

09 | 学习开源代码该如何入手？

李玥 2019-08-10



00:00

11:42

讲述：李玥 大小：13.40M

你好，我是李玥。对于很多开源软件来说，如果我们把它作为我们业务系统的重要组成部分之一，真正地用于生产，仅仅知道如何使用是远远不够的，你必须掌握它的实现原理和很多细节，这样才能找到最佳的使用姿势，当你的系统出现问题时，你才有可能基于它的实现原理，再根据一些现象来排查问题原因。

掌握这些开源软件的最佳方式就是去学习它的源代码。很多同学跟我说：“我也很想去看看一些开源软件的代码，也尝试去看过，但是面对上千个源码文件，几十万行代码，完全不知道从哪儿入手啊。”

这节课我们就针对这个情况来聊一聊，学习开源软件的代码该如何入手。

有一点我提前说明一下，对于这节课里面涉及到的一些名词，我会直接使用英文，主要目的是方便你直接对应到那些开源软件英文官网上的标题。

通过文档来了解开源项目

学习源代码应该从哪儿入手呢？**最佳的方式就是先看它的文档。**

通过看文档，你可以快速地掌握这个软件整体的结构，它有哪些功能特性，它涉及到的关键技术、实现原理和它的生态系统等等。在掌握了这些之后，你对它有个整体的了解，然后再去看它的源代码，就不会再有那种盲人摸象找不到头绪的感觉了。

首先强调一点是，你必须去看这些开源软件官方网站上的文档，尽量不要去网上搜一些翻译的中文文档。为什么呢？

因为这些开源软件，特别是一些社区活跃的软件，它的迭代是很快的，即使是自带官方中文翻译的项目，它的中文文档很多都会落后于英文版，你能看到的中文版很多时候都已经过时了。那非官方的翻译，问题可能就不止是过时的问题了，可能还会出现一些错漏的地方。所以，最好还是直接来看官方的英文文档。

如果说你的英文阅读水平确实有限，直接阅读英文文档有困难或者看得非常慢，怎么办？你还是要按照我接下来告诉你方法去看它的英文官网，即使阅读大段的技术文章有困难，网站的标题你总能看懂吧？找到你需要阅读的文章后，你可以在网上搜一下对应的中文版本，先看一遍中文版，然后再对着英文原版过一遍，弥补中文版可能过时或翻译不准确的问题。

开源社区经过这么多年的发展，它已经形成一个相对比较成熟的文化。每个开源软件，代码如何管理、社区成员如何沟通、如何协作这些都已经形成了一个比较固定的套路。大多数开源软件，它的官网和技术文档也是有一个相对比较固定的结构的。

接下来我们以Kafka的官网为例子，来说下怎么来看它的文档。

如果说你对这个项目完全不了解，没用过这个软件，你首先需要看的文档是Quick Start，按照Quick Start中的指导快速把它的环境搭起来，把它运行起来，这样你会对这个项目有个感性认识，也便于你在后续深入学习的时候“跑”一些例子。

然后你需要找一下它的Introduction，一般里面会有项目的基本介绍。这里面很重要的一点是，你需要找到这个项目用到的一些基本概念或者名词的介绍文档，在Kafka的文档中，这些内容就在Introduction里面，比如Topic、Producer、Consumer、Partition这些概念在Kafka中代表的含义。

有些开源项目会单独有一个Basic Concepts文档来讲这些基础概念。这个文档非常重要，因为这些开源社区的开发者都有个很不好的爱好：发明概念。很多开源项目都会自己创造一些名词或者概念，了解这些基本概念才有可能看懂它项目的其他文档。

对项目有个基本的了解之后呢，接下来你可以看一下它的使用场景、功能特性以及相关的生态系统的介绍。在Kafka中功能相关的内容在Use cases和EcoSystem两篇文章中，有些项目中会有类似名为Features的文档介绍功能和特性。

其中项目的生态系统，也就是EcoSystem，一般会介绍它这个项目适用的一些典型的使用场景，在某个场景下适合与哪些其他的系统一起来配合使用等。如果说你的系统不是特别特殊或者说冷门的话，你大概率可以在EcoSystem里面找到和你类似的场景，可以少走很多的弯路。

你在读完上面这些文档之后，对这个项目的整体应该会有一个比较全面的了解了，比如说：

- 这个�项目是干什么的？
- 能解决哪些问题？
- 适合在哪些场景使用？
- 有哪些功能？
- 如何使用？

对这些问题有一个初步的答案之后，接下来你就可以去深入学习它的实现原理了。这是不是意味着，你可以立即去看它的源码呢？这样做或许可行，但并不是最好的方法。

你知道大部分开源项目都是怎么诞生的吗？一般来说是这样的：某个大学或者大厂的科学家，某天脑海里突然出现了一个改变世界的想法，科学家们会基于这个想法做一些深入的研究，然后写了一篇论文在某个学术期刊或者会议上发表。论文发表后在业内获得很多的赞，这时候就轮到像Google、Facebook这样的大厂出手了：这个论文很有价值，不如我们把它实现出来吧？一个开源项目就这样诞生了。

所以，对于这样的开源项目，它背后的这篇论文就是整个项目的灵魂，你如果能把这篇论文看完并且理解透了，这个项目的实现原理也就清楚了。

对于Kafka来说，它的灵魂是这篇博文：[The Log: What every software engineer should know about real-time data's unifying abstraction](#)，对应的中文译稿在这里：《[日志：每个软件工程师都应该知道的有关实时数据的统一抽象](#)》。

这篇博文被评为[程序员史诗般必读文章](#)，无论你是不是想了解Kafka的实现原理，我都强烈推荐你好好读一下上面这篇博文。

学习完项目灵魂，就可以开始阅读源码了。

用以点带面的方式来阅读源码

需要注意的是，你在读源码的时候，千万不要上来就找main方法这样泛泛地去看，为什么？你可以想一下，一篇文章，它是一个线性结构，你从前往后读就行了。一本书呢？如果我们看目录的话，可以认为是个树状结构，但大多数的书的内容还是按照线性结构来组织的，你可以从前往后读，也可以通过目录就着读。

那程序的源代码是什么结构？那是一个网状结构，关系错综复杂，所以这种结构是非常不适合人类去阅读的。你如果是泛泛去读源代码，很容易迷失在这个代码织成的网里面。那怎么办？

我推荐大家阅读源码的方式是，**带着问题去读源码，最好是带着问题的答案去读源码**。你每次读源码之前，确定一个具体的问题，比如：

- RocketMQ的消息是怎么写到文件里的？
- Kafka的Coordinator是怎么维护消费位置的？

类似这种非常细粒度的问题，粒度细到每个问题的答案就是一两个流程就可以回答，这样就可以了。如果说你就想学习一下源代码，或者说提不出这些问题怎么办呢？答案还是，**看文档**。

确定问题后，先不要着急看源代码，而是应该先找一下是否有对应的实现文档，一般来说，核心功能都会有专门的文档来说明它的实现原理，比如在Kafka的文档中，DESIGN和IMPLEMENTATION两个章节中，介绍了Kafka很多功能的实现原理和细节。一些更细节的非核心的功能不一定有专门的文档来说明，但是我们可以去找一找是否有对应的Improvement Proposal。（Kafka的所有Improvement Proposals在[这里](#)。）

这个Improvement Proposal是什么呢？你可以认为它是描述一个新功能的文档，一般开源项目需要增加一个新的功能或者特性的时候，都会创建一个Improvement Proposal，一般标题都是“xIP-新功能名称”，其中IP就是Improvement Proposal的缩写，x一般就是这个开源项目的名称的首字母，比如Kafka中Improvement Proposal的标题都是以KIP开头。

每个Improvement Proposal都是有固定格式的，一般要说明为什么需要增加这个功能，会对系统产生那些影响和改变，还有我们最关心的设计和实现原理的简述。

你读完讲解实现的文档再去查看源代码，也就是我刚刚说的，不只是带着问题去读，而是带着答案去读源码。这样你在读源码的时候，不仅会是更容易理解源代码，还可以把更多的精力放在一些实现细节上，这样阅读源码的效果会更好。

使用这种以问题为阅读单元的方式来读源代码，你每次只要花很短的时间，阅读很少的一部分源码，就能解决一个问题，得到一些收获。这种方式其实是通过一个一个问题，在网状的源代码中，每次去读几个点组成的那一两条线。随着你通过阅读源码了解的问题越来越多，你对项目源码的理解也会越来越全面和深入。

小结

如果你想了解一个开源项目，学习它的代码，最佳的切入点就是去读它的官方文档，这些文档里面，最重要的灵魂就是项目背后的那篇论文，它一般是这个开源项目的理论基础。

在阅读源码的时候呢，最佳的方式是带着问题去阅读，最好是带着问题的答案去读，这样难度低、周期短、收获快。不要想着一定要从总体上去全面掌握一个项目的所有源代码，也没有必要。

思考题

课后，建议你找一个你熟悉的开源项目，可以是消息相关的，也可以是无关的开源项目，确定一个问题，用这节课中讲到的“带着问题和答案去读源码”的方法，去读一点源码。然后，最重要的是，把主要的流程用流程图或者时序图画出来，把重点的算法、原理用文字写出来。

欢迎你在留言区分享你的阅读源码的笔记，在这个过程中，如果遇到任何问题，你也可以在留言区提问。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给你的朋友。

极客时间

消息队列高手课

从源码角度全面解析MQ的设计与实现

李玥

京东零售技术架构部资深架构师

新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有现金奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

志恒Z

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。

Ctrl + Enter 发表 0/2000字

提交留言

精选留言(24)

芥末小龙

玥哥，以前带我的时候咋不提呢，如果知道早知道这样看源码，知道的原理多一些，我估计阿里P7就能过了。

作者回复：你提问呀

2019-08-12

40

4

linqw

学习完这篇写下自己的理解，老师有空帮忙看下呗
1、在还没阅读开源项目代码之前，会先把Java的一些常用的源代码先看，感觉所有的源代码都是这些基础的块块叠加上去的，最近也在看rocketmq源码，也是先把用到的一些其他框架先熟悉如netty、logger，然后看其官方文档，对整体的框架和基本的概念有比较感性的认知，再者写一些demo，然后先找rocketmq中的公共的模块的源代码先看，比如logger、logappend、namesrv，如namesrv内部维护broker、cluster、topic、queue的路由信息，producer再发送消息前，需先将namesrv broker的路由信息缓存到本地，为此需和namesrv维护长连接，定时的从namesrv获取新的路由信息，broker端需和namesrv为此常连接，broker定时的发送配置信息到namesrv，namesrv需开启定时任务将其超过时间没有收到心跳包的broker连接关闭。为了namesrv中的配置参数可以更改，可以从System中的properties中获取，也为了这些配置持久化也会在接收到配置参数时保存到文件中，namesrv需维护配置参数的设置获取，如发送消息前选择topic等，为了区分不同的命令，维护RequestProcessor，对接收到不同的RemotingCommand做不同的处理序列化和反序列化，参数检验等。很多开源框架都会做一个admin，便以实时的更改配置参数，实时的监控项目中的配置参数。
2、打算先把并发包先分析完，然后开始分析下rocketmq，这个是自己分析的一些知识点https://juejin.im/user/5bd8718051882528382d8728/posts

作者回复：👍👍👍

2019-08-10

2

innocent

您好，请问一下作为一个对Scala不是很熟悉的Java程序员如何阅读Kafka的源码。

作者回复：花点时间学一下基础语法，特别是lambda，至少读源码就没什么问题了。

2019-08-10

1

2

纯齐

请教下，有什么好办法找出开源项目的“灵魂”？

作者回复：建议去从文档中找。

如果这个�项目是基于某个Paper的，一般文档中的显著位置都会标注。

2019-08-12

1

陈琪

老师在后续课程是否可以再开放一些“程序员史诗般必读文章”给我们学习下

2019-08-12

1

陈华应

读源码的习惯是喜欢按照一条主线来看，经常会被一些细节绕晕。多谢老师指路。
后续就按照：灵魂->官方文档->问题&答案找答案->核心类结构图+流程图->原理细节文字步骤说明->点成线->线成网->实际应用（可遇不可求）->官方文档->灵魂
多写总结，多思考知识点之间多关联，多动手，一切都是水到渠成的事儿~~~

作者回复：👍👍👍

2019-08-11

1

知非

这篇文章，解决我很大的疑惑啊！自己看代码就是先找main，真尴尬。。

2019-08-10

1

Harry

老师说的确实不错，终于看到源码阅读方式了很赞👍

2019-08-12

easy

看源码还是习惯于看整体流程，接着看自己关注的某几个模块，随后根据需去看相关模块

2019-08-12

吴宇晨

带着问题读源码👍

2019-08-12

DeathKnightH

以前看源码都是直接quick start跑个示例，然后就去查reference和API了，然后遇到问题就去找问题相关的源码位置，完全没有看过EcoSystem和Use cases，也没看过开源项目背后的理论基础，怪不得每次看完就忘🙄记笔记都没啥用。

2019-08-12

海罗沃德

看Spring源码时候都是在IDE里直接点到Spring某个类具体到方法上，然后打上断点，运行时一层一层debug看，不过仅限于Spring的框架，Kafka这样的中间件不在你自己项目里的就没法这么看了

2019-08-12

miniluo

开发使用开源框架遇到一些比较棘手问题就会看相关的实现的源代码。

2019-08-10

Knight²⁰¹⁸

我最大的问题是无法把看到得，学到得行文档

2019-08-10

苦行僧

带着问题debug

2019-08-10

谢清

授之以鱼，也授之以渔👍👍👍

2019-08-10

miofy

带着问题看源码类似于监督学习，不带问题看源码类似于非监督学习，两者各有场景。碰到问题查源码是前者，没碰到问题想总结用后者。

2019-08-10

王莹

看完本章受益匪浅，最近读源码尝试实践在本地Quick Start，然后直接Debug定位核心主要流程，再划分一个小小的问题域，Debug走读代码。

2019-08-10

许童童

带着问题和答案去读源码，并结合官方文档，实在不行你还可以跟着别人的源码解析文章去读源码。读源码的过程的确是很享受的，有时读到一些精髓，甚至会大呼：哦~原来是这样。
我读源码的时候 喜欢新建一个文本，遇到重点的流程和跳转就记录到这个文本里面，就不会会走迷宫一样迷失在各种调用中。

2019-08-10

null

渐渐认识到，阅读原文文档是特别重要的事，虽然起初看起来很吃力，但基本上某一个技术领悟经常的的单词也是有限的，多看多记，稳

2019-08-10

游弋云端

“日志：每个软件工程师都应该知道的有关实时数据的统一抽象”，这个之前阅读过，当时就觉得写得非常好，被老师推荐为灵魂文章了。

2019-08-10

盛

学习了

2019-08-10