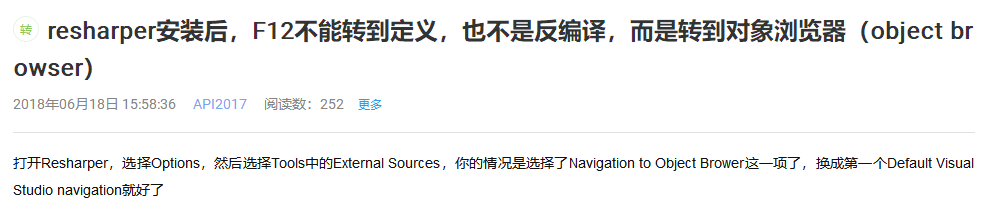
记录some

是

终于搞定了resharper



[http://localhost:3410/Image/Viewer?DicomDirPath=E:\work\ct-dhw\DICOMDIR](http://localhost:3410/Image/Viewer?DicomDirPath=E:/work/ct-dhw/DICOMDIR)

1. 访问：http://localhost:3410/Image/Viewer?DicomDirPath=E:\work\ct-dhw\DICOMDIR
2. 过滤器验证[TokenAuthentication]

是

综合点：

1. @Styles.Render("~/Front/style/ImageStyle")调用css
2. @Scripts.Render("~/bundles/ImageViewer")

是

异步问题：

看到：

你这是客户端异步调用，跟服务端关系不大。  
服务端调用使用异步一般都是微服务才会用到的，一个api调用其他api，这个时候用到异常效果就很好。一个独立的api基本用不到，直接同步就行了。

## **async & await**

如果使用await，在调用 await taskcall() 时不会阻塞get主方法线程，主方法线程会被释放，新的线程执行完成task后继续执行await后的代码减少线程切换开销，而之前的线程则空闲了。

[HttpGet]public async Task<string> Get(){

var info = string.Format("api执行线程:{0}", Thread.CurrentThread.ManagedThreadId);

var infoTask = await TaskCaller();

var infoTaskFinished = string.Format("api执行线程（task调用完成后）:{0}", Thread.CurrentThread.ManagedThreadId);

return string.Format("{0},{1},{2}", info, infoTask, infoTaskFinished);

}

private async Task<string> TaskCaller(){

await Task.Delay(500);

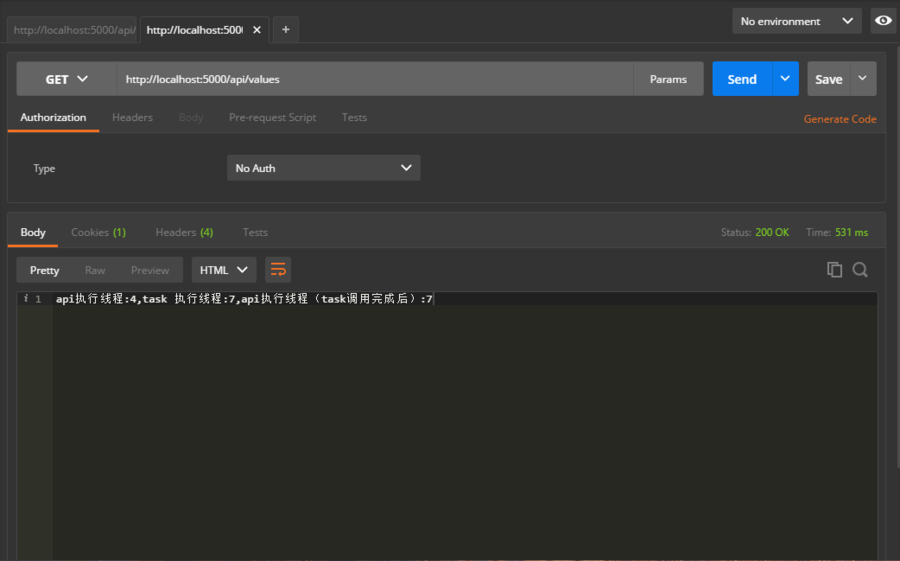
return string.Format("task 执行线程:{0}", Thread.CurrentThread.ManagedThreadId);

}

**代码执行流程**



**输出结果如下**



## **总结**

Task.result 与 await关键字 具有类似的功能可以获取到任务的返回值，但是本质上Task.result会让外层函数执行线程阻塞直到任务执行完成，而使用await关键字外层函数线程则不会阻塞，而是通过任务执行线程来执行await后的代码。

查找2:

## **性能提升四：async/await（异步方法）**

当在大型项目中会出现并发现象，常见的情况例如注册，此时有若干个用户同时在注册时，则会导致当前请求阻塞并且页面一直无响应最终导致服务器崩溃，为了解决这样的问题我们需要用到异步方法，让多个请求过来时，线程池分配足够的线程来处理多个请求，提高线程池的利用率 ！如下：

public async Task<IHttpActionResult> Register(Employee model)

{

var result = await UserManager.CreateAsync(model);

return Ok(result);

}

查找3：

异步返回普通对象：

WebAPI也是支持异步返回对象的：

    public async Task<string> Get()  
    {  
        await Task.Delay(100);  
        return "hello";  
    }

异步返回的时候，服务器异步等待函数执行完成后再将返回值返回给对象。由于这个过程对于客户端来说是透明的，这里就不列举报文了。

some追加：

比方获取Dicom文件，从Dicom这边的webapi进来。

第一个功能：获取缩略图压缩包

第二个功能：