Databind

Release 0.1.0

Adam Thompson-Sharpe

Aug 03, 2021

CONTENTS

Contents:

CHAPTER

ONE

DATABIND CLI

1.1 What can be transpiled

Either single mcfunction files or entire datapack folders can be transpiled. When transpiling an entire folder, Databind will look for `.databind` files and leave other files alone. Passing databind folder is required for using `:func`.

Note that the namespace inference used for `:func` assumes a typical datapack file structure (`<datapack>/data/<namespace>/functions` for functions), but it **does not check if this is the case.** A `minecraft/tags/functions/` folder may be generated in an unexpected place if an invalid folder is passed.

1.2 Using the CLI

1.2.1 From an installation

To transpile a single file, run `databind file.databind`. A file called *databind-out.mcfunction* will be generated. To transpile a datapack folder, run *databind path/to/datapack*.

1.2.2 With `cargo run`

After building Databind yourself, you can use `cargo run` to run it. Everything works almost the exact same. You just need to add two dashes (`--`) after `run` (eg. `cargo run -- file.databind` or `cargo run -- --help`).

More information is available from the CLI help menu (`databind --help`).

CHAPTER

TWO

SYNTAX

A table of the syntax for different operations.

Syntax	Operation
`:var varName .= <int>`	Define a new variable
`:obj objectiveName <objective>`	Define a new scoreboard objective
`:sobj objectiveName <objective> <target> <assignment operator> <int>`	Set the value of an objective for a given target (eg. `@a` or `PlayerName`)
`:var varName <assignment operator> <int>`	Update the value of an existing variable
`:tvar varName`	Used to test variables in `if` commands (eg. `execute if :tvar varName matches 1`)
`:func name`	Define a function. Generates a new mcfunction file
`:endfunc`	Close a function definition.
`:call <function>`	Call a function. Can infer namespace based on directory (see function calling example)
Assignment Operators	
`+=`	Add to a variable.
`-=`	Subtract from a variable.
`=`	Set the value of a variable.

CHAPTER

THREE

EXAMPLES

Various examples on how to use Databind and its features.

Contents:

3.1 Variable Examples

Examples using variables.

Contents:

3.1.1 Create, Modify & Test

Example

```
# Create a variable called example and set it to 2
:var example .= 2
# Add 1 to example
:var example += 1
# Subtract 2 from example
:var example -= 2
# Set example to 1
:var example = 1
# Say something if example is 1
execute if :tvar example matches 1 run say Variable example is equal to 1!
```

Transpiled

```
scoreboard objectives add example dummy
scoreboard players set --databind example 2
scoreboard players add --databind example 1
scoreboard players remove --databind example 2
scoreboard players set --databind example 1
execute if score --databind example matches 1 run say Variable example is equal to 1!
```

3.2 Objective Examples

Examples using objectives.

Contents:

3.2.1 Create Objective

Create a scoreboard objective.

Example

```
# Create an objective points and set everyone's score to 100
:obj points dummy
:sobj points @a = 100
```

Transpiled

```
scoreboard objectives add points dummy
scoreboard players set @a points 100
```

3.3 Function Examples

Examples using functions.

Contents:

3.3.1 Calling

Different ways to call a function.

`function` command

Built into mcfunctions. Requires a namespace.

`example/data/example/functions/load.databind`

```
:func example_func
say Hello, World!
:endfunc

function example:example_func
```

`:call` (infer namespace)

Add namespaces to functions while transpiling. Allows more freedom with directory names.

`example/data/example/functions/load.databind`

```
:func example_func
say Hello, World!
:endfunc

:call example_func
```

Transpiled, `:call example_func` becomes `function example:example_func`.

`:call` (explicit namespace)

`example/data/example/functions/load.databind`

```
:func example_func
say Hello, World!
:endfunc

:call example:example_func
```

Effectively the same as the `function` command.

3.3.2 Loop

Functions that loop until a counter reaches 0.

Example

`loop_example/data/loop/functions/load.databind`

```
:var counter .= 5

:func loop_main
execute if :tvar counter matches ..0 run tellraw @a "Counter has reached 0"
execute if :tvar counter matches 1.. run :call loop_above_0
:endfunc

:func loop_above_0
tellraw @a "Counter is 1 or higher"
:var counter -= 1
:call loop_main
:endfunc
```

Transpiled

`loop_example.databind/data/loop/functions/load.mcfunction`

```
scoreboard objectives add counter dummy
scoreboard players set --databind counter 5
```

`loop_example.databind/data/loop/functions/main.mcfunction`

```
execute if score --databind counter matches ..0 run tellraw @a "Counter has reached 0"
execute if score --databind counter matches 1.. run function loop:counter_above
```

`loop_example.databind/data/loop/functions/counter_above.mcfunction`

```
tellraw @a "Counter is 1 or higher"
scoreboard players remove --databind counter 1
function loop:main
```

3.3.3 Simple Function

Example

A function that increments a counter and logs when it's run.

```
:var counter .= 0
:func example
tellraw @a "Example_function run"
:var counter += 1
:endfunc
```

Transpiled

`example.databind/data/example/functions/load.mcfunction`

```
scoreboard objectives add counter dummy
scoreboard players set --databind counter 0
```

`example.databind/data/example/functions/example.mcfunction`

```
tellraw @a "Example_function run"
scoreboard players add --databind counter 1
```

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- search