Other languages: 繁體中文

Controller - HttpSimpleController

You could use the `drogon_ctl` command line tool to quickly generate custom controller class source files based on `HttpSimpleController`. The command format is as bellow:

```
drogon_ctl create controller <[namespace::]class_name>
```

We create one controller class named `TestCtrl`:

```
drogon_ctl create controller TestCtrl
```

As you can see, there are two new files, TestCtrl.h and TestCtrl.cc. Now, let's have a look at them:

TestCtrl.h：

```
#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
HttpResponsePtr &)> &&callback)override;
    PATH_LIST_BEGIN
    //list path definitions here;
    //PATH_ADD("/path","filter1","filter2",HttpMethod1,HttpMethod2...);
    PATH_LIST_END
};
```

TestCtrl.cc:

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

Each HttpSimpleController class can only define one Http request handler, and it is defined by a virtual function override.

The route (or called mapping) from the URL path to the handler is done by a macro. You could add multipath mappings with the `PATH_ADD` macro. All `PATH_ADD` statements should be set between the `PATH_LIST_BEGIN` and `PATH_LIST_END` macro statements.

The first parameter is the path to be mapped, and parameters beyond the path are constraints on this path. Currently, two types of constraints are supported. One is the `HttpMethod` enum Type, which means the Http method allowed. The other type is the name of the `HttpFilter` class. One can configure any number of these two types of constraints, and there are no order requirements for them. For Filter, please refer to Middleware and Filter.

Users can register the same Simple Controller to multiple paths, or register multiple Simple Controllers on the same path (using different HTTP methods).

You could define an HttpResponse class variable, and then use the callback() to return it:

```
    //write your application logic here
    auto resp=HttpResponse::newHttpResponse();
    resp->setStatusCode(k200OK);
    resp->setContentTypeCode(CT_TEXT_HTML);
    resp->setBody("Your Page Contents");
    callback(resp);
```

> **The mapping from the above path to the handler is done at compile time. In fact, the drogon framework also provides an interface for runtime completion mapping. The runtime mapping allows the user to map or modify the mapping through configuration files or other user interfaces without recompiling this program (For performance reasons, it is forbidden to add any controller mapping after running the app().run() method).**

Next: HttpController