07 Item jako jídlo

V této lekci si ukážeme, jak nastavit, aby se náš item dal jíst a při snězení dal hráči nějaký efekt.

Na toto se v módech používá třída `FoodComponent`. Ta obsahuje veškeré nastavení, které se týká jídla.

Ve třídě `ModItems` si přidáme proměnnou typu `FoodComponent`. U té si nastavíme, jaké má vlastnosti a poté tento komponent aplikujeme na náš item.

> ⚠ **Důležité**
>
> Tuto proměnnou musíme v kódu umístit před proměnnou s naším itemem, protože jinak by nám editor napsal chybu, že tento komponent nemůžeme použít.

Proměnnou si pojmenujeme podle toho, co bude dělat. Například pokud bude hráči dávat efekt rychlosti, tak ji můžeme pojmenovat `SPEED_FOOD_COMPONENT`.

Nová komponenta se vytváří pomocí `new FoodComponent.Builder()`. Zatím nám editor bude zobrazovat chybu, ale to je správně, protože komponentu nemáme dopsanou.

```
1  public class ModItems {
2      public static final FoodComponent SPEED_FOOD_COMPONENT = new
   FoodComponent.Builder()
3
4      public static final Item MAGIC_DUST = register(
5              new Item(new
   Item.Settings().maxCount(16).food(SPEED_FOOD_COMPONENT)),
6              "magic_dust"
7      );
```

Jednotlivé vlastnosti nastavujeme pomocí volání metod. Pro přehlednost doporučuji každou metodu psát na samostatný řádek.

- `alwaysEdible`
  - tímto nastavíme, že item můžeme jíst kdykoliv - i když má hráč plný hunger bar

```
1  public static final FoodComponent SPEED_FOOD_COMPONENT = new
   FoodComponent.Builder()
2          .alwaysEdible()
```

- `nutrition`

- určuje, kolik hunger baru se hráči doplní
- parametr je celé číslo

```
1  public static final FoodComponent SPEED_FOOD_COMPONENT = new
   FoodComponent.Builder()
2          .alwaysEdible()
3          .nutrition(1)
```

- `saturationModifier`
  - upravuje, kolik hunger baru se hráči doplní
  - parametr je desetinné číslo (za číslo musíme napsat `f`)

```
1  public static final FoodComponent SPEED_FOOD_COMPONENT = new
   FoodComponent.Builder()
2          .alwaysEdible()
3          .nutrition(1)
4          .saturationModifier(1f)
```

Vzorec na doplnění jídla

Celý vzorec pro výpočet, kolik se hráči doplní jídla je `nutrition * saturationModifier * 2`

Jídlo funguje stejně jako životy, takže `1` je půlka jednoho masa v hunger baru

- `snack`
  - nastavuje dobu jezení na polovinu (`0.8s`). Jinak je tato doba `1.6s`

```
1  public static final FoodComponent SPEED_FOOD_COMPONENT = new
   FoodComponent.Builder()
2          .alwaysEdible()
3          .nutrition(1)
4          .saturationModifier(1f)
5          .snack()
```

- `statusEffect`
  - určuje, jaký efekt hráč po snězení dostane
  - jako parametr uvádíme objekt třídy `StatusEffectInstance`. U efektu můžeme uvést následující základní parametry
    - typ efektu
      - seznam je ve třídě `StatusEffects`
    - délka trvání efektu

        - uvádí se opět v jednotce *tick* - pro zjednodušení můžeme číslo vynásobit `20` a dostaneme délku trvání v sekundách
    - síla efektu
        - číslo od `1` do `255`
- další parametr uvádíme, jaká je šance, že hráč tento efekt dostane
    - hodnota je desetinné číslo od `0` do `1` (`1` je `100%`)

```
1  public static final FoodComponent SPEED_FOOD_COMPONENT = new
   FoodComponent.Builder()
2    .alwaysEdible()
3    .nutrition(1)
4    .saturationModifier(1f)
5    .snack()
6    .statusEffect(new StatusEffectInstance(StatusEffects.SPEED, 5 * 20, 1),
   1f)
```

Tento řádek nastavuje, že hráč dostane effect rychlosti na 5 sekund se sílou efektu 1 a dostane ho vždy (100% šance).

Efektů můžeme přidat i více. Například můžeme přidat, že je 10% šance (průměrně by se tento efekt měl aplikovat 1 z 10 snězení itemu), že na 10 sekund dostane efekt nevolnosti se sílou 1.

```
1  public static final FoodComponent SPEED_FOOD_COMPONENT = new
   FoodComponent.Builder()
2          .alwaysEdible()
3          .nutrition(1)
4          .saturationModifier(1f)
5          .snack()
6          .statusEffect(new StatusEffectInstance(StatusEffects.SPEED, 5 *
   20, 1), 1f)
7          .statusEffect(new StatusEffectInstance(StatusEffects.NAUSEA, 10 *
   20, 1), 0.1f)
```

Na závěr musíme použít metodu `build()` , aby se komponent vytvořil.

```
public static final FoodComponent SPEED_FOOD_COMPONENT = new FoodComponent.Builder()
        .alwaysEdible()
        .nutrition(1)
        .saturationModifier(1f)
        .snack()
        .statusEffect(new StatusEffectInstance(StatusEffects.SPEED, 5 * 20, 1), 1f)
        .statusEffect(new StatusEffectInstance(StatusEffects.NAUSEA, 10 * 20, 1), 0.1f)
        .build();
```

Tento komponent poté můžeme aplikovat na náš item tak, že po vytvoření nastavení použijeme metodu `food` a jako parametr uvedeme náš vytvořený komponent.

```
public static final Item MAGIC_DUST = register(
        new Item(new Item.Settings().maxCount(16).food(SPEED_FOOD_COMPONENT)),
        "magic_dust"
);
```

Celý začátek naší třídy bude vypadat následujícím způsobem. Je zde ukázaný jen začátek souboru a jsou zde zvýrazněné změněné řádky.

```
public class ModItems {
    public static final FoodComponent SPEED_FOOD_COMPONENT = new FoodComponent.Builder()
                        .alwaysEdible()
                        .nutrition(1)
                        .saturationModifier(1f)
                        .snack()
                        .statusEffect(new StatusEffectInstance(StatusEffects.SPEED, 5 * 20, 1), 1f)
                        .statusEffect(new StatusEffectInstance(StatusEffects.NAUSEA, 10 * 20, 1), 0.1f)
                        .build();

    public static final Item MAGIC_DUST = register(
            new Item(new Item.Settings().maxCount(16).food(SPEED_FOOD_COMPONENT)),
            "magic_dust"
    );
```