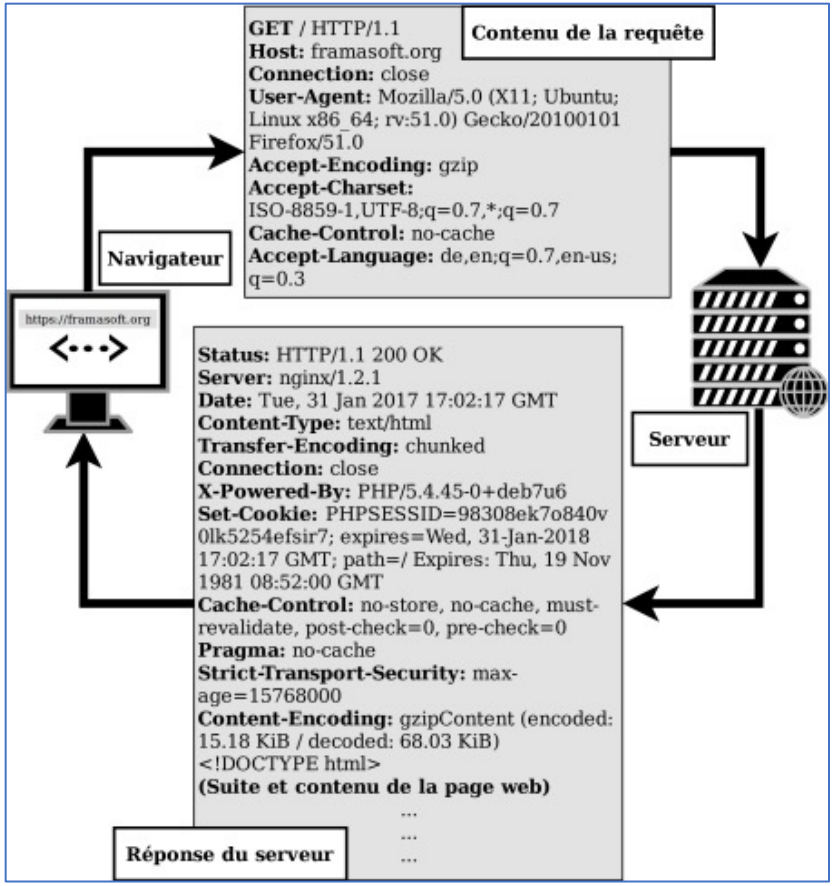
REQUÊTES HTTP

Express JS

Les requêtes HTTP sont la base de la communication sur le Web. Elles permettent aux clients (généralement les navigateurs) de demander des ressources ou d'envoyer des informations à un serveur.



Voici un aperçu des principales méthodes de requête HTTP :

1. GET

- Description : Demande une ressource spécifiée.
- Utilisation : Principalement utilisé pour récupérer des données.
- Exemple : Lorsque vous accédez à une page web, votre navigateur envoie une requête GET pour obtenir le contenu de la page.

2. POST

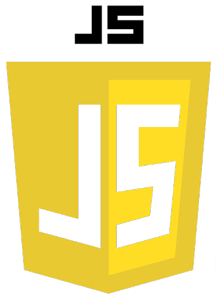
- Description : Soumet des données à une ressource spécifiée pour être traitées.
- Utilisation : Souvent utilisé pour envoyer des données de formulaire à un serveur.
- Exemple : Lorsque vous remplissez un formulaire en ligne et cliquez sur "Envoyer", une requête POST est généralement utilisée pour envoyer les données du formulaire au serveur.

3. PUT

- Description : Met à jour une ressource spécifiée avec les données fournies.
- Utilisation : Utilisé pour mettre à jour une ressource existante ou créer une ressource si elle n'existe pas.
- Exemple : Si vous avez une application de gestion de tâches, vous pourriez utiliser une requête PUT pour mettre à jour une tâche existante.

4. DELETE

- Description : Supprime une ressource spécifiée.
- Utilisation : Utilisé pour supprimer une ressource.
- Exemple : Dans une application de gestion de tâches, vous pourriez utiliser une requête DELETE pour supprimer une tâche.



REQUÊTE HTTP GET

Express JS

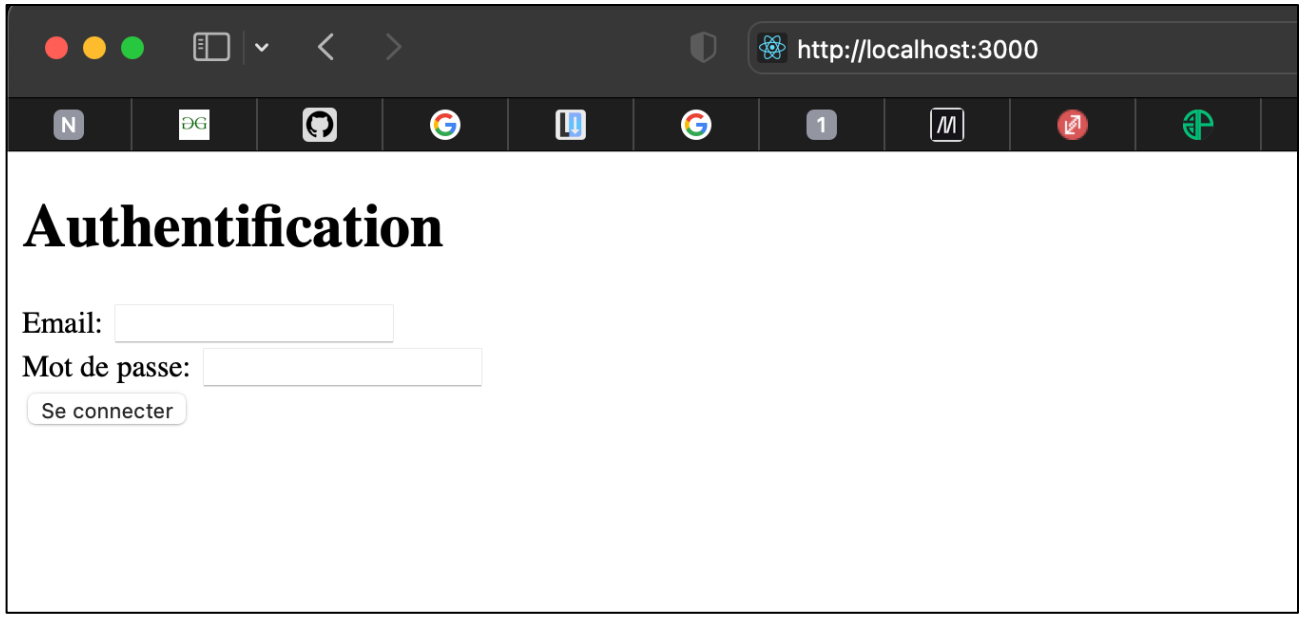
Une requête GET est constituée de trois éléments principaux :

- La ligne de requête (request line) : Cette ligne spécifie la méthode HTTP utilisée pour la requête (dans ce cas, GET), ainsi que l'URL de la ressource demandée.
- Les en-têtes de requête (request headers) : Les en-têtes de requête sont des informations supplémentaires envoyées avec la requête, telles que les informations d'authentification, …Les en-têtes de requête sont facultatifs et peuvent être absents d'une requête.
- La charge utile (payload) : Dans une requête GET, la charge utile est généralement vide car toutes les informations sont transmises dans l'URL de la requête à l'aide des paramètres d'URL.

```
GET / HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0)
Gecko/20100101 Firefox/65.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

COMMENT REALISER UNE REQUÊTE HTTP GET ?

Avec une navigateur web :



Avec curl:

```
curl http://www.example.com
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>Mon site web</title>
  </head>
  <body>
    <h1>Bienvenue sur mon site web !</h1>
    <p>C'est ma première page web créée avec Express.</p>

    <button id="logout-button">Se déconnecter</button>

    <script src="auth.js"></script>
  </body>
</html>
```

```
curl –I http://www.example.com
```

Avec l'option –I, on ne voit que l'entête de la réponse

```
HTTP/1.1 200 OK
X-Powered-By: Express
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Last-Modified: Mon, 27 Feb 2023 14:15:14 GMT
ETag: W/"12b-186933a429f"
Content-Type: text/html; charset=UTF-8
Content-Length: 299
Date: Tue, 28 Feb 2023 16:37:07 GMT
Connection: keep-alive
Keep-Alive: timeout=5
```

REQUÊTE HTTP GET EN JAVASCRIPT

Express JS

```
// URL de l'API ou du service que vous souhaitez interroger
const url = 'https://api.example.com/data';

// Utilisation de la fonction fetch pour envoyer une requête GET
fetch(url)
  .then(response => {
    // Vérification si la requête a réussi
    if (!response.ok) {
      throw new Error('Erreur réseau lors de la récupération des données.');
    }
    // Parse la réponse en JSON
    return response.json();
  })
  .then(data => {
    // Traitement des données reçues
    console.log(data);
  })
  .catch(error => {
    // Gestion des erreurs éventuelles
    console.error('Il y a eu un problème avec l\'opération fetch:', error.message);
  });
```

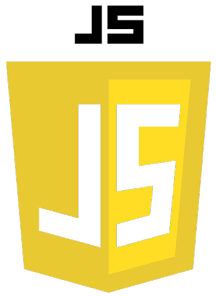
SERVEUR WEB EXPRESS SUR REQUÊTE HTTP GET

Ce code utilise le module express pour créer et définit une route pour la page d'accueil en utilisant la méthode HTTP GET. Lorsque l'application reçoit une requête GET pour la racine (/) de l'URL, elle envoie une réponse avec le texte Bienvenue sur la page d'accueil !.
Le serveur est démarré en écoutant les connexions sur le port 3000 en utilisant la méthode listen.

```
const express = require('express');

const app = express();

// Route pour la page d'accueil
app.get('/', (req, res) => {
  res.send('Bienvenue sur la page d\'accueil !');
});

// Démarrage du serveur
app.listen(3000, () => {
  console.log('Le serveur est démarré sur le port 3000.');
});
```

App est l'instance principale de l'application express.

get() est une méthode associée à cette instance.



SERVEUR WEB EXPRESS


```
app.get('/', (req, res) => {
  res.send('Bienvenue sur la page d\'accueil !');
});
```

La méthode app.get() prend deux arguments : le premier est la route, qui est une chaîne de caractères représentant l'URL de la route, et le second est la fonction de rappel qui est exécutée lorsque la route est accédée. Cette fonction prend deux arguments : l'objet requête (req) et l'objet réponse (res) qui représentent respectivement la requête entrante et la réponse sortante.

```
app.listen(3000, () => { console.log('Le serveur est démarré sur le port 3000.'); });
```

La méthode listen() met en place un serveur HTTP qui écoute les connexions sur le port spécifié.

Lorsque app.listen() est appelé, le serveur est démarré et commence à écouter les connexions sur le port 3000. La fonction de rappel est exécutée une fois que le serveur est en cours d'exécution, et dans cet exemple, elle affiche un message dans la console pour indiquer que le serveur est prêt à accepter les connexions.

REQUÊTE HTTP AVEC PARAMÉTRES D'URL

Express JS

Dans une requête GET, les données peuvent être transmises à l'aide des paramètres d'URL. Les paramètres d'URL sont des chaînes de caractères qui sont ajoutées à l'URL de la requête et qui permettent de transmettre des informations supplémentaires.
Les paramètres d'URL sont ajoutés à l'URL en utilisant le caractère ? suivi de paires clé-valeur séparées par le caractère &. Par exemple, pour transmettre les valeurs nom et prenom dans une requête GET, nous pouvons ajouter les paramètres d'URL suivants à l'URL de la requête :

```
http://localhost:3000/traitement-formulaire?nom=Doe&prenom=John
```

Dans cet exemple, les paramètres nom et prenom sont transmis à la route /traitement-formulaire en tant que paires clé-valeur séparées par le caractère &.

Avec curl :

```
curl -G --data-urlencode "nom=Dupont" --data-urlencode "prenom=Jean"
http://localhost:3000/traitement-formulaire
```

RÉCUPÉRATION AVEC PARAMÉTRES D'URL

Les paramètres d'URL peuvent être récupérés dans Express en utilisant l'objet de requête (req.query), comme ceci :

```
const express = require("express");
const app = express();

// Route pour traiter les données de formulaire
app.get("/traitement-formulaire", (req, res) => {
  const nom = req.query.nom;
  const prenom = req.query.prenom;
  res.send(`Bonjour ${prenom} ${nom} !`);
});

// Démarrage du serveur
app.listen(3000, () => {
  console.log("Le serveur est démarré sur le port
3000.");
});
```

Dans cet exemple, nous avons défini une route pour traiter les données de formulaire (/traitement-formulaire) en utilisant la méthode app.get(). Nous avons récupéré les paramètres d'URL nom et prenom à l'aide de req.query, puis envoyé un message de bienvenue personnalisé en utilisant ces paramètres.

COMMENT REALISER UNE REQUÊTE HTTP POST ?

- Depuis le formulaire d'une page web
- En utilisant curl


```
curl –X POST –d 'nom=Dupont&prenom=Jean' http://localhost:3000
```

Dans cet exemple, la commande curl envoie une requête HTTP POST à l'URL http://localhost:3000 avec des données encodées en URL. Les options utilisées sont les suivantes :

- X POST : Cette option spécifie la méthode HTTP à utiliser (dans ce cas, POST).
- d 'nom=Dupont&prenom=Jean' : Cette option spécifie les données à envoyer avec la requête. Les données sont encodées en URL et comprennent deux champs : nom avec la valeur Dupont et prenom avec la valeur Jean.

Vous pouvez également spécifier l'en-tête de requête Content-Type pour indiquer que les données sont encodées en URL, comme ceci :

```
curl –X POST –H "Content–Type: application/x–www–form–urlencoded"
–d 'nom=Dupont&prenom=Jean' http://localhost:3000/api/data
```

REQUÊTE POST AVEC CURL

Express JS

Pour réaliser une requête HTTP POST avec des données JSON en utilisant curl, vous pouvez utiliser l'option -d et spécifier les données JSON à envoyer avec la requête. Voici un exemple de commande curl pour envoyer une requête HTTP POST avec des données JSON :

```
curl -X POST -H "Content-Type: application/json"
-d '{"email":"user@example.com","password":"password"}'
http://localhost:8080/login
```

```
> POST /login HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.71.1
> Accept: */*
> Content-Type:application/json
> Content-Length: 41
>
```

```
< HTTP/1.1 200 OK
< X-Powered-By: Express
< Content-Type: application/json; charset=utf-8
< Content-Length: 164
< ETag: W/"a4-qlTTRQT61dLsRRbYqENESNjSumo"
< Date: Wed, 01 Mar 2023 14:19:03 GMT
< Connection: keep-alive
< Keep-Alive: timeout=5
```

TRAITEMENT DES REQUÊTES POST AVEC EXPRESS

```
const express = require("express");
const app = express();
const bodyParser = require("body-parser");

// Configuration de body-parser pour traiter les requêtes POST
app.use(bodyParser.urlencoded({ extended: true }));

// Définition de la route pour la page d'accueil (GET)
app.get("/", (req, res) => {
  res.send("Bienvenue sur la page d'accueil !");
});

// Définition de la route pour le traitement du formulaire (POST)
app.post("/traitement-formulaire", (req, res) => {
  const nom = req.body.nom;
  const prenom = req.body.prenom;
  res.send(`Bonjour ${prenom} ${nom} !`);
});

// Démarrage du serveur
app.listen(3000, () => {
  console.log("Le serveur est démarré sur le port 3000.");
});
```

Ceci est un exemple avec une requête post.
Les données sont transmises dans le body