Other languages: 繁體中文

Session

Session is an important concept of the web application. It is used to save the state of the client on the server. Generally, it cooperates with the browser's cookie, and drogon provides support for the session. Drogon **close** the session selection by default, you can also close or open it through the following interface:

```
        void disableSession();
        void enableSession(const size_t timeout=0, Cookie::SameSite
sameSite=Cooie::SameSite::kNull);
```

The above methods are all called through the HttpAppFramework singleton. The timeout parameter represents the time when the session is invalid. The unit is second. The default value is 1200. That is, if the user does not access the web application for more than 20 minutes, the corresponding session will be invalid. Setting timeout to 0 means that drogon will retain the user's session for the entire lifetime; The sameSite parameter changes the SameSite attribute of the Set-Cookie HTTP response header.

Make sure your client supports cookies before opening the session feature. Otherwise, drogon will create a new session for each request without SessionID cookie, which will waste memory and computing resources.

Session object

The session object type of drogon is drogon::Session, which is very similar to HttpViewData. It can access any type of object through keywords; support concurrent reading and writing; please refer to the description of Session class for specific usage;

The drogon framework will pass the session object to the HttpRequest object and pass it to the user. The user can get the Session object through the following interface of the HttpRequest class.

```
SessionPtr session() const;
```

The interface returns a smart pointer of the Session object, through which various objects can be accessed;

Examples of sessions

We add a feature that requires session support. For example, we want to limit the user's access frequency. After a visit, if it is accessed again within 10 seconds, it will return an error, otherwise it will return ok. We need to record the last access time in the session, and then compare it with the time of this visit, you can achieve this function.

We create a Filter to implement this function, assuming the class name is TimeFilter, the implementation is as follows:

```
#include "TimeFilter.h"
#include <trantor/utils/Date.h>
#include <trantor/utils/Logger.h>
#define VDate "visitDate"
void TimeFilter::doFilter(const HttpRequestPtr &req,
                          FilterCallback &&cb,
                          FilterChainCallback &&ccb)
{
    trantor::Date now=trantor::Date::date();
    LOG_TRACE<<"";
    if(req->session()->find(VDate))
    {
        auto lastDate=req->session()->get<trantor::Date>(VDate);
        LOG_TRACE<<"last:"<<lastDate.toFormattedString(false);
        req->session()->modify<trantor::Date>(VDate,
                                            [now](trantor::Date &vdate) {
                                                vdate = now;
                                            });
        LOG_TRACE<<"update visitDate";
        if(now>lastDate.after(10))
        {
            //10 sec later can visit again;
            ccb();
            return;
        }
        else
        {
            Json::Value json;
            json["result"]="error";
            json["message"]="Access interval should be at least 10
seconds";
            auto res=HttpResponse::newHttpJsonResponse(json);
            cb(res);
            return;
        }
    }
    LOG_TRACE<<"first access,insert visitDate";
    req->session()->insert(VDate,now);
    ccb();
}
```

We then register a lambda expression to the `/slow` path and attach the TimeFilter with the following code:

```
drogon::HttpAppFramework::instance()
            .registerHandler("/slow",
                            [=](const HttpRequestPtr &req,
                                std::function<void (const HttpResponsePtr
&)> &&callback)
                            {
                                Json::Value json;
                                json["result"]="ok";
                                auto
resp=HttpResponse::newHttpJsonResponse(json);
                                callback(resp);
                            },
                            {Get,"TimeFilter"});
```

Call the framework interface to open the session:

```
drogon::HttpAppFramework::instance().enableSession(1200);
```

Recompile the entire project with CMake, run the target program webapp, and you can see the effect through the browser.

Next: Database