Other languages: 繁體中文

drogon_ctl - Command

After the **Drogon** framework is compiled and installed, it is recommended to create your first project using the command line program `drogon_ctl` which is installed alongside the framework, for convenience there is the shortened command `dg_ctl`. Users can choose according to their preferences.

The main function of the program is to make it easy for users to create various drogon project files. Use the `dg_ctl help` command to see the functions it supports, as follows:

```
$ dg_ctl help
usage: drogon_ctl <command> [<args>]
commands list:
create                  create some source files(Use 'drogon_ctl help
create' for more information)
help                    display this message
version                 display version of this tool
press                   Do stress testing(Use 'drogon_ctl help press' for
more information)
```

Version subcommand

The `version` subcommand is used to print the drogon version currently installed on the system, as follows:

```
$ dg_ctl version
     _
  __| |_ __ ___   __ _  ___  _ __
 / _` | '__/ _ \ / _` |/ _ \| '_ \
| (_| | | | (_) | (_| | (_) | | | |
 \__,_|_|  \___/ \__, |\___/|_| |_|
                 |___/

drogon ctl tools
version:0.9.30.771
git commit:d4710d3da7ca9e73b881cbae3149c3a570da8de4
compile config:-O3 -DNDEBUG -Wall -std=c++17 -I/root/drogon/trantor -
I/root/drogon/lib/inc -I/root/drogon/orm_lib/inc -I/usr/local/include -
I/usr/include/uuid -I/usr/include -I/usr/include/mysql
```

Create sub command

The `create` subcommand is used to create various objects. It is currently the main function of drogon_ctl. Use the `dg_ctl help create` command to print detailed help for this command, as follows:

```
$ dg_ctl help create
Use create command to create some source files of drogon webapp

Usage:drogon_ctl create <view|controller|filter|project|model> [-options]
<object name>

drogon_ctl create view <csp file name> [-o <output path>] [-n <namespace>]|
[--path-to-namespace] //create HttpView source files from csp file

drogon_ctl create controller [-s] <[namespace::]class_name> //create
HttpSimpleController source files

drogon_ctl create controller -h <[namespace::]class_name> //create
HttpController source files

drogon_ctl create controller -w <[namespace::]class_name> //create
WebSocketController source files

drogon_ctl create filter <[namespace::]class_name> //create a filter named
class_name

drogon_ctl create project <project_name> //create a project named
project_name

drogon_ctl create model <model_path> //create model classes in model_path
```

- **View creation**

  The `dg_ctl create view` command is used to generate source files from csp files, see the View section. In general, this command does not need to be used directly. It is better practice to configure the cmake file to executed this command automatically. The command example is as follows, assuming the csp file is `UsersList.csp`.

  ```
  dg_ctl create view UsersList.csp
  ```

- **Controller creation**

  The `dg_ctl create controller` command is used to help the user create the controller's source files. The three controllers currently supported by drogon can be created by this command.

  - The command to create an HttpSimpleController is as follows:

  ```
  dg_ctl create controller SimpleControllerTest
  dg_ctl create controller webapp::v1::SimpleControllerTest
  ```

  The last parameter is the controller's class name, which can be prefixed by a namespace.

  - The command to create an HttpController is as follows:

```
dg_ctl create controller -h ControllerTest
dg_ctl create controller -h api::v1::ControllerTest
```

  - The command to create a WebSocketController is as follows:

```
dg_ctl create controller -w WsControllerTest
dg_ctl create controller -w api::v1::WsControllerTest
```

- **Filter creation**

  The `dg_ctl create filter` command is used to help the user create the source files for filters, see the Middleware and Filter section.

```
dg_ctl create filter LoginFilter
dg_ctl create filter webapp::v1::LoginFilter
```

- **Create project**

  The best way the user creates a new Drogon application project is via the drogon_ctl command, as follows:

```
dg_ctl create project ProjectName
```

  After the command is executed, a complete project directory will be created in the current directory. The directory name is `ProjectName`, and the user can directly compile the project in the build directory (cmake .. && make). Of course, it does not have any business logic.

  The directory structure of the project is as follows:

```
├── build                          Build folder
├── CMakeLists.txt                 Project cmake configuration file
├── cmake_modules                  Cmake scripts for third-party
libraries lookup
│   ├── FindJsoncpp.cmake
│   ├── FindMySQL.cmake
│   ├── FindSQLite3.cmake
│   └── FindUUID.cmake
├── config.json                    The configuration file of the drogon
application, please refer to the introduction section of the
configuration file.
├── controllers                    The directory where the controller
```

```
source files are stored
├── filters                      The directory where the filter files
are stored
├── main.cc                      Main program
├── models                       The directory of the database model
file, model source file creation see 11.2.5
│   └── model.json
├── tests                        The direftory for unit/integration
tests
│   └── test_main.cc             Entry point for tests
└── views                        The directory where view csp files
are stored, the source file does not need to be manually created by
the user, and csp files are automatically preprocessed to obtain view
source files when the project is compiled.
```

- **Create models**

  Use the `dg_ctl create model` command to create database model source files. The last parameter is the directory where models is stored. This directory must contain a model configuration file named `model.json` to tell dg_ctl how to connect to the database and which tables to be mapped.

  For example, if you want to create models in the project directory mentioned above, execute the following command in the project directory:

  ```
  dg_ctl create model models
  ```

  This command will prompt the user that the file will be overwritten directly. After the user enters `y`, it will generate all the model files.

  Other source files need to reference model classes should include model header files, such as:

  ```
  #include "models/User.h"
  ```

  Note that the models directory name is included to distinguish between multiple data sources in the same project. See ORM.

Stress Testing

One can use the `dg_ctl press` command to do a stress testing, there are several options for this command.

- `-n num` Set the number of requests(default : 1)
- `-t num` Set the number of threads(default : 1), Set the number to the number of CPUs to achieve maximum performance
- `-c num` Set the number of concurrent connections(default : 1)
- `-q` No progress indication(default: no)

For example, users can test an HTTP server as follows:

```
dg_ctl press -n1000000 -t4 -c1000 -q http://localhost:8080/
dg_ctl press -n 1000000 -t 4 -c 1000
https://www.domain.com/path/to/be/tested
```

Next: AOP Aspect-Oriented-Programming