Sommaire

Introduction

Afin que votre mod soit détecté par FML, qu'il soit chargé, et de bien gérer les sides (client et serveur), il faut créer une base de votre mod ensuite que des proxy. C'est ce que nous allons faire dans ce tutoriel. Tous les mods ont cette même base, sauf les plugins FML (anciennement connu sous le nom de coremods, que nous verrons plus tard).

Vous avez deux dossier dans le projet Minecraft, un nommé src/main/java dans lequel il faudra placer tous les codes java (fichier /java) et un deuxième nommé src/main/resources dans lequel il faudra placer tous le reste (fichier .info, .png, .lang, .obj, etc ...).

Pré-requis

- Installation de l'espace de travail pour Eclipse

Code

La classe principale :

Commencez par créer deux packages (clic droit sur src/main/java → new → Package).
Si vous avez un nom de domaine, par convention on utilise <extension du domaine>.<nom du domaine> pour le début des packages, donc pour nous, ce sera fr.minecraftforgefrance. Pour minecraftforge.net, net.minecraftforge, etc ... Si vous n'avez pas de nom domaine, vous pouvez utiliser votre email (exemple : com.google.robin4002).
Après le début du nom de package, ajoutez le nom de votre mod, puis common pour le premier package, et client pour le second.
Dans mon cas, je vais nommer ce mod tutoriel (par convention, on ne met pas non plus de majuscule au nom de package), donc mes deux packages sont : fr.minecraftforgefrance.tutoriel.common et fr.minecraftforgefrance.tutoriel.client
Vous l'avez sûrement comprit, le package client va servir pour toutes les classes client, et le package common pour toutes les classes communes au client et au serveur.
Faites un clic droit sur le package common, et créez une nouvelle classe. Cette classe va être votre classe principale, dans mon cas je vais l'appeler ModTutoriel (par convention, les noms de classes commencent toujours par une majuscule).
En dessous de la déclaration de package, ajoutez @Mod. Vous allez avoir une erreur, passez la souris dessus et cliquez sur "import 'Mod' (cpw.mods.fml.common.Mod)". @Mod va se mettre en gris, mais il va toujours avoir une erreur. Il suffit de passer à nouveau la souris dessus, et cliquer sur "Add missing attributes". Un argument modid va s'ajouter, avec comme valeur un String vide. Placez simplement dedans le modid de votre mod, dans mon cas "modtutoriel". Éviter les majuscules dans votre modid ainsi que les espaces.
Dans la foulé, ajoutez les arguments name et version :

```
@Mod(modid = "modtutoriel", name = "Mod Tutoriel", version = "1.0.0")
```

name : le nom du mod, ici vous pouvez mettre des majuscules et des espaces.
version : la version de votre mod.
Votre classe devrait ressembler à ceci :

```
package fr.minecraftforgefrance.tutoriel.common;
import cpw.mods.fml.common.Mod;
```

```
@Mod(modid = "modtutoriel", name = "Mod Tutoriel", version = "1.0.0")
public class ModTutoriel
{
    // La suite va se mettre ici
}
```

Ensuite, nous allons ajoutez l'instance du mod. Dans la zone que j'ai mit en commentaire juste au dessus (// La suite va se mettre ici) ajoutez :

```
    @Instance("modtutoriel")
    public static ModTutoriel instance;
```

C'est l'instance du mod. Pour faire simple, c'est un objet static qui renvoie sur votre classe principale. L'instance va être utilisé principalement dans le réseau (pour les gui, les paquets, et les mobs). Si vous connaissez le java, vous allez dire qu'il y a un problème. En effet, on peut croire que instance est null, puisqu'il n'a pas été initialisé. En fait, c'est l'annotation @Instance qui va initialiser la variable, il est donc important de ne rien mettre entre @Instance("modtutoriel") et public static ModTutoriel instance. Les deux doivent être à la suite.
Pour ceux qui ne l'ont pas devinés, @Instance est suivi du modid. Vous devez donc mettre votre modid.

Maintenant nous allons voir quelques méthodes :

```
    @EventHandler
    public void preInit(FMLPreInitializationEvent event)
    {
    }

    @EventHandler
    public void init(FMLInitializationEvent event)
    {
    }

    @EventHandler
    public void postInit(FMLPostInitializationEvent event)
    {
    }
```

Ajoutez-les après public static ModTutoriel instance;.
@EventHandler est une méthode nécessaire pour dire à FML que la méthode qui suit correspond à une étape du chargement du jeu. C'est aussi ce qui va faire que fml va appeler cette méthode (si vous ne mettez pas @EventHandler, votre méthode ne sera jamais exécuté). Ensuite les 3 méthodes ont pour argument un event qui correspond à une étape de chargement du jeu.
FMLPreInitializationEvent : c'est la première exécuté, elle va servir pour lire le fichier de configuration, enregistrer les blocs, les items, la plupart des choses qui se trouve dans le GameRegistry.
FMLInitializationEvent : toutes les choses secondaires iront ici, enregistrement des events, des rendus, des recettes, etc ...
FMLPostInitializationEvent : la dernière chose faite avant le lancement du jeu, permet de principalement d'interagir avec les autres mods.
Il existe d'autres event moins utilisés comme FMLServerStartingEvent qui sert à enregistrer les commandes. Les autres ne m'ont jamais servit. Vous pouvez voir la liste complète dans la javadoc de @EventHandler (il vous suffit de passer la souris dessus).

Le mcmod.info :

Nous allons en profiter qu'un exemple tout prêt soit disponible depuis la 1.7 pour le compléter. Ouvrez le dossier src/main/resources puis ouvrez le fichier mcmod.info (soit avec l'éditeur d'eclipse, soit avec un éditeur externe à eclipse, peu importe).
Dans modid, mettez votre modid, et dans name le nom de votre mod. Dans description, faites une courte description de votre mod, dans les cases version et mcversion laissez les valeurs par défaut, elles seront automatiquement remplacez lors du build du mod. Dans

url, mettez le lien de votre mod, vous pouvez laisser vide updateUrl, Concernant authors son nom a changé, remplacez-le par authorList. Il s'agit d'une liste, si vous êtes plusieurs développeurs :

```
"authorList": ["auteur1", "auteur2", "auteur3", "auteur4", "etc .."],
```

Sinon :

```
"authorList": ["auteur"],
```

Dans credits, vous pouvez citer les contributeurs, ou la personne qui vous à donné l'idée de faire ce mod, ou encore citer minecraftforgefrance car sans nous votre mod n'existerai pas :troll:
Dans logoFile, donnez le chemin complet vers le fichier png de votre logo (avec /assets/modid/logo.png, le fichier logo.png sera dans src/main/resources/assets/modid/.
screenshots est une liste comme authors, je ne l'ai jamais utilisé mais il me semble que comme pour le logoFile, il faut mettre les chemins vers les screenshots.
Laissez dependencies vide, il ne fonctionne pas, si votre mod est dépendant d'un autre, utilisez dependencies dans l'annotation @Mod.

Les proxy :

L'un des plus gros avantage de forge, c'est qu'il permet de faire des mods universels, il n'y a donc qu'une seule version de votre mod pour le client et le serveur. Les proxy vont être très utilisé pour ça, car en fonction du side (client ou serveur) les fonctions seront soit appelé dans le client proxy, soit dans le common proxy.
Créez deux nouvelles classes, ClientProxy et CommonProxy. Mettez ces deux classes soit dans un nouveau package dédié aux proxy (fr.minecraftforgefrance.tutoriel.proxy dans mon cas), soit CommonProxy dans le package common et ClientProxy dans le package client. Le client proxy doit hériter du CommonProxy (public class ClientProxy extends CommonProxy).
Ajoutez une méthode nommé registerRender du type void dans les deux proxy, avec l'annotation @Override dans le ClientProxy.
Dans cette méthode ajoutez

```
System.out.printnl("méthode côté serveur/client");
```

Vous devrez maintenant avoir :

```
package fr.minecraftforgefrance.tutoriel.proxy;
public class CommonProxy
{
        public void registerRender()
        {
                System.out.println("méthode côté serveur");
        }
}
```

Et pour le client proxy :

```
package fr.minecraftforgefrance.tutoriel.proxy;
public class ClientProxy extends CommonProxy
{
        @Override
        public void registerRender()
        {
                System.out.println("méthode côté client");
        }
}
```

Bien sûr, afficher ceci n'a aucun intérêt dans un mod, c'est purement démonstratif. Actuellement la méthode registerRender n'est pas appelé dans la classe principale, donc même si vous lancer le jeu, il n'y aura rien dans la console. Nous allons donc déclarer les proxy et appeler cette méthode dans la méthode init.
Retournez dans la classe principale et ajoutez en dessus de l'instance et au dessus de la méthode preInit :

```
    @SidedProxy(clientSide = "le.package.de.votre.client.proxy.ClientProxy", serverSide
= "le.package.de.votre.common.proxy.CommonProxy")
    public static CommonProxy proxy;
```

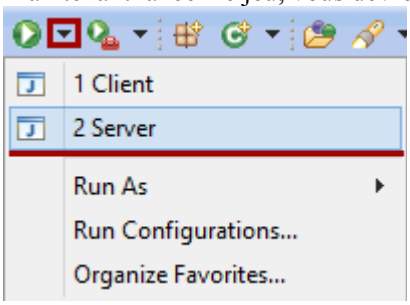
Replacer par le chemin par ce qui vous concerne, dans mon cas :

```
    @SidedProxy(clientSide = "fr.minecraftforgefrance.tutoriel.proxy.ClientProxy",
serverSide = "fr.minecraftforgefrance.tutoriel.proxy.CommonProxy")
```

Pour finir, nous allons appeler cette dans la fonction init :

```
    @EventHandler
    public void init(FMLInitializationEvent event)
    {
        proxy.registerRender();
    }
```

Maintenant lancez le jeu, vous devrez voir méthode côté client. Lancer le serveur :



Vous devrez voir méthode côté serveur dans les logs.
Voila donc pour le principe des proxy, on peut passer à la suite.

Les fichiers de lang :

Pour finir, nous allons nous occuper des fichiers .lang et des autres ressources du mod. Ouvrez votre dossier forge/src/main/resources. Normalement il y a déjà le fichier mcmod.info. Créez un nouveau dossier nommé "assets", et dans ce dossier créez un dossier portant votre modid en minuscule. Dans ce dossier, créez un dossier "textures" et un dossier "lang".
Le dossier textures servira plus tard pour les textures. Allez dans le dossier lang, et créez de fichier : en_US.lang et fr_FR.lang. Vous pouvez faire d'autres fichiers de lang pour les autres langages, ils sont détectés automatiquement. Nous les compléterons plus tard.

Résultat

Voir le commit sur github
Le commit sur github montre clairement où ont été placés les fichiers, ainsi que ce qui a été ajouté et retiré dans le fichier.

Crédits

Rédaction :

- **robin4002**

Correction :

- **Superloup10**

Retour sur le site