Other languages: 繁體中文

Aspect Oriented Programming

AOP(Aspect Oriented Programming) is a programming paradigm that aims to increase modularity by allowing the separation of cross-cutting concerns(Quoted from Wikipedia).

Limited to the features of the C++ language, Drogon does not provide a flexible AOP solution like Spring, but a simple AOP in which all join points are predefined in the framework, and by using the framework's AOP series interfaces, one can register handlers(called 'advices' in Drogon) onto specific join points.

Predefined joinpoints

Drogon provides seven joinpoints for users. When the application runs to the joinpoints, the user-registered handlers(Advices) are called one by one. The description of these joinpoints is as follows:

- Beginning: As the name implies, the joinpoint is at the beginning of the program. Specifically, all handlers registered at this joinpoint are executed immediately after the app().run() method has finished initialization. At this joinpoint, all the controllers, filters, plugins, and database clients have been built completely, users can get the desired object reference or perform some other initialization work here. The advice on the joinpoint is only run once, the call signature of the advice is `void()`, the registration interface is `registerBeginningAdvice`;

- NewConnection: Advices registered to this joinpoint are called when each new TCP connection is established. The call signature of the advice is `bool(const trantor::InetAddress &, const trantor::InetAddress &)`, where the first argument is the remote address of the TCP connection and the second one is the local address. Note that the return type is bool, if the user returns false, the corresponding connection will be disconnected. The registration interface is `registerNewConnectionAdvice`;

- HttpResponseCreation: Advices registered to this joinpoint are called when each HTTP Response object is created. The call signature of the Advice is `void(const HttpResponsePtr &)`, where the parameter is the newly created object, and the user can perform some unified operations on all Responses with this joinpoint, such as adding a special header, etc. This joinpoint affects all Responses, including 404 or any drogon internal error response, and also including all responses generated by user's application. The registration interface is `registerHttpResponseCreationAdvice`;

- Sync: This joinpoint is located at the front end of Http request processing. Users can intercept this request by returning a non-empty Response object. The call signature of Advices is `HttpRequestPtr(const HttpRequestPtr &)`. The registration interface is `registerSyncAdvice'.

- Pre-Routing: Advices registered to this joinpoint are called immediately after the request is created and before it matches any handler paths. Advices for the joinpoint have two call signatures, `void(const HttpRequestPtr &,AdviceCallback &&,AdviceChainCallback &&)` and `void(const HttpRequestPtr &)`, the previous one is exactly the same as the call signature of the filter's `doFilter` method. In fact, they all run in the same way (please refer to [05-Filter]), users can intercept the client request or let it pass through this joinpoint. The advice with second call signature has no interception capability, but the overhead of it is lower, if the user does not intend to intercept requests, please select this kind of advices. The registration interface is `registerPreRoutingAdvice`;

- Post-Routing: Advices registered to this joinpoint are called immediately after the request matches a handler path, The call signatures of Advices are the same as the above joinpoint's. The registration interface is `registerPostRoutingAdvice`;

- Pre-Handling: Advices registered to this joinpoint are called immediately after the request is approved by all filters and before it is handled, The call signatures of Advices are the same as the above joinpoint's. The registration interface is `registerPostRoutingAdvice`;

- Post-Handling: Advices registered to this joinpoint are called immediately after the request is handled and a response object is created by the handler, The call signature of Advices is `void(const HttpRequestPtr &, const HttpResponsePtr &)`, The registration interface is `registerPostHandlingAdvice`;

AOP schematic

The following figure shows the location of the above four joinpoints in the HTTP Requests processing flow, where the red dots represent the joinpoints and the green arrows represent the asynchronous calls.

AOP

Next: Benchmarks