B0 - HUB

B-INN-000

Workshop Bot Discord

Créer son premier Bot Discord

Workshop Bot Discord

language: NodeJS
build tool: npm

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

Workshop Bot Discord

Pré-requis

Si vous n'avez pas déjà installé les outils nécessaires à ce workshop suivez les instructions situées ici :

`https://github.com/Fazanwolf/workshop-bot-discord/blob/master/REQUIREMENTS.md`

Configuration Discord

Création d'une Application Discord

Pour pouvoir créer votre bot, vous devrez créer une application discord.
Pour ce faire, créez une nouvelle application sur le Portail Développeur, nommez votre application et acceptez les
clauses.

Portail Développeur

Sur l'onglet `General Information`, vous pouvez customiser votre application (nom, description, icon de profil, etc)

Transformer une Application en Bot

Pour pouvoir invitez votre application sur votre serveur discord de test, vous devez la définir comme bot.
Allez sur l'onglet `Bot` et ajoutez un bot.

Parametrez l'`Authorization Flow` de votre bot comme vous le souhaitez.

Les privilèges

Pour que votre bot puisse utiliser l'API de Discord, il vous faut activer certain privilège.

Qu'est-ce qu'une API ?

Pour ce faire, sur l'onglet Bot, dans `Privileged Gatewy Intent` ajoutez les Gateway que vous souhaitez.

Ne partagez surtout pas votre `BOT_TOKEN`

Ajouter votre bot à un serveur

Maintenant que votre bot est paramètré, il faut l'ajouter à un serveur.
Sur l'onglet `OAuth2`, dans le sous-onglet `URL Generator`, sélectionnez la méthode `In-app Authorization`, cochez `bot`
et ajoutez vos permissions.

Un bon bot est un bot avec des permissions clairement définis.

Copiez le lien généré dans un navigateur et sélectionnez votre serveur.

CONFIGURATION

L'ENVIRONNEMENT

Par sécurité, créez un fichier `.env` à la racine de votre repository.
Dans ce fichier, vous aurez besoin:

- BOT_TOKEN: dans l'onglet `Bot` > `Reset Token`
- CLIENT_ID: l'ID de votre application
- GUILD_ID: Developer Mode > Clique droit sur le serveur où se trouve votre bot > `Copy ID`
- LOG_ID: l'ID d'un channel de votre serveur pour vos logs

```
BOT_TOKEN=VOTRE_BOT_TOKEN
CLIENT_ID=VOTRE_CLIENT_ID
GUILD_ID=VOTRE_GUILD_ID
LOG_ID=VOTRE_CHANNEL_ID
```

Vous ne devez jamais partager votre `BOT_TOKEN`

Pour utiliser les variables d'environnement en Node.js, utilisez `dotenv`

LE PROJET

Le template vous épargne la création d'un projet avec npm

Listes des packages du projet:
- dotenv
- discord.js
- fs
- path

Assurez-vous d'installer les packages nécessaires

Le bot: Moche et pas pratique mais fonctionnel

Afin de voir si votre configuration de bot est fonctionnelle, nous allons utiliser le minimum requis pour faire un bot.

Mise en place du Client Discord

Pour commencer, nous allons créer notre client Discord.
Notre client (`Client`) a besoin de permissions pour fonctionner.
Nous allons donc les définir avec des GatewayIntentBits.

Documentation Discord.js

Ma première commande

Pour faire simple, vous allez créer un tableau de commande.
Une commande est un objet qui stock:
- name: Le nom de ta commande
- description: La description de ta commande

Cherchez comment faire un objet et des arrays par vous-même

Enregistrer les commandes

Pour enregistrer nos commandes, vous aurez besoin de créer une variable `rest`.
`rest` est créer à partir de REST, **version 10** de discord.js.
On utilise ensuite la méthode `put()` avec comme paramètre, la `Routes` pour les commandes d'une application et votre tableau de commandes.

Vous pouvez définir si vous souhaitez enregistrer globalement ou par guild en modifiant la route

Supprimer les commandes

Pour supprimer les commandes qui ne sont plus dans votre code ou à jour.
On utilise la méthode `delete()` avec comme paramètre, la `Routes` pour les commandes d'une application et votre tableau de commandes.

Evènement: “Bot: Je suis prêt!”

Pour notre premier “écouteur” d’évènement et pour pouvoir tester notre bot.
On va avoir besoin d’écouter l’évènement qui permet de savoir si votre bot est prêt à l’usage.
Pour cela, on va créer un listener qui affichera `Bot: Je suis pret!`
Voici un exemple de listener:

```
client.on(Events.TonEvents, async () => {
   // Ton code
});
```

Regardez les évènements `Client`

Attention aux évènements qui doivent être exécuter qu’une seule fois

Tester

Pour tester, vous devrez ajouter `client.login(TON_BOT_TOKEN);` à la toute fin du fichier `bot.js`
La commande a utilisé pour démarer le bot:

```
npm run start
```

Exécuter une commande

Votre commande est enregistré dans la guild mais n’effectue rien pour le moment.
Pour exécuter nos commandes, on doit aussi écouter un évènement.
Une fois le listener, l’évènement et les paramètres mis en place, on vérifie:
- Le type d’interaction corresponds à celui d’une `SlashCommand`, retourner si ce n’est pas le cas.
- Le nom de la commande correspond à celle de notre commande, coder le coeur de notre commande.
Le but de la commande `help` est de répondre à l’utilisateur avec un message d’aide en embed.

Logguer des actions

Pour logguer des actions, on va créer un listener qui écoute un évènement.
Ce listener aura pour évènement la suppression des messages et devra envoyer un message dans un channel choisis.

Félicitation, vous avez réussi à faire un bot trash mais ce n’est que le début du workshop.

Le bot: avec architecture

Création d'un type

On va gérer nos différentes fonctions avec une enumération `InteractionType` qui aura les types d'interaction.
Les types:
- Slash = "slash"
- Button = "button"
- Select = "select"
- Modal = "modal"

Architecture Commandes

Dans le dossier `commands`, créez un fichier `help.slash.js`.
Dans ce fichier, nous allons stocker la commande.
La commande sera sous la forme:

```
module.exports = {
    data: "...",// Le builder type de votre commande
    execute(interaction) {
        // Votre code
    }
};
```

Documentation Discord.js

Mise en place d'un gestionnaire de commandes

Pour gérer nos commandes, nous allons créer un gestionnaire de commandes.
Cela va nous permettre d'automatiser et rendre plus lisible le code.
Pour cela, nous allons créer une Collection `slash` dans client, qui va nous permettre de stocker nos commandes.

Une `Collection` est une `Map` implémenter pour Discord

Vous pouvez ensuite créer une fonction `registerSlashCommand` qui aura:
- En paramètre, le client
- En retour, un tableau de data des commandes enregistrées en JSON

Cette fonction devra stocker les commandes dans un tableau et les enregistrer dans la Collection.
Vous pouvez supprimer le tableau de commande utilisé précédemment et appeler la fonction `registerSlashCommand`
.

Architecture Evènements

Dans le dossier `events`, créez un fichier `ready.slash.js`.
Dans ce fichier, nous allons stocker le listener précédemment créer.
Votre fichier doit exporter un objet avec:
- Le nom de l'évènement
- Un booléen pour savoir si l'évènement
- Une fonction nommée `execute` qui aura les paramètres nécessaires à l'évènement

Copiez tous vos listeners sous cette forme.

Mise en place d'un gestionnaire d'évènements

Vous devez créer une fonction `registerEvents` qui va utiliser les fonctions chargées dans `client.on` et `client.once`.

Si vous avez bien fait votre gestionnaire de commandes, vous pouvez récupérer le corps de votre fonction `registerSlashCommand`

Appelez ensuite la fonction `registerEvents` dans votre fichier `bot.js`.

Créer une interaction: Bouton

Pour créer une interaction de type bouton, vous allez créer un fichier dans `commands` nommé `click-me.slash.js` et `click-me.bouton.js`.
Dans le fichier `click-me.slash.js`, vous allez créer une `data` de type `SlashCommand` et un `execute` qui répondra à l'utilisateur avec un bouton.
Dans le fichier `click-me.bouton.js`, vous allez créer une `data` de type `Button` qui aura les même informations que le boutton de la commande `click-me.slash.js` et un `execute` qui répondra avec une image.

Mise en place d'une gestionnaire d'interaction

Créez une fonction `registerInteraction` qui prendra client, un type.
Cette fonction fonctionnera de la même manière que `registerSlashCommand`, à la différence que vous devrez gérer chaque type.

Attention à la key que vous enregistrez dans la Collection

Appelez ensuite la fonction `registerInteraction` dans votre fichier `bot.js` avant les évènements.

Créer des interactions

Amusez-vous à créer des intéractions de type `Selection` et `Modal`.

Documentation Discord.js

Optimisation

Vous pouvez repasser sur votre code et optimiser le code qui se duplique.

Vous pouvez passer des fonctions dans les paramètres de vos fonctions, faire des boucles, créer des fichiers ou dossiers pour mieux structurer.