



# Routes et templates

# Décorateurs

# Décorateurs simples

```
utilisateur = "admin"

def acces(fonction):
    def resultat():
        print("L'accès n'est pas possible")
    if utilisateur != "admin":
        return resultat
    return fonction

@acces
def check_acces():
    print("J'ai bien accès")
```

# Décorateurs simples

```
check_acces()
```

```
utilisateur = "not_admin"

@accès
def check_acces():
    print("J'ai bien accès")

check_acces()
```

# Décorateurs avec paramètres

```
def acces(fonction):  
    def resultat(*params):  
        print("L'accès n'est pas possible pour " + params[0])  
  
        if utilisateur != "admin":  
            return resultat  
  
    return fonction
```

# Décorateurs avec paramètres

```
utilisateur = "not_admin"

@accès
def check_accès(utilisateur):
    print("J'ai bien accès en tant que " + utilisateur)
check_accès(utilisateur)

utilisateur = "admin"

@accès
def check_accès(utilisateur):
    print("J'ai bien accès en tant que " + utilisateur)
check_accès(utilisateur)
```

# Une première route

# Une route GET

```
# dans app/app.py
...

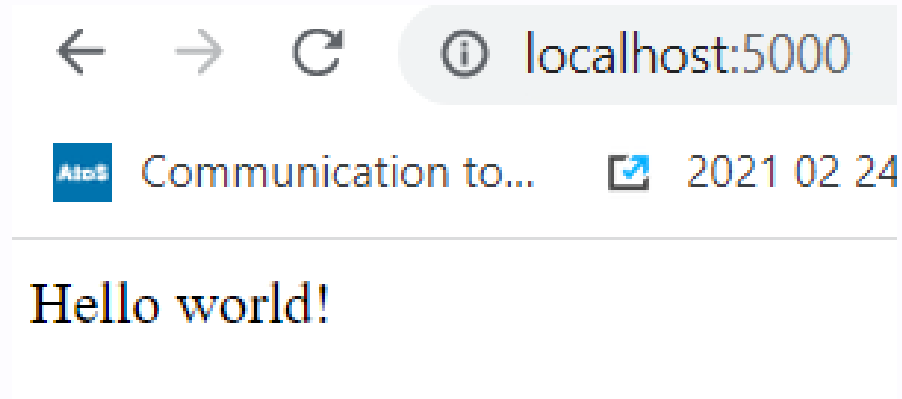
@app.route("/")
@app.route("/home")
def home():
    return "Hello world!"
```



# Une route GET

```
* Serving Flask app 'app.app'  
WARNING: This is a development server. Do not use it in a production deployment.  
* Running on http://127.0.0.1:5000  
Press CTRL+C to quit
```

# Une route GET



# Une route GET

```
* Serving Flask app 'app.app'
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [07/Jan/2023 13:17:19] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [07/Jan/2023 13:17:20] "GET /favicon.ico HTTP/1.1" 404 -
```

# Une route GET avec paramètres

# L'exemple de Wikidata

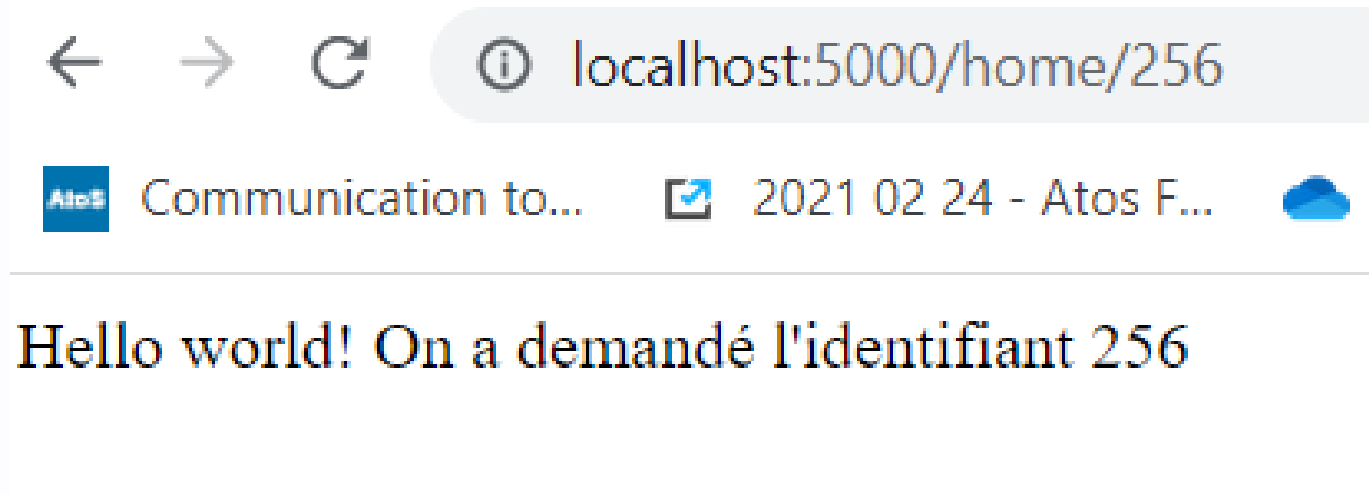
<https://www.wikidata.org/wiki/Q142>

# Définir la route

```
# app/app.py

@app.route("/home/<string:id>")
def home(id:str):
    return "Hello world! On a demandé l'identifiant " + id
```

# Définir la route



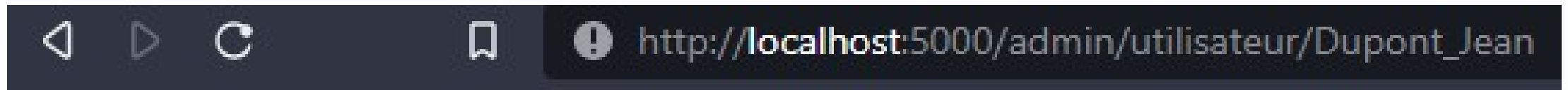
# Types de paramètres

| type   | exemple                              |
|--------|--------------------------------------|
| string | application                          |
| int    | 3                                    |
| float  | 1.23                                 |
| any    | n'importe quoi                       |
| uuid   | 123e4567-e89b-12d3-a456-426614174000 |
| path   | /chemin/vers/ressource               |



# Exercice

Développer la route qui permette d'afficher ce qui suit:



Le nom est Dupont et le prénom est Jean

# Exercice

```
# app/app.py

@app.route("/admin/utilisateur/<string:nom_prenom>")
def utilisateur(nom_prenom):
    nom_prenom = nom_prenom.split("_")
    return "Le nom est " + nom_prenom[0] + " et le prénom est " + nom_prenom[1]
```

# Templates

```
@app.route("/home")
def home():
    return '''
        <html>
            <body>
                <nav>
                    <ol>
                        <li>Lien1</li>
                        <li>Lien2</li>
                    </ol>
                </nav>
            </body>
        </html>
    '''
```

# Templates : structure de l'application

```
| app
|   |- app.py
|   |- templates
|       |- accueil.html
| run.py
```

# Construction de la route

```
from flask import render_template

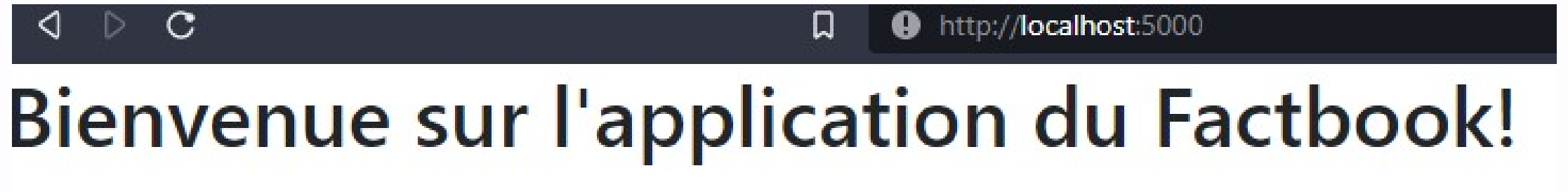
...

@app.route("/")
def accueil():
    return render_template("pages/accueil.html")
```

# Création du template

```
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Factbook</title>
  </head>
  <body>
    <h1>Bienvenue sur l'application du Factbook!</h1>
  </body>
</html>
```

# Résultat



# Une meilleure structure applicative

```
app
| app.py
| templates/
| routes/
| |-- __init__.py
| |-- generales.py
```



# Une meilleure structure applicative

```
#app.py
...
from .routes import generales
```

```
# generales.py
from ..app import app
from flask import render_template

@app.route("/")
def accueil():
    return render_template("accueil.html")
```

# Jinja2



# Transmission de paramètres

```
@app.route("/pays/<string:nom>")  
def pays(nom):  
    return render_template("pays.html", pays=nom)
```

```
<div class="container">  
    <h1>Bienvenue sur l'application du Factbook!</h1>  
    <p>Voici le lien vers le site officiel : <a  
        href="https://www.cia.gov/the-world-factbook/">  
        https://www.cia.gov/the-world-factbook/</a></p>  
    <p>Vous avez choisi d'afficher les données du pays suivant: {{pays}}</p>  
</div>
```

# Filtres

```
<div class="container">
  <h1>Bienvenue sur l'application du Factbook!</h1>
  <p>Voici le lien vers le site officiel : <a
    href="https://www.cia.gov/the-world-factbook/">
    https://www.cia.gov/the-world-factbook/</a></p>
  <p>Vous avez choisi d'afficher les données du pays suivant: {{pays}}</p>
  <p>Longueur de la chaîne de caractères du pays choisi: {{pays|length}} caractères</p>
</div>
```

# Des conditions dans Jinja

```
{% if pays|length > 10 %}  
    <p>C'est un long nom de pays</p>  
{% else %}  
    <p>C'est un nom de pays plutôt court</p>  
{% endif %}
```

# Des boucles

```
...  
return render_template("template.html", liste_villes=["Paris","Marseille"])
```

```
{%if liste_villes %}  
  <ul>  
    {%for ville in liste_villes %}  
      <li>{{ville}}</li>  
    {%endfor%}  
  </ul>  
{%else%}  
  <p>Il n'y a pas de ville</p>  
{%endif%}
```

# Les dictionnaires: rappels Python

```
dictionnaire = [  
    {"cle1" : "valeur1",  
     "cle2": 123,  
     "cle3": [1,2,3,4],  
     "cle4": {  
         "sous_cle1": "etc."  
     }  
    }  
]
```

# Les dictionnaires dans Jinja

```
{%for objet in dictionnaire %}  
  <p>Cle1 : {{objet.cle1}}</p>  
  <p>Sous_cle1 de cle4 : {{objet.cle4.sous_cle1}}</p>  
  {%for chiffre in objet.cle3 %}  
    <p>Chiffre: {{chiffre}}</p>  
  {%endfor%}  
{%endfor%}
```



# Exercice

Voici un dictionnaire:

```
[{"nom": "France",  
  "capitale": "Paris",  
  "continent": "Europe"},  
 {"nom": "Etats-Unis",  
  "capitale": "Washington",  
  "continent": "Amérique"},  
 {"nom": "Egypte",  
  "capitale": "Le Caire",  
  "continent": "Afrique"},  
 {"nom": "Chine",  
  "capitale": "Pékin",  
  "continent": "Asie"}]
```

Routes et templates

# Exercice

Obtenir le résultat suivant:



Bienvenue sur l'application du Factbook!

Voici le lien vers le site officiel : <https://www.cia.gov/the-world-factbook/>

| # | Nom        | Capitale   | Continent |
|---|------------|------------|-----------|
| 1 | France     | Paris      | Europe    |
| 2 | Etats-Unis | Washington | Amérique  |
| 3 | Egypte     | Le Caire   | Afrique   |
| 4 | Chine      | Pékin      | Asie      |

# Exercice

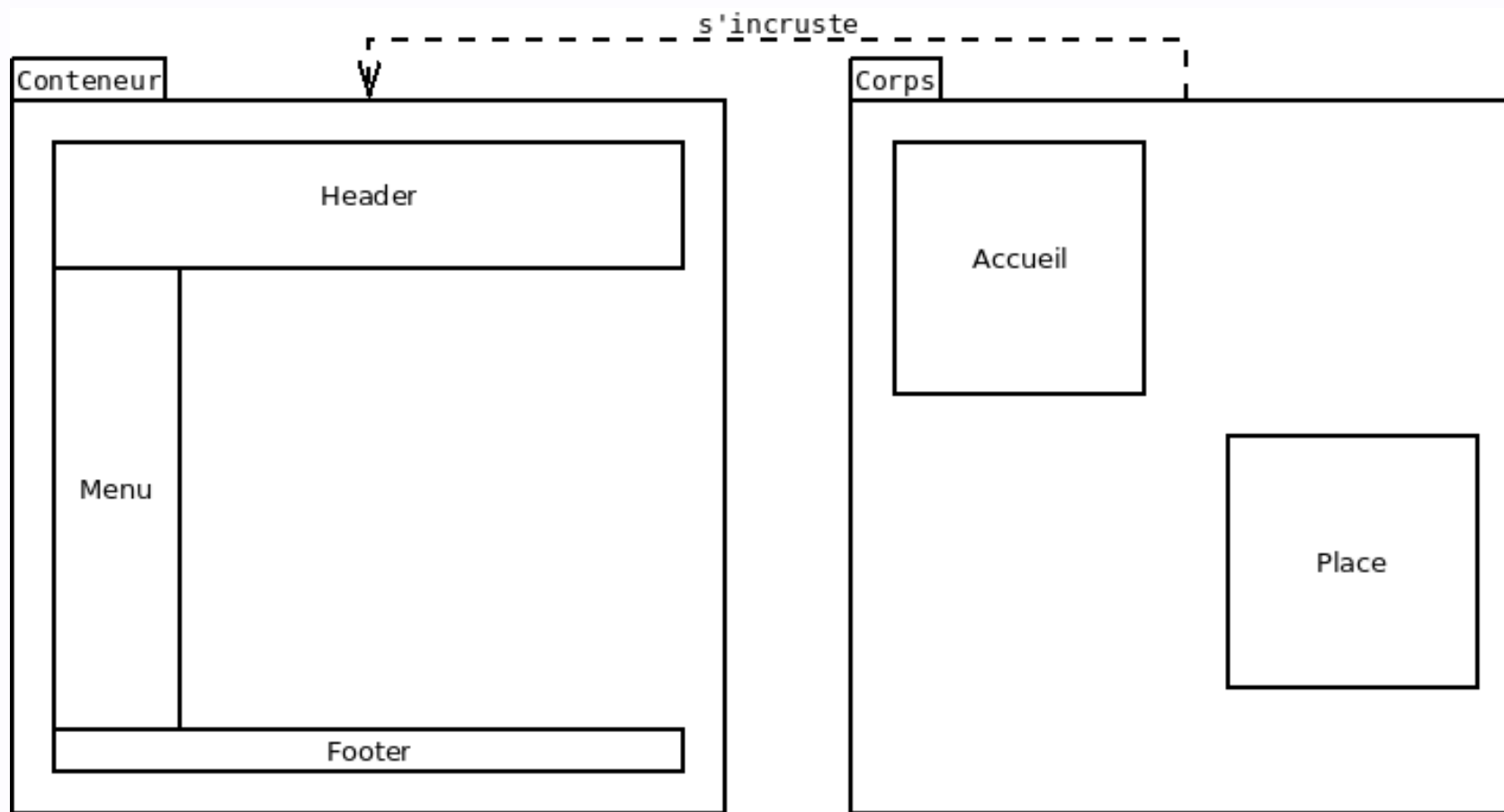
```
@app.route("/pays")
def pays():
    donnees = [{"nom": "France",
                "capitale": "Paris",
                "continent": "Europe"},
               {"nom": "Etats-Unis",
                "capitale": "Washington",
                "continent": "Amérique"},
               {"nom": "Egypte",
                "capitale": "Le Caire",
                "continent": "Afrique"},
               {"nom": "Chine",
                "capitale": "Pékin",
                "continent": "Asie"}]
    return render_template("pages/pays.html", donnees=donnees)
```

# Routes et templates

## Exercice

```
{%if donnees%}
<table class="table">
  <thead>
    <tr>
      <th scope="col">#</th>
      <th scope="col">Nom</th>
      <th scope="col">Capitale</th>
      <th scope="col">Continent</th>
    </tr>
  </thead>
  <tbody>
    {%for pays in donnees%}<tr>
      <th scope="row">{{loop.index}}</th>
      <td>{{pays.nom}}</td>
      <td>{{pays.capitale}}</td>
      <td>{{pays.continent}}</td>
    </tr>{%endfor%}
  </tbody>
</table>
{%endif%}
```

# Extension de templates



# Extends: le conteneur

```
<!-- partials/conteneur.html -->
<!doctype html>
<html lang="en">
<head>
    <title>Factbook | {{sous_titre}}</title>
</head>
<body>
    {% block body%}{%endblock%}
</body>

</html>
```

# Extends: le contenu

```
<!-- pages/pays.html -->
{% extends "partials/conteneur.html" %}

{% block body %}

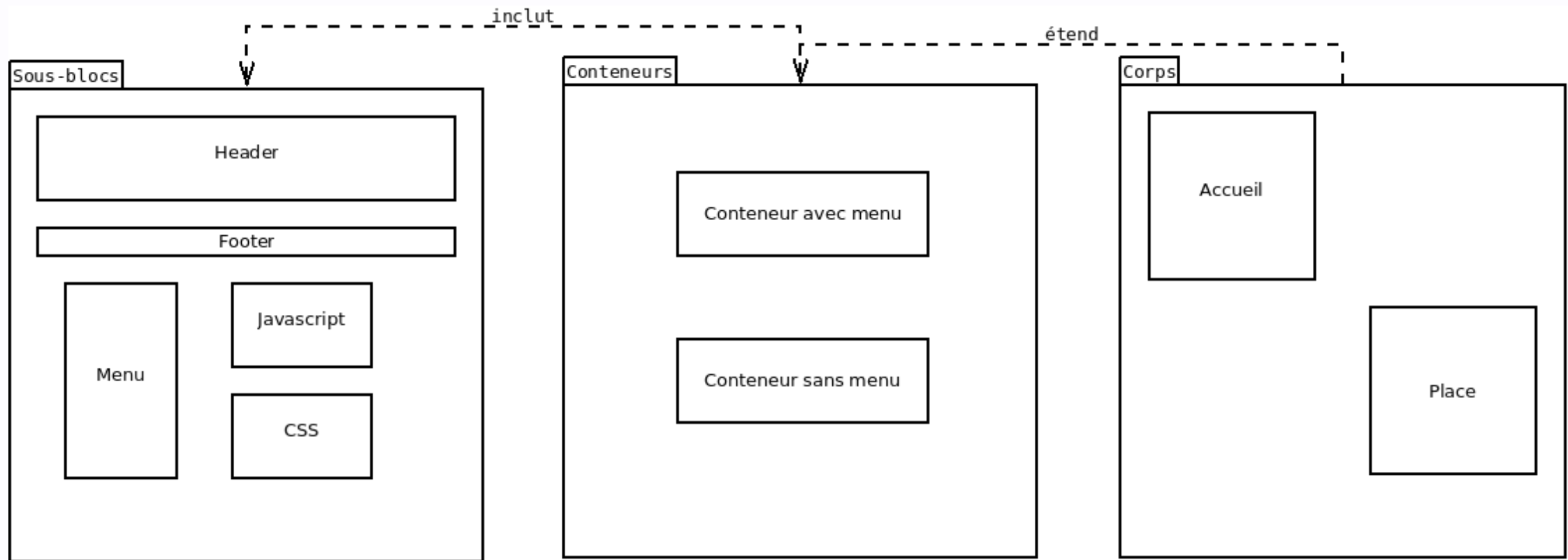
    <div class="container">
        <h1>Bienvenue sur l'application du Factbook!</h1>
        <p>Voici le lien vers le site officiel : <a
            href="https://www.cia.gov/the-world-factbook/">
            https://www.cia.gov/the-world-factbook/</a></p>
    </div>
{% endblock %}
```

# Un petit récapitulatif de l'application

```
app
| app.py
| templates/
|   |-- partials/
|   |   |-- conteneur.html
|   |-- pages/
|   |   |-- pays.html
| routes/
|   |-- __init__.py
|   |-- generales.py
```



# Inclusion de templates



# Inclusion

```
<!-- partials/conteneur.html -->
<head>
  <meta charset="UTF-8">
  <title>{%block titre %}{%endblock%} | {{sous_titre}}</title>
  {% include "partials/css.html" %}
  {% include "partials/metadata.html" %}
  {% include "partials/js.html" %}
  {% block js %}{%endblock%}
  {% block css %}{%endblock%}
</head>
```

# Liens internes

A ne pas faire !

```
<a href="localhost:5000/pays/{{nom_pays}}">{{nom_pays}}</a>
```

Utiliser `url_for()`

```
<a href="{{url_for('pays', nom=nom_pays)}}">{{nom_pays}}</a>
```

# Exercice: héritages et liens

A partir de l'exercice précédent avec le dictionnaire, adapter l'application selon les points suivants:

- utiliser les héritages Jinja (extends et include)
- sur la colonne du nom du pays, ajouter un lien vers une route `/pays/<string:nom>` qui affiche ce qui suit pour la France (et "Aucune ville connue" pour les autres pays):

# Exercice: résultat attendu



# Exercice

L'application complète se trouve dans le dépôt  
[Séance2/url\\_for](#)