Java 專案：NBT

TYIC 桃高資訊社

NBT

NBT(named binary tag)
是 Minecraft 中幾乎所有資料的儲存格式
NBT 最常見的形式是以字串呈現的 SNBT(stringified NBT)
兩者可以互相轉換

NBT 共有 13 種資料型別
幾乎可以與 Java 中的資料型別對應
SNBT 的寫法也與 JSON 非常相似

SNBT

NBT 資料型別	Java 資料型別	備註
位元組(byte)	byte	SNBT 中使用 "<number>b" 表示，如 1b
布林(boolean)	boolean	SNBT 中的 false、true 可以分別與位元組 0b、1b 互通
短整數(short)	short	SNBT 中使用 "<number>s" 表示，如 5s
整數(int)	int	如 666
長整數(long)	long	如 -987651
單倍精度浮點數(float)	float	如 2.718f
雙倍精度浮點數(double)	double	如 3.14159265358
字串(string)	java.lang.String	SNBT 中可使用一對單引號('')表示字串 如 'tyic'、"tysh"

SNBT

NBT 資料型別	Java 資料型別	備註
位元組陣列(byte array)	byte[]	SNBT 使用 [B;byte1,byte2,...] 表示 如 [B;-10b,false,true]
整數陣列(int array)	int[]	SNBT 使用 [I;int1,int2,...] 表示 如 [I;1111,10,-5]
長整數陣列(long array)	long[]	SNBT 使用 [L;long1,long2,...] 表示 如 [L;1314,-520,888]
串列(list)	java.util.List<T>	SNBT 使用 [element1,element2,...] 表示 如 [1b,true,false] 串列中所有元素的型別需相同 須注意此與上方的任何陣列不相等
複合資料(compound)	java.util.Map<String,?>	SNBT 使用 {key1:value1,key2:value2,...} 表示 鍵類似於字串，但可不加引號 值可以為任意型別 如 {School:'TYSH',"Since":1941S}

SNBT

SNBT 範例如下：

```snbt
{lit_total_time: 1600s, cooking_time_spent: 159s, x: -1530, y:
79, z: -2005, cooking_total_time: 200s, Items: [{count: 62,
Slot: 0b, id: "minecraft:raw_iron"}, {count: 63, Slot: 1b, id:
"minecraft:coal"}, {count: 2, Slot: 2b, id:
"minecraft:iron_ingot"}], id: "minecraft:furnace",
lit_time_remaining: 1042s, RecipesUsed:
{"minecraft:iron_ingot_from_smelting_raw_iron": 2}}
```

物品堆疊元件

物品堆疊元件(item stack component、data component)
是用於儲存物品堆疊固定(已知)會有的額外資料
如界伏盒內的物品、耐久度、最大耐久度、其餘自訂資料等
物品堆疊元件為鍵值映射
且鍵需要被註冊，值則為符合指定資料型別的任意值
物品堆疊元件的字串型式為 `[key1=value1,key2=value2,...]`
但其最終會被轉換為 NBT 的複合資料
以 SNBT 表示為 `{key1:value1,key2:value2,...}`
物品堆疊元件介面為 `net.minecraft.component.ComponentType<T>`

物品堆疊元件

範例：製作一個物品「TNT 遙控器」
對一個 TNT 方塊右鍵後便會綁定該 TNT
再對空氣或其他非 TNT 方塊右鍵後
便會點燃之前所綁定的 TNT

因 TNT 遙控器需紀錄所綁定的 TNT
故需要使 TNT 遙控器有能記錄 TNT 座標的物品堆疊元件

物品堆疊元件

ComponentType<T> 建造者由呼叫靜態方法 <T>builder() 取得

並且需要設定編解碼器(codec)

以用於資料序列化(serialize)和反序列化(deserialize)

大多數類別中都有靜態欄位 CODEC 可供使用

```java
package org.tyic.tyicmod;

import (...)

public class ModDataComponentTypes {
    public static final ComponentType<BlockPos> BLOCK_POS = register("block_pos", BlockPos.CODEC);

    public static <T> ComponentType<T> register(String id, Codec<T> codec) {
        return Registry.register(Registries.DATA_COMPONENT_TYPE, Identifier.of(TyicMod.MOD_ID, id),
                ComponentType.<T>builder().codec(codec).build());
    }

    public static void init() {
        TyicMod.LOGGER.info("Registering mod data component types.");
    }
}
```

ModDataComponentTypes.java

物品堆疊元件

使用 ItemStack
的動態方法
get、set、remove
控制物品堆疊元件

```java
package org.tyic.tyicmod.item;

import (...)

public class TntRemoteItem extends Item {
    public TntRemoteItem(Settings settings) {
        super(settings);
    }

    @Override
    public ActionResult use(World world, PlayerEntity user, Hand hand) {
        if (world.isClient()) return ActionResult.PASS;
        BlockHitResult blockHitResult = raycast(world, user, RaycastContext.FluidHandling.NONE);
        ItemStack stack = user.getStackInHand(hand);
        if (blockHitResult.getType() == HitResult.Type.BLOCK) {
            BlockPos blockPos = blockHitResult.getBlockPos();
            if (world.getBlockState(blockPos).isOf(Blocks.TNT)) {
                stack.set(ModDataComponentTypes.BLOCK_POS, blockPos);
                user.sendMessage(Text.translatable("tooltip.tyicmod.tnt_remote.binding_tnt",
                        blockPos.getX(), blockPos.getY(), blockPos.getZ()).withColor(Colors.GREEN), true);
                return ActionResult.SUCCESS;
            }
        }
        BlockPos tntBlockPos = stack.get(ModDataComponentTypes.BLOCK_POS);
        if (tntBlockPos == null) return ActionResult.PASS;
        if (!world.getBlockState(tntBlockPos).isOf(Blocks.TNT)) {
            stack.remove(ModDataComponentTypes.BLOCK_POS);
            return ActionResult.PASS;
        }
        user.sendMessage(Text.translatable("tooltip.tyicmod.tnt_remote.priming_tnt")
                .withColor(Colors.RED), true);
        world.removeBlock(tntBlockPos, false);
        TntBlock.primeTnt(world, tntBlockPos);
        stack.remove(ModDataComponentTypes.BLOCK_POS);
        return ActionResult.SUCCESS;
    }
```

Text.translatable
第一個參數為翻譯鍵名
該在地化文字可為格式化字串
後方不定長度引數為格式化引數

向玩家發送訊息
第一個參數為 Text 介面
第二個參數若為 true
則會顯示在快捷欄上方
否則會顯示在訊息欄

在指定座標生成 TNT 實體

TntRemoteItem.java (1/2)

物品堆疊元件

欲增加物品描述(tooltip)
需要覆寫 appendTooltip 方法
並在其中呼叫 tooltip.add(Text text) 方法

客戶端專屬內容是無法直接在通用程式碼使用
如 net.minecraft.client
.gui.screen.Screen 類別
但因 appendTooltip 方法
只會在客戶端被呼叫
因此可以在通用程式碼中
新增公開靜態欄位
並在客戶端初始化時
修改此欄位
這樣就能安全的使用
客戶端專屬內容

```java
package org.tyic.tyicmod;

import (...)

public class TyicModClient implements ClientModInitializer {
    @Override
    public void onInitializeClient() {
        Util.hasShiftDown = Screen::hasShiftDown;
    }
}
```

回傳玩家是否按下 Shift 鍵

TyicModClient.java

```java
package org.tyic.tyicmod;

import (...)

public abstract class Util {
    public static Supplier<Boolean> hasShiftDown = () -> false;
    public static final Text PRESS_SHIFT =
            Text.translatable("tooltip.tyicmod.press_shift").withColor(Colors.CYAN);
}
```

Util.java

```java
    @Override
    public void appendTooltip(ItemStack stack, TooltipContext context, List<Text> tooltip, TooltipType type) {
        BlockPos tntBlockPos = stack.get(ModDataComponentTypes.BLOCK_POS);
        if (tntBlockPos != null)
            if (Util.hasShiftDown.get())
                tooltip.add(Text.translatable("tooltip.tyicmod.tnt_remote.binding_tnt",
                        tntBlockPos.getX(), tntBlockPos.getY(), tntBlockPos.getZ()).withColor(Colors.GREEN));
            else tooltip.add(Util.PRESS_SHIFT);
        super.appendTooltip(stack, context, tooltip, type);
    }
}
```

TntRemoteItem.java (2/2)

物品堆疊元件

註冊物品：

```java
package org.tyic.tyicmod.item;

import (...)

public class ModItems {
    (...)
    public static final Item TNT_REMOTE =
            register("tnt_remote", TntRemoteItem::new, new Item.Settings().maxCount(1));
    (...)
}
```

ModItems.java

在地化：

```json
{
  (...),
  "item.tyicmod.tnt_remote": "TNT Remote",
  "tooltip.tyicmod.press_shift": "Press Shift to display more information",
  "tooltip.tyicmod.tnt_remote.binding_tnt": "Binding TNT: x: %d, y: %d, z: %d",
  "tooltip.tyicmod.tnt_remote.priming_tnt": "!!!Priming TNT!!!",
  (...)
}
```

en_us.json

```json
{
  (...),
  "item.tyicmod.tnt_remote": "TNT 遙控器",
  "tooltip.tyicmod.press_shift": "按 Shift 以顯示更多資訊",
  "tooltip.tyicmod.tnt_remote.binding_tnt": "綁定 TNT： x: %d, y: %d, z: %d",
  "tooltip.tyicmod.tnt_remote.priming_tnt": "！！！點燃 TNT！！！",
  (...)
}
```

zh_tw.json

物品堆疊元件

紋理：assets/tyicmod/textures/item/tnt_remote.png

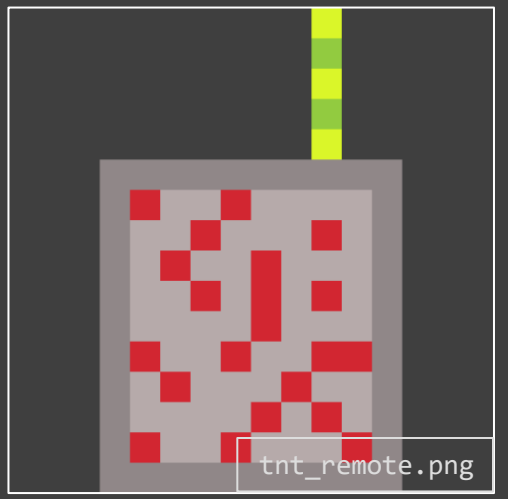
像素：16x16

模型(左下)：

assets/tyicmod/models/item/tnt_remote.json

物品模型映射(右下)：

assets/tyicmod/items/tnt_remote.json




```json
{
  "parent": "minecraft:item/generated",
  "textures": {
    "layer0": "tyicmod:item/tnt_remote"
  }
}
```

tnt_remote.json

```json
{
  "model": {
    "type": "minecraft:model",
    "model": "tyicmod:item/tnt_remote"
  }
}
```

tnt_remote.json

展示影片：https://youtu.be/fsUf3-sfBlg

方塊實體

方塊實體(block entity、tile entity)
是用於儲存方塊狀態之外的任意資料
並且每刻(tick，20 刻 = 1 秒)皆會更新
因此常用於一些能存放東西的方塊，或每刻都要執行功能的方塊
如儲物箱、熔爐、釀造台、日光感應器、海靈核心等
方塊實體類別為 net.minecraft.block.entity.BlockEntity
方塊實體類型類別為
net.minecraft.block.entity.BlockEntityType<T>
我們需要註冊的是方塊實體類型

方塊實體