves.info@gmail.com") and a "Pays préféré" dropdown menu (showing "France" selected, with other options like "Chine", "Espagne", "Maroc", and "Turquie" visible). At the bottom of the form are two buttons: "Annulation" and "Envoyer".

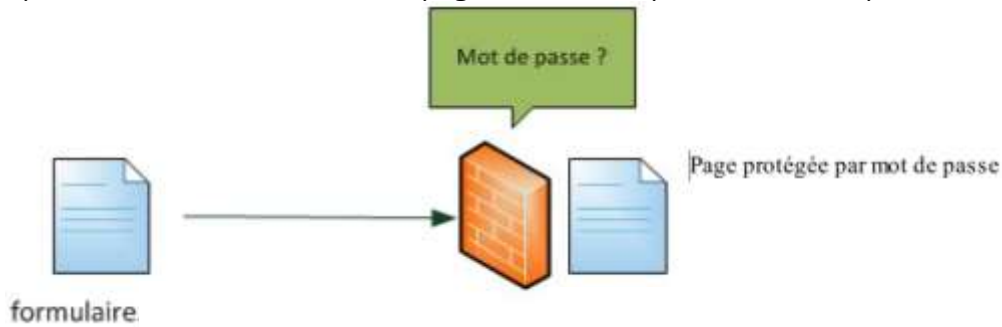
Page cible.html dans le navigateur:



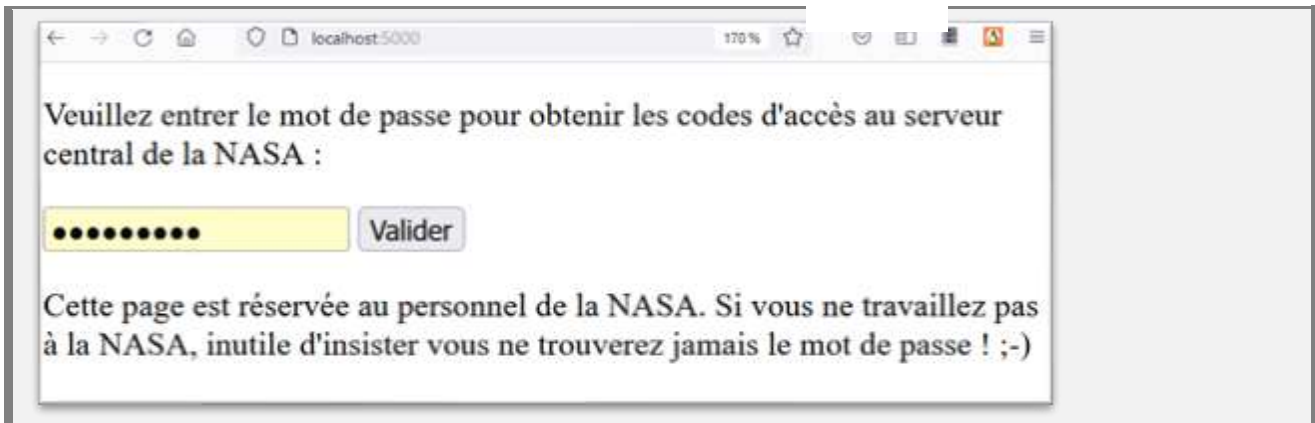
The screenshot shows a web page titled "cible.html" with a light gray background. On the left side, there is a white box containing the following text: "Nom: Mr reymondie", "né le :2021-10-06", "email: yves.reymondie@gmail.com", and "Pays préféré: maroc". Below this text is a purple link that says "Retour vers le formulaire".

## 5 Test5 : Exercice Post - Page protégée par un mode de passe

Tu veux mettre en ligne une page web pour donner des informations confidentielles à certaines personnes. L'accès à cette page est limité par un mot de passe.



Page index.html dans le navigateur:



Page cible.html dans le navigateur si le mot de passe est incorrect:



Page cible.html dans le navigateur si le mot de passe est correct:

