Le dév web en D

Vla comment c'est simple

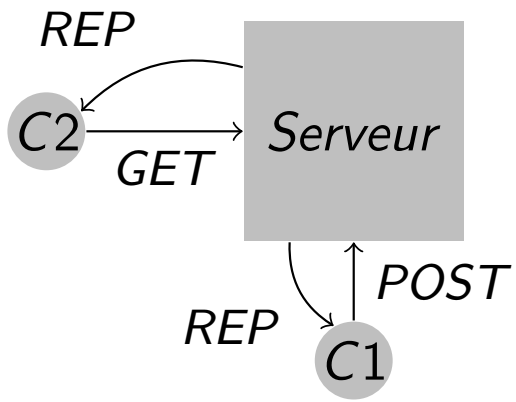
Emile Cadorel

7 mai 2018

Sommaire

Le web - principes fondamentaux

- Des client
- Un serveur
- Des requêtes Http
- Des réponses



Le web - les requêtes - POST vs GET

GET et *POST* sont des requêtes HTTP, envoyées au serveur, qui attendent une réponse. Elle sont envoyé lors d'un événement simple - validation d'un formulaire, chargement d'une nouvelle page ...

GET

- Requête visible dans l'url

  `(/myrequest?myvar=12&myvar2=salut)`

- Modifiable par l'utilisateur

POST

- Requête non visible
- Modifiable par l'utilisateur mais pas monsieur tout le monde

Le web - requêtes asynchrones

Limitations

On a vu des requêtes synchrones, c'est à dire que chaque réponse ouvre une nouvelle page.

AJAX

Asynchronus **J**avascript **A**nd **X**ml permet de faire des requêtes en arrière plan.

Principe de fonctionnement :

- Un évenement a lieu - bouton, chargement d'image ...
- Un objet XMLHttpRequest crée et envoyé par le javascript au serveur
- Le serveur fait comme d'habitude
- Le javascript lis la réponse et modifie la page

Sommaire

Le web en D - vibe.d

Vibe.d est une bibliothèque de programmation réseau. Une partie de vibe.d est utile pour gérer les requêtes HTTP.
La programmation d'un serveur va se faire en 3 parties - **MVC** :

- Modèle - accès à la base, ajout, modification, suppréssion
- Vue - affichage des données, - page web envoyé côté client
- Controleur - récupération des requêtes et génération des réponses.

Attention

Les parties doivent communiquer mais pas interférer, en aucun cas, le controleur peut accéder à la base de données sans passer par le modèle.

Le web en D - création d'un contrôleur

```d
auto router = new URLRouter();
router.registerWebInterface(new MyWebInterface());

auto settings = new HTTPServerSettings();
settings.port = cast(short) 8080;
settings.bindAddresses = ["::", "0.0.0.0"]; // Addr any

listenHTTP(settings, router);
```

Le web en D - création d'un contrôleur

```
class MyWebInterface {


    /**
     * Requete GET sans nom, et sans parametres.
     * Appele par defaut lors du listenHTTP
     */
    void get () {
        // Envoi la page index.dt
        render!"index.dt";
    }
```

Le web en D - Création d'un formulaire

```
html
   head
   body
      form(action="/login", method="POST")
         input(type="text", name="nom")
         input(type="text", name="prenom")
         input(type="submit", value="Envoyer")
```

Va être prise en charge par la fonction suivante :

```
void postLogin (string nom, string prenom) {
    // On affiche la page bonjour, en lui passant des
        donnees
    render!("bonjour.dt", nom, prenom);
}
```

Le web en D - Utilisation des données dans les Vues

```
html
  head
  body
    h1 Bienvenue #{prenom} #{nom}
```

Le web en D - les sessions

Il est parfois utile de savoir à quel client se rattache une requête. Pour ça on utilise des variables de session.
On influe sur la session dans la réponse, et on récupère la session courante dans la requête.
Dans un webservice comme MyWebInterface :

- request - est la requête en cours
- response - est la réponse en cours

Le web en D - les sessions

```
void postLogin(string nom, string prenom) {
    if(request.session) response.terminateSession();
    auto session = response.startSession();
    session.set("user", tuple(nom, prenom));
    render!("bonjour.dt", nom, prenom);
}
```

Le web en D - les sessions

```
void getWho () {
    if (request.session) {
        auto nom = request.session.get ("user") [0];
        auto prenom = request.session.get ("user") [1];
        render!("bonjour.dt", nom, prenom);
    } else render!"index.dt";
}
```

Le web en D - les redirections

Les redirections sont des requêtes envoyé par le serveur au serveur. Elle sont lié à un client, et servent à changer les requêtes dans certains cas (e.g. un client n'est pas connecté, mais veut accéder à une page, on le redirige sur la page d'acceuil).

```
void getData () {
    if (!request.session) {
        redirect ("login", "sponge", "bob"); // fait un
            post(login, nom="sponge", prenom="bob")
    } else {
        // renvoi les donnees
    }
}
```

Sommaire

MongoDB

MongoDB

MongoDB est une base de données addressable en JSON, elle est différente des bases SQL-like dans le sens, où les tables ne sont pas lié, et les données sont des documents, dont les champs ne sont pas figé.

Une base possède :

- Un ensemble de collections
- Un ensemble de documents, dans les collections.

MongoDB

```
[
    {"_id" : 019873897437101, "nom" : "martin", "prenom"
        : "felipe"},
    {"_id" : 098189189738921, "nom", "abidbol", "prenom"
        : "georges", "classe" : "1000%"}
]
```

MongoDB

```
// Trouve tous les utilisateurs
db.users.find({});

// Trouve uniquement ceux dont le nom est abidbol
db.users.find({"name" : "abidbol"});
```

Sommaire

MongoDB en D - connexion à la base

```d
auto client   = connectMongoDB("127.0.0.1");
auto collection = client.getCollection("anniv.users");
```

MongoDB en D - serialisation des donnees

On peut utiliser les structures de D comme des élements utilisable de la base de données.

```
struct Utilisateur {
    @name ("_id") BsonObjectID id;
    string nom;
    string prenom;
}

collection.insert (Utilisateur (
    BsonObjectID.generate (), "abidbol", "georges"
));
```

MongoDB en D - Récupération des données

Là aussi c'est très simple, on peut utiliser les tableau associatif de D, ou une string json.

```
auto user = collection.findOne!Utilisateur(
    ["nom" : "abidbol"]
);

auto user2 = collection.findOne!Utilisateur(
    // q{} veut dire la meme chose que ""
    q{{"user" : "abidbol", "prenom" : "georges"}}
);
```

On appel les éléments utilisé pour trouver un document des sélécteurs, ceux ci peuvent servir à :

- Trouver
- Mettre à jour
- Supprimer

Sommaire

Ajax - envoie de requêtes

```
var xhttp = new XMLHttpRequest();
// (methode, url, synchrone ?)
xhttp.open("GET", "data?name=mydata", true);
xhttp.send();
```

Ajax - récupération des réponses

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
    // 200 est le code de succes
    if (this.readystate == 4 && this.status == 200) {
        document.getElementById("data").innerHTML =
            this.responseText;
    }
}
xhttp.open("GET", "data?name=mydata", true);
xhttp.send();
```

Ajax - Bonne pratique

Le D et le javascript ont un support du format JSON. Donc le mieux c'est de communiquer en JSON.
En JS :

```
var obj = JSON.parse(this.responseText);
document.getElementById("data").innerHTML =
    obj.name;
```

En D :

```
auto obj = parseJSON(q{{"name" : "abidbol"}});
writeln(obj["name"].str);
```

Conclusion

Le web en D

C'est pas compliqué.