02 Vytvoření projektu

V této lekci si ukážeme, jak vytvořit nový projekt s módem. Jako první mód si ukážeme, jak do hry přidat vlastní item, takže podle toho bychom měli níže volit název módu.

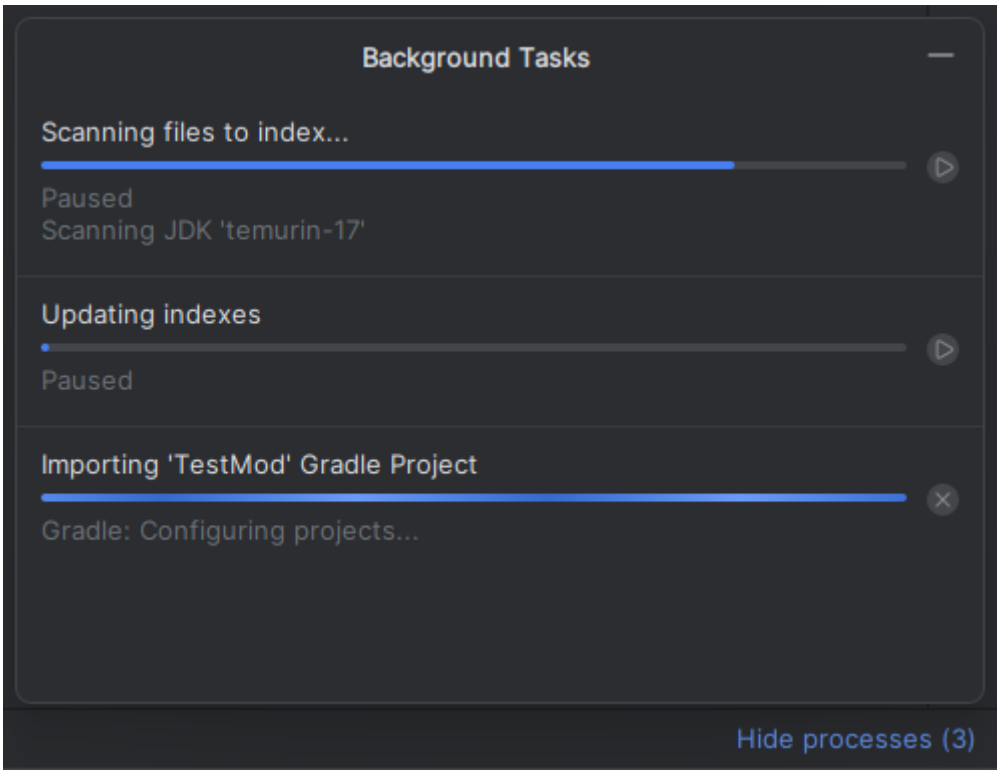
Pro vytvoření projektu použijme stránky fabricmc.net, kde kromě užitečných informací najdeme také generátor projektu. Projekt bychom si sice mohli vytvořit sami, ale bylo by to zbytečně zdlouhavé. Tento generátor nám vytvoří základní strukturu projektu a nastaví vše potřebné. Generátor najdeme na stránce https://fabricmc.net/develop/template/.

Tam zadáme následující informace:

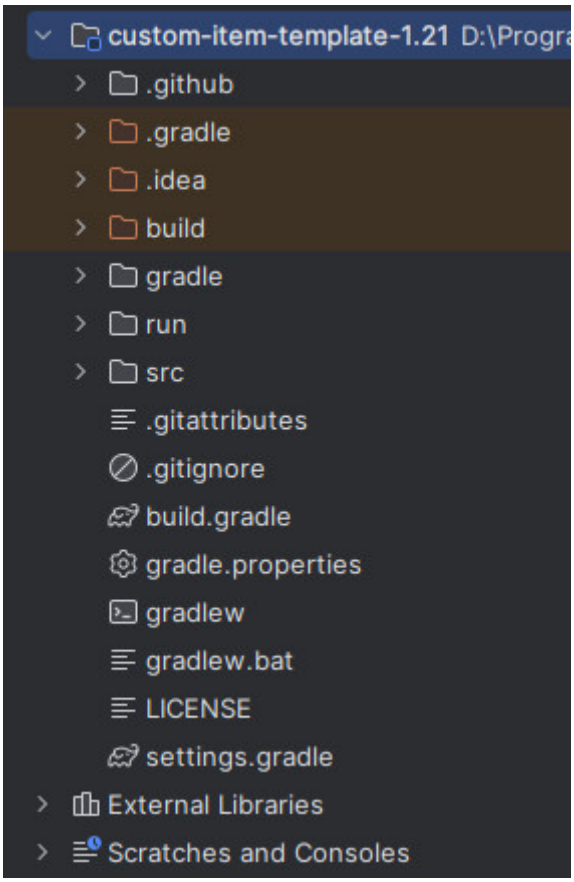
- *Mod Name* - název módu
  - používáme znaky bez diakritiky
  - název by měl odpovídat obsahu módu
  - například `Custom item`
- *Package Name* - název balíčku
  - používá se v programovacím jazyce Java na odlišení jednotlivých balíčků, ze kterých se poté tvoří výsledný program
  - název by měl být unikátní, proto můžeme použít například `jmeno.nazev-modu`
  - například tedy `martin.customitem`
- *Minecraft Version* - verze Minecraftu, pro kterou je mód určený
  - pro naše účely budeme volit vždy `1.21`
    - je to hlavně z důvodu, abychom si pak všechny módy mohli zahrát zároveň a také, abychom nemuseli pořád stahovat nové verze Minecraftu
- *Advanced Options* - pokročilé možnosti
  - z dalších možností necháme zvolené pouze `Split client and common sources`
    - tím se nám kód rozdělí na 2 části - část pro server a část pro hráče

Po stažení je potřeba extrahovat soubory, protože se nám stáhl *zip* soubor. To uděláme tak, že na soubor klikneme pravým tlačítkem myši a zvolíme možnost *Extrahovat vše*. Pak jen vybereme složku, do které chceme projekt s módem umístit. Jakmile máme soubory extrahované, tak projekt otevřeme tak, že soubor *build.gradle* otevřeme pomocí ItelliJ IDEA.

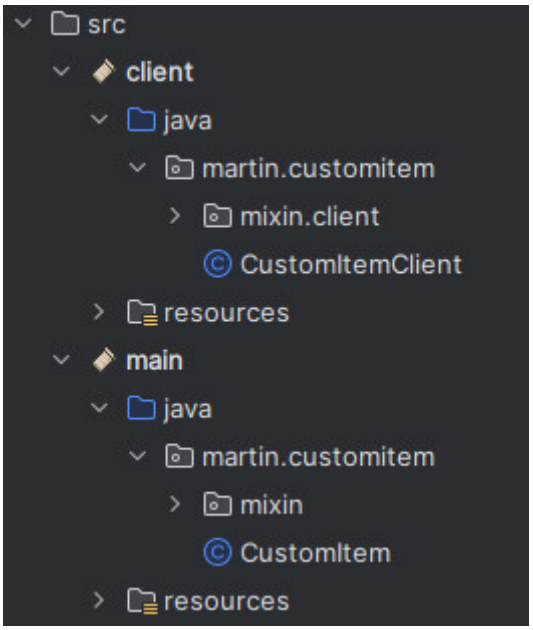
Po otevření nového projektu je důležité nechat doběhnout veškeré procesy (vidíme v pravém dolním rohu). Tyto procesy provádí nastavení projektu, vytváří veškeré potřebné soubory a případně stahují dodatečné soubory potřebné pro tvorbu módu. Například stahují i samostatnou verzi Minecraftu, takže nemusíme nic nastavovat a Minecraft můžeme spustit i s naším módem přímo z editoru.



Po doběhnutí všech procesů bychom měli na levé straně vidět následující strukturu projektu:



Nás bude zajímat hlavně složka `src`, kde najdeme zdrojové soubory módu.



Zde můžeme vidět rozdělení kódu na `client` a `main`. Toto rozdělení je důležité hlavně při použití módu na serveru, kde část kódu ve složce `client` běží u hráče na počítači a ovlivňuje, jak hra vypadá. Část ve složce `main` pak běží na serveru a ovlivňuje, co se ve hře děje.

Základní termíny jazyka Java

Nyní si vysvětlíme základní termíny, které budeme u programování používat. V jazyce Java se program skládá z **tříd** (`class`) a ty obsahují **metody**. Metody pak dále obsahují kód, který se spustí po zavolání dané metody. Metody mohou mít také parametry. Ty se uvádí do závorek. Náš mód nyní obsahuje 2 třídy `TestMod` a `TestModClient`. Tyto třídy dále obsahují v základu metody `onInitialize()` a `onInitializeClient()`. Ty se spouští ihned při načtení módu, tedy při samotném spouštění hry. Do těchto metod budeme dávat kód, kterým provede počáteční nastavení módu. Například přidání nových bloků, itemů nebo příkazů.

Nás bude nyní zajímat hlavně třída `CustomItem` ve části `main`. Ta by nyní měla obsahovat následující kód:

```java
package martin.customitem;

import ...

public class CustomItem implements ModInitializer {  1 usage
    public static final String MOD_ID = "custom-item";  1 usage

    // This logger is used to write text to the console and the log file.
    // It is considered best practice to use your mod id as the logger's name.
    // That way, it's clear which mod wrote info, warnings, and errors.
    public static final Logger LOGGER = LoggerFactory.getLogger(MOD_ID);  1 usage

    @Override
    public void onInitialize() {
        // This code runs as soon as Minecraft is in a mod-load-ready state.
        // However, some things (like resources) may still be uninitialized.
        // Proceed with mild caution.

        LOGGER.info("Hello Fabric world!");
    }
}
```

První 2 řádky můžeme prozatím přeskočit. Zajímat nás bude až řádek `public class CustomItem...` - definice třídy s názvem `CustomItem`.

Kód ve třídě píšeme vždy do složených závorek `{}`.

> **Psaní složených závorek**
>
> `{` - Ctrl+Alt+B
> `}` - Ctrl+Alt+N

Proměnné

Proměnné se v programování používají na ukládání dat, se kterými bude program dále pracovat. Každá proměnná má nějaký **typ**. Ten určuje, jaká data proměnná obsahuje. Například typ `String` je na text, `int` je na celá čísla, `boolean` je na pravdivostní hodnoty (pravda, nepravda). Další si ukážeme, až budeme potřebovat.

Proměnné se definují následujícím způsobem a hodnota se přiřazuje pomocí `=`

```
typ nazev = hodnota;
```

> **Info**

Na konci každého řádku musíme psát `;` (středník).

Dále můžeme upravovat přístup k proměnné. Například, abychom k hodnotě mohli přistupovat jen z některých částí kódu nebo abychom nemohli hodnotu proměnné změnit po vytvoření a hodnota zůstala stejná po celou dobu běhu programu. Tyto **modifikátory píšeme před typ proměnné**.

- `public` - proměnná je dostupná ze všech částí kódu
- `private` - proměnná je dostupná jen z třídy, ve které jsme ji definovali
- `static` - k proměnné můžeme přistupovat i bez vytvoření objektu třídy
- `final` - hodnotu proměnné nemůžeme po vytvoření změnit

```
1   public static final String MOD_ID = "custom-item";
```

Tímto například definujeme proměnnou s názvem `MOD_ID`, která může obsahovat pouze text (typ `String`), nemůžeme ji měnit (`static final`), je dostupná ze všech částí našeho módu (`public`) a obsahuje hodnotu `custom-item`, což je název našeho módu.

> ⓘ `String`
>
> Hodnota typu `String` (textový řetězec) se vždy uvádí do uvozovek `" "`

> ⓘ **Konstatní hodnoty**
>
> Pro **konstantní (neměnné) proměnné** (`static final`) se v Javě pro název používají vždy **velká písmena**.

```
1   public static final Logger LOGGER = LoggerFactory.getLogger(MOD_ID);
```

Stejně tak máme vytvořenou další proměnnou s název `LOGGER`, která je typu `Logger` a také je dostupná odkudkoliv a nemůžeme ji měnit.

Samotný objekt třídy `Logger` se vytvoří tak, že zavoláme metodu `getLogger()` ze třídy `LoggerFactory`.

> ⓘ **Volání metody**
>
> Metody voláme pomocí `.` tečky.

Do metody `getLogger()` musíme doplnit ještě argumenty (argumenty se uvádí do závorek) s tím, jak chceme *logger* pojmenovat. Tento název se nám pak bude zobrazovat v konzoli a díky tomu můžeme odlišit, co do konzole vypsat Minecraft a co náš mód. Pro toto můžeme použít název našeho módu, který máme uložený v proměnné `MOD_ID`. Do závorek stačí tedy napsat jen název proměnné.

Logování

Proměnnou typu `Logger` můžeme použít na vypisování informací v určitých částech kódu. Můžeme vypisovat například chyby nebo užitečné informace, abychom věděli, že mód běží správně nebo když dojde k chybě, tak bychom věděli, kde a případně proč k ní došlo.

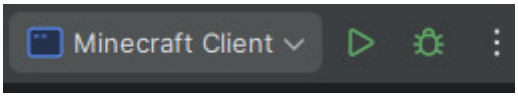
V metodě `onInitialize()` pak máme následující řádek:

```
1   LOGGER.info("Hello Fabric world!");
```

Zavoláním metody `info()` z našeho pojmenovaného `Loggeru` můžeme do konzole hry vypsat nějakou zprávu. Tuto zprávu napíšeme do uvozovek `" "`.

Spuštění hry

Hru spouštíme vpravo nahoře. Měli bychom mít zvolenou možnost `Minecraft Client`. První možností spustíme hru klasicky. Druhou možností pak spustíme ve speciálním režimu, kdy můžeme aplikovat změny bez nutnosti restartovat hru nebo když kód narazí na chybu, tak nám ukáže, kde přesně k chybě došlo. Tento režim se nazývá tzv. *Debug* režim.



Nyní je jedno, jakým způsobem hru spustíme, ale obecně doporučuji při vývoji používat *Debug* režim.

Po spuštění hry se nám ve spodní části obrazovky zobrazí konzole, kde se nám vypisují informace ze hry. Toto je právě *logování*. Samotná hry i módy zde mohou vypisovat informace o jejich běhu.

```
[17:57:23] [Render thread/INFO] (Minecraft) Environment: Environment[sessionHos
[17:57:23] [Render thread/INFO] (Minecraft) Setting user: Player189
[17:57:24] [Render thread/INFO] (custom-item) Hello Fabric world!
[17:57:24] [Render thread/INFO] (Indigo) [Indigo] Registering Indigo renderer!
```

Můžeme zde najít i zprávu z našeho módu:

```
1   [17:57:24] [Render thread/INFO] (custom-item) Hello Fabric world!
```

Vidíme, že se nám zobrazuje název našeho módu `custom-item` a zpráva napsaná v metodě `info()`, Pokud zprávu změníme a hru spustíme znovu, tak se vypíše naše nová zpráva.