Ceci est bien une **page dynamique :** à chaque fois que vous actualisez la page dans votre navigateur, l'heure courante s'affiche. En effet, à chaque fois que vous actualisez la page, vous effectuez une nouvelle requête HTTP et en réponse à cette requête, le serveur HTTP **crée et envoie une nouvelle page HTML**.

Remarque importante : Il est très important de comprendre que la page HTML envoyée par le serveur au client ne contient plus les paramètres {{heure}}, {{minute}} et {{seconde}}.
Au moment de **créer la page**, le serveur remplace ces paramètres par les valeurs passées en paramètres de la fonction "render_template". On peut s'en rendre compte en regardant le code source de la page (code source sur Chrome).
De plus, vous avez dû remarquer que lorsque vous lancez directement **index.html** dans votre navigateur, vous n'obtenez pas la date courante car le fichier HTML ne connait pas la valeur des paramètres.

Pour aller un peu plus loin : Le fichier **index.html** ne contient pas du HTML ! En effet, les paramètres {{heure}}, {{minute}} et {{seconde}} n'existent pas en HTML.
Le fichier "index.html" est écrit avec un **langage de template** nommé Jinja 2 qui ressemble beaucoup au HTML, mais rajoute des fonctionnalités par rapport au HTML, notamment les variables entourées d'une double accolade comme {{heure}}. Il englobe également le CSS.

3) Formulaire avec Python Flask

Dernière partie, qui va finalement nous ramener à notre convertisseur IEEE-754, que nous avons délaissé depuis un moment… Nous allons maintenant nous intéresser à la gestion des formulaires.

a. Méthode POST

❐ — Exercice 7 —

- Dans le corps du fichier **index.html** de l'ex 3, rajoutez :

```
<form action="http://localhost:5000/resultat" method="post">

<label>Nom</label> : <input type="text" name="nom" />
<label>Prénom</label> : <input type="text" name="prenom" />
<input type="submit" value="Envoyer" />

</form>
```

- Créez un nouveau fichier que l'on appellera **résultat.html** et placez-le dans le dossier **templates**.
  Ce fichier contient le code suivant :

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="utf-8">
        <title>Résultat</title>
    </head>
    <body>
        <p>Bonjour {{prenom}} {{nom}}, j'espère que vous allez bien.</p>
    </body>
</html>
```

- Finalement, dans le fichier **views.py**, importez l'instruction `request` de la bibliothèque flask ainsi que la fonction décorée suivante :

```
@app.route('/resultat',methods = ['POST'])
def resultat():
  result = request.form
  n = result['nom']
  p = result['prenom']
  return render_template("resultat.html", nom=n, prenom=p)
```

- Relancez views.py puis tapez localhost:5000 dans la barre d'adresse de votre navigateur.

Explication générale :

(1) Nous effectuons une requête HTTP avec l'URL "/", le serveur HTTP génère une page HTML à partir du fichier "index.html". Cette page, qui donne l'heure, contient aussi un formulaire comme indiqué par la balise form. Cette page est affichée par le navigateur (le client).

(2) La balise form a deux attributs : action="http://localhost:5000/resultat" et method="post". Ces 2 attributs indiquent que le client devra effectuer une requête de type POST (voir le protocole HTTP) dès que l'utilisateur appuiera sur le bouton "Envoyer". Cette requête POST sera envoyée à l'URL "http://localhost:5000/resultat" comme indiquée dans l'attribut **action**. Les données "nom" et "prenom" saisies dans le formulaire seront envoyées au serveur par l'intermédiaire de cette requête.

(3) Le template "resultat.html" utilise ces deux paramètres ("nom" et "prenom"). En réponse à la requête POST, le serveur renvoie une page HTML créée à partir du template "resultat.html" et des paramètres "nom" et "prenom". Si l'utilisateur a saisi "James" et "Bond", le navigateur affichera "Bonjour James Bond, j'espère que vous allez bien."

Quelques points de détails :

- @app.route('/resultat',methods = ['POST']) précise la méthode à utiliser pour cette requête HTTP alors que @app.route('/') ne précise rien. En l'absence de précision, c'est la méthode GET, qui laisse les paramètres visibles, qui sera utilisée.
- Dans views.py, "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" et "prenom") et comme valeurs ce qui a été saisi par l'utilisateur.
  Si l'utilisateur saisit "Martin" et "Sophie", le dictionnaire "request.form" sera :{'nom':'Martin', 'prenom':'Sophie'} . Nous aborderons cette notion de dictionnaire très bientôt.

b. Méthode GET

Pour gérer un formulaire, il est aussi possible d'utiliser la méthode GET à la place de la méthode POST. Voyons quelles différences cela engendre.

❐ — Exercice **8** —

- Dupliquez l'exercice 4 et nommez la copie Exo5. Dans index.html, modifiez la méthode POST et méthode GET.
- Le fichier resultat.html reste inchangé.
- Modifiez le fichier views.py afin d'appeler la méthode GET :

```
@app.route('/resultat',methods = ['GET'])
def resultat():
  result=request.args
  n = result['nom']
  p = result['prenom']
  return render_template("resultat.html", nom=n, prenom=p)
```

- Résumez les points que nous avons du modifier, ainsi que le nom des fichiers que nous avons modifié :

_____

_____

_____

- Relancez views.py puis tapez localhost:5000 dans la barre d'adresse de votre navigateur, entrez des paramètres et validez. Observez la barre d'adresse de votre navigateur. Que remarquez-vous ?

_____

_____

Conclusion :

- Comme indiqué précédemment, dans le cas de l'utilisation d'une méthode "POST" les données issues d'un formulaire sont envoyées au serveur sans être directement visibles, alors que dans le cas de l'utilisation d'une méthode "GET", les données sont visibles (et accessibles dans la barre d'adresse) puisqu'elles sont envoyées par l'intermédiaire de l'URL.

- D'un point de vue sécurité, c'est même pire que cela : les données envoyées par l'intermédiaire d'une méthode "GET" peuvent être modifiées directement dans l'URL.

❐ — Exercice **9** —

- Ouvrez votre navigateur Web et tapez dans la barre d'adresse "localhost:5000".
- Une fois la page web affichée dans votre navigateur, Saisissez "René" pour le prénom et "LaTaupe" pour le nom puis validez en cliquant sur le bouton "Envoyer".
- Une fois que le message "Bonjour René LaTaupe, j'espère que vous allez bien." apparait, modifier directement l'URL dans la barre d'adresse : passez de "localhost:5000/resultat?nom=LaTaupe&prenom=René" à "localhost:5000/resultat?**nom=Frog&prenom=Froggy**", validez votre modification en appuyant sur la touche "Entrée".
- Que se passe-t-il ? Expliquez pourquoi .

_____

_____

Technique :

serveur/URL?**args :** args représentent les arguments récupérables par une requête GET

Conclusion :

Il faut éviter d'utiliser la méthode "GET" pour transmettre des données issues d'un formulaire vers un serveur, à moins que l'on souhaite pouvoir partager ces informations (comme sur un site marchand). En effet, ces données sont directement lisibles dans la barre d'adresse, ce qui est rédhibitoire en terme de sécurité.

Il est aussi important de bien comprendre que la méthode "POST" n'offre pas non plus une sécurité absolue. A l'aide de Fiddler, on peut en effet lire les données transmises à l'aide de la méthode "POST" en analysant la requête HTTP. **Seule l'utilisation du protocole sécurisé HTTPS** garantit un transfert sécurisé des données entre le client et le serveur grâce au chiffrement des données.