NETCore提供了三种不同类型用于生产的REST API： `HttpWebRequest;WebClient;HttpClient`，开源社区创建了另一个名为`RestSharp的库`。如此多的http库，该怎样选择呢?

HttpWebRequest

这是.NET创建者最初开发用于使用HTTP请求的标准类。使用`HttpWebRequest`可以让开发者控制请求/响应流程的各个方面，如 timeouts, cookies, headers, protocols。另一个好处是`HttpWebRequest`类不会阻塞UI线程。例如，当您从响应很慢的API服务器下载大文件时，您的应用程序的UI不会停止响应。然而，强大的个性化操作带来了极大的复杂性。为了简单起见，`GET`您需要至少五行代码;

```
HttpWebRequest http = (HttpWebRequest)WebRequest.Create("
http://example.com
HttpWebRequest http = (HttpWebRequest)WebRequest.Create("
                WebResponse response = http.GetResponse();
                Stream stream = response.GetResponseStream();
                using (var streamtemn = File.Create("路径"))
                {
                    stream.CopyTo(streamtemn);
                }
```

如果对http协议不是了如指掌，使用**HttpWebRequest**会增加你的开发成本，除非你需要非常细节的处理和底层的控制，另外HttpWebRequest库已经过时，不适合业务中直接使用，他更适用于框架内部操作。

WebClient

`WebClient`是一种更高级别的抽象，是`HttpWebRequest`为了简化最常见任务而创建的，使用过程中你会发现他缺少基本的header，timeoust的设置，不过这些可以通过继承httpwebrequest来实现。使用`WebClient`可能比`HttpWebRequest`直接使用更慢（大约几毫秒）。但这种“低效率”带来了巨大的好处：它需要更少的代码和隐藏了细节处理，更容易使用，并且在使用它时你不太可能犯错误。同样的请求示例现在很简单只需要两行而且内部周到的处理完了细节：

```
using (WebClient webClient = new WebClient()) { webClient.DownloadFile("http://example.com", "路径");
}
```

HttpClient

`HttpClient`提供强大的功能，提供了异步支持，可以轻松配合async await 实现异步请求，具体使用可参考：NetCore 2.1中的HttpClientFactory最佳实践

RestSharp

restsharp是开源社区贡献，具有HttpWebRequest的细节控制和`WebClient`的使用简单的优点从而让他功能强大的同时又简化了操作(从他定义的接口可以看出真是一个优秀的`http`库啊😊)

结论

- HttpWebRequest 已经不推荐直接使用了，这已经作为底层机制，不适合业务代码使用
- WebClient 不想为http细节处理而头疼的coder而生，由于内部已经处理了通用设置，某些情况可能导致性能不是很理想
- RestSharp 兼具强大功能和友好api很适合业务中使用
- HttpClient 更加适用于异步编程模型中

参考：

https://stackoverflow.com/questions/22791376/is-httpwebrequest-or-webclient-faster/22792326#22792326

https://stackoverflow.com/questions/20530152/deciding-between-httpclient-and-webclient

https://social.msdn.microsoft.com/Forums/vstudio/en-US/2ce80a71-1ced-4bcd-adb4-88eef6e6a42d/httpclient-vs-httpwebrequest?forum=wcf

https://stackify.com/restsharp/?utm_referrer=https%3A%2F%2Fwww.google.com%2F