

Other languages: [繁體中文](#)

# Quick Start

---

## Static Site

Let's start with a simple example that introduces the usage of drogon. In this example we create a project using the command line tool `drogon_ctl`:

```
drogon_ctl create project your_project_name
```

There are several useful folders in the project directory already:

|                  |  |
|------------------|--|
| — build          | Build folder                           |
| — CMakeLists.txt | Project cmake configuration file       |
| — config.json    | Drogon application configuration file  |
| — controllers    | The folder where the controller source |
| files are stored |  |
| — filters        | The folder where the filter files are  |
| stored           |  |
| — main.cc        | Main program                           |
| — models         | The folder of the database model file  |
| — model.json     |  |
| — views          | The folder where view csp files are    |
| stored           |  |

Users can put various files (such as controllers, filters, views, etc.) into the corresponding folders. For more convenience and less error, we strongly recommend that users create their own web application projects using the `drogon_ctl` command. See [drogon\\_ctl](#) for more details.

Let's look at the main.cc file:

```
#include <drogon/HttpAppFramework.h>
int main() {
    //Set HTTP listener address and port
    drogon::app().addListener("0.0.0.0", 80);
    //Load config file
    //drogon::app().loadConfigFile("../config.json");
    //Run HTTP framework, the method will block in the internal event loop
    drogon::app().run();
    return 0;
}
```

Then build your project as below:

```
cd build
cmake ..
make
```


After the compilation is complete, run the target `./your_project_name`.

Now, we simply add one static file `index.html` to the Http root path:

```
echo '<h1>Hello Drogon!</h1>' >>index.html
```

The default root path is `"/"`, but could also be modified by `config.json`. See [Configuration File](#) for more details. Then you can visit this page by URL `"http://localhost"` or `"http://localhost/index.html"` (or the IP of the server where your wepapp is running).

Hello Drogon!

If the server cannot find the the page you have requested, it returns a 404 page: 404 page

**Note: Make sure your server firewall has allowed the 80 port. Otherwise, you won't see these pages.(Another way is to change your port from 80 to 1024(or above) in case you get the error message below):**

```
FATAL Permission denied (errno=13) , Bind address failed at 0.0.0.0:80 -
Socket.cc:67
```

We could copy the directory and files of a static website to the startup directory of this running webapp, then we can access them from the browser. The file types supported by drogon by default are

- html
- js
- css
- xml
- xsl
- txt
- svg
- ttf
- otf
- woff2
- woff
- eot
- png
- jpg
- jpeg
- gif

- bmp
- ico
- icns

Drogon also provides interfaces to change these file types. For details, please refer to the [HttpAppFramework API](#).

## Dynamic Site

Let's see how to add controllers to this application, and let the controller respond with content.

One can use the `drogon_ctl` command line tool to generate controller source files. Let's run it in the `controllers` directory:

```
drogon_ctl create controller TestCtrl
```

As you can see, there are two new files, `TestCtrl.h` and `TestCtrl.cc` :

`TestCtrl.h` is as follows:

```
#pragma once
#include <drogon/HttpSimpleController.h>
using namespace drogon;
class TestCtrl:public drogon::HttpSimpleController<TestCtrl>
{
public:
    virtual void asyncHandleHttpRequest(const HttpRequestPtr &req,
                                        std::function<void (const
HttpRequestResponsePtr &)> &&callback)override;
    PATH_LIST_BEGIN
    //list path definitions here;
    //PATH_ADD("/path", "filter1", "filter2", HttpMethod1, HttpMethod2...);
    PATH_LIST_END
};
```

`TestCtrl.cc` is as follows:

```
#include "TestCtrl.h"
void TestCtrl::asyncHandleHttpRequest(const HttpRequestPtr &req,
                                        std::function<void (const
HttpRequestResponsePtr &)> &&callback)
{
    //write your application logic here
}
```

Let's edit the two files and let the controller handle the function response to a simple "Hello World!"

TestCtrl.h is as follows:

```
#pragma once
#include <drogon/HttpSimpleController.h>
using namespace drogon;
class TestCtrl:public drogon::HttpSimpleController<TestCtrl>
{
public:
    virtual void asyncHandleHttpRequest(const HttpRequestPtr &req,
                                        std::function<void (const
HttpRequestResponsePtr &)> &&callback)override;
    PATH_LIST_BEGIN
    //list path definitions here

    //example
    //PATH_ADD("/path","filter1","filter2",HttpMethod1,HttpMethod2...);

    PATH_ADD("/",Get,Post);
    PATH_ADD("/test",Get);
    PATH_LIST_END
};
```

Use PATH\_ADD to map the two paths '/' and '/test' to handler functions and add optional path constraints (here, the allowed HTTP methods).

TestCtrl.cc is as follows:

```
#include "TestCtrl.h"
void TestCtrl::asyncHandleHttpRequest(const HttpRequestPtr &req,
                                        std::function<void (const
HttpRequestResponsePtr &)> &&callback)
{
    //write your application logic here
    auto resp=HttpResponse::newHttpResponse();
    //NOTE: The enum constant below is named "k2000K" (as in 200 OK), not
    "k2000K".
    resp->setStatusCode(k2000K);
    resp->setContentTypeCode(CT_TEXT_HTML);
    resp->setBody("Hello World!");
    callback(resp);
}
```

Recompile this project with CMake, then run the target `./your_project_name`:

```
cd ../build
cmake ..
make
./your_project_name
```

Type "`http://localhost/`" or "`http://localhost/test`" in the browser address bar, and you will see "Hello World!" in the browser.

**Note:** If your server has both static and dynamic resources, Drogon uses dynamic resources first. In this example, the response to `GET http://localhost/` is `Hello World!` (from the `TestCtrl` controller file) instead of `Hello Drogon!` (from the static file `index.html`).

We see that adding a controller to an application is very simple. You only need to add the corresponding source file. Even the main file does not need to be modified. This loosely coupled design is very effective for web application development.

**Note:** Drogon has no restrictions on the location of the controller source files. You could also save them in `./` (the project root directory), or you could even define a new directory in `CMakeLists.txt`. It is recommended to use the controllers directory for the convenience of management.

## Next: [Controller - Introduction](#)

---