3-3HttpServer

2020年4月5日 8:50

HttpServer的内容是需要仔细的全部明白的.

HttpServer:
它的参数接受一个application工厂参数,这个参数必须满足Send+Sync.
绑定一个socket地址,使用bind()函数,它可能会被多次的调用.
要绑定一个ssl socket使用bind_openssl()或者bind_rustls().
要运行server使用,HttpServer::run().

Run()函数被调用后,因为在async中,server并不会立刻运行,这个时候,我们可以启动另外一个线程,进行server的创建,并要求System的创建和运行.
Let server = HttpServer::run();

Server.pause();
Server.resume();
Server.stop(true);

多线线程(workers):
一旦创建了多个workers,每一个workers分别接受一个单独的application的副本来处理请求.application state不会在多线程之间共享,每一个handlers处理的仅仅是自己的副本而已.

Keep-Alive:

actix能保持一段时间的连接来等待一个请求的数据.

.keep_alive(75)或.keep_alive(Some(75))或者.keep_alive(KeepAlive::TimeOut(75)) - 保持75秒的连接

.keep_alive(None)或者.keep_alive(KeepAlive::Disable) - 关闭keep alive

.keep_alive(KeepAlive::Tcp(75)) - 使用SO_KEEPALIVE socket的选项

Keep alive功能计时开始于响应连接类型被设置的那一刻:
默认的HttpResponse::connection_type没有被设置.
这种情况下,keep alive是不会开始计时的.也就是说keep alive功能不会启用.
所以在这种情况下,keep alive功能启动取决于请求的http的版本号.

HTTP/1.0 - keep alive不启用
HTTP/1.1和HTTP/2,0 - keep alive功能启用.

connection-type可以在响应中设置.

```
async fn index(req: HttpRequest) -> HttpResponse {
    HttpResponse::Ok()
        .connection_type(http::ConnectionType::Close) // <- Close connection
        .force_close() // <- Alternative method
        .finish()
} // 这样keep alive计时就会开始了
```

优雅的关机:
SIGINT - 强制关闭workers
SIGTERM - Graceful 关闭workers
SIGQIUT - 强制关闭