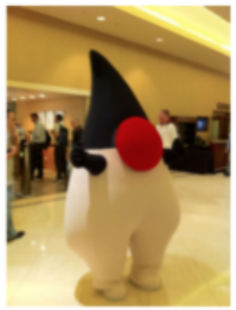
FEIGN + CREST = REST CLI

Me

@binout https://github.com/binout



Équation

FEIGN + CREST = REST CLI

FEIGN
librairie pour faire un client http

CREST
librairie pour faire une Command Line Interface

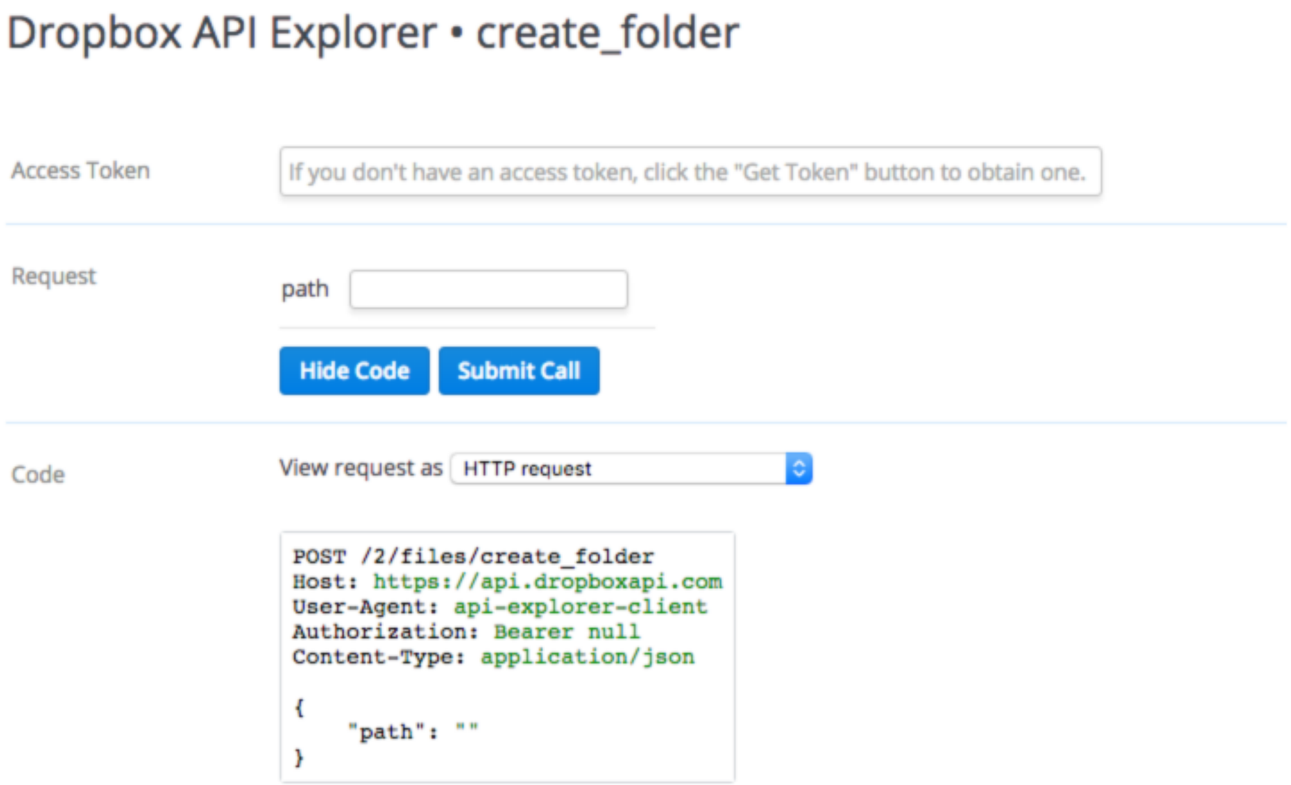
Cas d'utilisation

> *Je veux faire un outil en ligne de commande pour gérer ma Dropbox*

En écrivant le moins de code possible !

API Dropbox

https://dropbox.github.io/dropbox-api-v2-explorer/



Feign

https://github.com/OpenFeign/feign

- client http en java
- pas de dépendances
- des modules d'extension : jackson, gson, jaxb, jaxrs …

Maven

```
<dependency>
    <groupId>com.netflix.feign</groupId>
    <artifactId>feign-core</artifactId>
    <version>${feign.version}</version>
</dependency>
```

Juste une interface

```
public interface Dropbox {

    @RequestLine("POST /files/list_folder")
    @Headers("Content-Type: application/json")
    FolderList listFolder(Path path);

    @RequestLine("POST /files/create_folder")
    @Headers("Content-Type: application/json")
    void createFolder(Path path);

    @RequestLine("POST /users/get_current_account")
    Account currentAccount();
```

Feign Builder

```
static Dropbox api() {
        return Feign.builder()
                .encoder(new GsonEncoder())
                .decoder(new GsonDecoder())
                .requestInterceptor(r -> r.header("Authorization", "Bearer "
+ apiKey()))
                .target(Dropbox.class, "https://api.dropboxapi.com/2");
    }
```

Pas plus compliqué que ça !

```
public static void main(String[] args) {
        Dropbox.api()
                .listFolder(new Path(""))
                .stream()
                .map(Folder::getName)
                .forEach(System.out::println);
    }
```

```
/Photos
/Public
/Documents
```

CREST

https://github.com/tomitribe/crest

- permet de faire rapidement des outils en ligne de commande
- basé sur des annotations (*JAX-RS style*)
- gère la validation des paramètres et la génération d'aide

Maven

```
<dependency>
    <groupId>org.tomitribe</groupId>
    <artifactId>tomitribe-crest-api</artifactId>
    <version>${tomitribe.version}</version>
</dependency>
<dependency>
    <groupId>org.tomitribe</groupId>
    <artifactId>tomitribe-crest</artifactId>
    <version>${tomitribe.version}</version>
    <scope>runtime</scope>
</dependency>
```

Une méthode = une commande

```
@Command
    public String whoami() {
        return API.currentAccount().getName().getDisplayName();
    }
```

- le nom de la méthode est le nom de la commande
- le retour de la méthode est envoyée vers la sortie standard

Une commandes avec des paramètres

```
@Command
    public void mkdir(Path path) {
        API.createFolder(path);
    }
```

- tout paramètre de la méthode est un paramètre **obligatoire** de la commande

Une commande avec des options

```
@Command
    public String ls(@Option("path") @Default("") Path path) {
        return API.listFolder(path).stream()
                .map(Folder::getName)
                .collect(joining(lineSeparator()));
    }
```

- tout paramètre de la méthode annoté avec `@Options` est un paramètre **falculatif** de la commande
- `@Default` permet de donner une valeur par défaut d'une option

Appeler une commande

```
$ java -jar dropbox-cli.jar mkdir
Missing argument: Path...

Usage: mkdir Path
```

Crest fournit une classe Main par défaut : `org.tomitribe.crest.Main`

Intégration avec JLine

- complétion des commandes
- ajoute des commandes basiques : `help`, `exit`, `clear` ...
- gère un historique
- support le pipe |

Maven

```
<dependency>
    <groupId>org.tomitribe</groupId>
    <artifactId>tomitribe-crest-cli</artifactId>
    <version>${tomitribe.version}</version>
</dependency>
```

```
public static void main(String[] args) throws Exception {
        new CrestCli().run(args);
    }
```

Commmand Line Interface

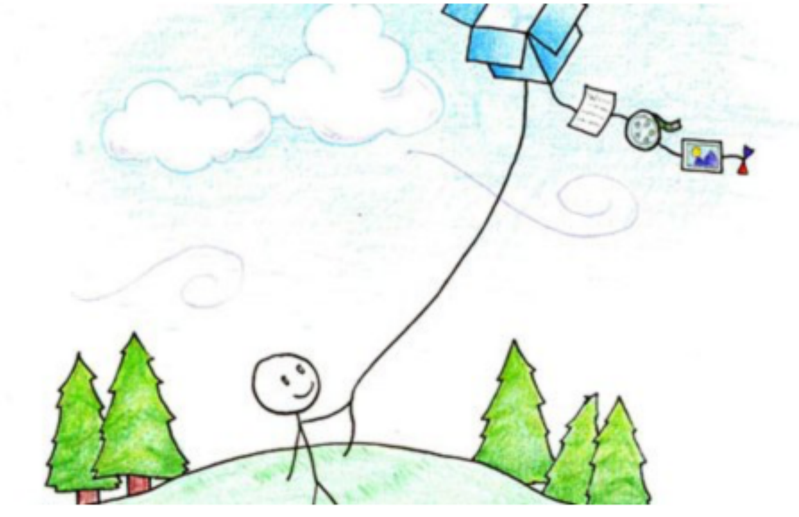
```
$ java -jar dropbox-cli.jar

benoit@1.8.0 $ help
Commands:

    clear
    exit
    help
    ls
    mkdir
    whoami

benoit@1.8.0 $ whoami
Benoit Prioux
```

Démo Dropbox



Pas mal non ?

Très peu de code pour faire un client http en ligne de commande :

- une interface `Feign` (+ les POJOs pour le mapping Json)
- une classe de commandes `CREST`

Ajouter une nouvelle commande :

- ajouter une méthode dans l'interface `Feign`
- ajouter une méthode dans la classe des commandes `Crest`

One more command ?

> *Et si on codait ensemble la génération de lien temporaire ?*

create_shared_link_with_settings

```
POST /sharing/create_shared_link_with_settings
Content-Type: application/json
{
    "path": "...",
    "settings": {}
}
```