课题：开源

作者：张舒来

Form: NGBF Kingstar

2014-3-31

[一、开源介绍 3](#_Toc7192)

[1.1 开源概念 3](#_Toc14274)

[1.2 开源历史 3](#_Toc31557)

[1.3 开源提倡组织 4](#_Toc14574)

[1.4 开源定义(Open Source Definition) 4](#_Toc17483)

[1. 自由的再发包 4](#_Toc32347)

[2. 源代码 5](#_Toc39)

[3. 衍生产物 5](#_Toc23931)

[4. 原作者源代码的完整健全性 5](#_Toc1348)

[5. 不可以区别对待个人和团队 5](#_Toc22220)

[6. 不可以区别对待软件的使用领域 5](#_Toc12028)

[7. 证书的分发 5](#_Toc2502)

[8. 证书不可以对某个特殊产品有特别规定 6](#_Toc12869)

[9. 证书不可以限制其他软件 6](#_Toc9571)

[10. 证书必须保持技术中立 6](#_Toc166)

[二、常见证书介绍 6](#_Toc8631)

[2.1 BSD(original BSD license、FreeBSD license) 6](#_Toc29667)

[2.2 MIT 7](#_Toc13169)

[2.3 ASL(Apache License， Version 2.0、Apache License， Version 1.1、Apache License， Version 1.0) 7](#_Toc4102)

[2.4 MPL开源协议(The Mozilla Public License) 8](#_Toc12347)

[2.5 CDDL（Common Development and Distribution License） 8](#_Toc22189)

[2.6 CPL(Common Public License) version 1.0 9](#_Toc4952)

[2.7 GPL(GNU General Public License) 9](#_Toc15810)

[2.8 LGPL(GNU Lesser General Public License) 10](#_Toc3057)

[2.9 AGPL(GNU Affero General Public License) 11](#_Toc11534)

[3.0 总结： 11](#_Toc2790)

[三、开源产品介绍 11](#_Toc16709)

[3.1 Android 12](#_Toc13351)

[3.2 Spring Framework 12](#_Toc18549)

[五、NGBF 第三方开源引用 14](#_Toc30689)

[六、总结 15](#_Toc19207)

[6.1 开源的必然性 16](#_Toc31236)

[6.2 开源代码的类型和开源的收益 16](#_Toc19890)

[6.3 开源项目的盈利模式 16](#_Toc25315)

[6.4 开源社区的建设 16](#_Toc7182)

# 一、开源介绍

## 1.1 开源概念

在工业生产中，开源是很多发展模式中的一种。

它使得一个产品的设计和蓝图能在一个自由证书的授权下，被所有人接触到。

进而，任何人都可以在此设计和蓝图的基础上进行连续的改进，并再次分发。学者们将开源看作是开发合作(Open Collaboration)的一种进化模式。

Open Collaboration：

指任何的生产和创新系统，其依赖的参与者是基于全球的，但参与者们相互又是在组织上较为宽松的关系，并且通过广泛交流来创建具有商业价值的产品或服务。他们将产出的产品或服务提供给所有的参与者和非参与者。

开源随着互联网的发展而被广泛接受，因为开发者意识到大量的代码在不同项目中被重复开发。开放源码使得自身的生产模式，交流途径和交流社区的多样性得到增强。

通常，开源，指的是，一个软件程序，其源代码被开放给普通公众使用或者基于其最初的设计上进行修改。开源，是作为专有软件的回应而产生的。

开源不局限于计算机行业，这种商业模式同样被用在食谱，太阳光电技术和药品研发中。

## 1.2 开源历史

分享科技信息的概念出现在比互联网和个人计算机更早的年代。

例如，在汽车行业的早些年里，二冲程汽油引擎的专利(最初由George B. Selden起草)被一些大资本家控制。凭借控制专利，他们垄断了整个行业，并迫使汽车生产商满足他们的要求，否则就会面临起诉。在1911年，亨利福特挑战了Selden专利，导致该专利变得几乎毫无价值，并催生了汽车制造商协会。这个机构在全美汽车制造商中提出了一个协议：每个公司都可以继续开发新技术并起草专利，但是这些专利被开放分享给所有制造商，并不需要收取任何的费用。在那时，美国正处于二战中，期间，92个福特专利和515个其他公司专利被无偿地分享给全美的汽车制造商。

和开源标准十分类似的，当年有权利接触美国高等研究计划署网络(Advanced Research Projects Agency Network, ARPAnet)[[1]](#footnote-0)的研究院通过修正意见书征求(Request For Comments)来开发电信网络协议。1960年代的这种合作流程带来的1969互联网的诞生。

早年源码自由分享的例子，还包括IBM在50和60年代发出它的操作系统及其他程序的源码，以及SHARE[[2]](#footnote-1)组织的建立，促进软件技术的交换。

程序代码，在1980年代，被提供在BBS网络上。那时候开源有时也是一种必然。因为很多分发的软件由BASIC或者其他被转译的语言写出来，只能源码分发，并没有分离出可执行的二进制文件。

早年，在Usenet上，开发者按BSD协议来将分发代码分发到comp.os.linux上，也是在comp.os.linux上，BSD协议的发展被广泛讨论。Linux引用了这种模式。

大多数的经济学家认同开源产品有对信息(知识)有利的一面。它使得对已有产品的再开发的成本处于临界值(非常低)。基础经济学原理上来说，低成本意味着更多的消费和更多的迭代更新。但也有人认为，开源会导致最初的创造者没法收回最初开发的成本，从来没有经济动力再进行创新。最终导致消费者无产品可消费。

## 1.3 开源提倡组织

Open Source Initiative(开源提倡组织)，OSI是一个1998年成立在加利福尼亚的公益组织，享有501(c)3税额豁免的权利。

他们是OSD[[3]](#footnote-2)的管理者，和在开源领域被广泛认可的开源证书审查和认可的机构。

但是，他们所做的不局限于此，他们积极参与所有和开源相关的事件，并给所有从事开源软件事业的人提供咨询和帮助。

## 1.4 开源定义(Open Source Definition)

### 1. 自由的再发包

如果一个给于开源证书的软件，被用作一个整合性软件发包的一个组件，并且这个整合性软件中含有许多来自不同来源的程序，则这个开源证书不应该限制任何机构把该软件作为一个组件进行销售或赠与。在软件销售过程中，这个许可不应要求忠诚或者索要另外的费用。

### 2. 源代码

程序必须含有源代码，必须有源代码发包和编译后的发包。如果一些产品的发包中没有源码，那么它必须提供一个公众熟知的渠道，通过不多于一个合理的复制的费用，或者免费的在线下载，来获取源代码。源代码必须是一个可以让程序员进行修改的合适的格式。蓄意地弄乱源代码是被禁止的。一些中间产物的格式，例如预处理和翻译的输出内容是不被认可的。

### 3. 衍生产物

许可必须允许修改和衍生产物，并必须允许他们拥有和原软件同样的许可。

### 4. 原作者源代码的完整健全性

证书不能禁止对源码的修改发包，除非证书允许给源码打补丁发包，已达到在构建时修改程序的目的。证书必须明确地指出允许用修改的源代码来进行构建软件。证书可能要求衍生产品使用和原软件不同的名字和版本号。

### 5. 不可以区别对待个人和团队

证书不可以区别对待个人和团队，即证书对待所有个人和团队的效用都必须一致。

### 6. 不可以区别对待软件的使用领域

证书不应该限制任何人在其专注领域运用该软件。例如， 证书不可以限制软件被商用，也不可以限制软件被用于基因研究。

### 7. 证书的分发

赋予软件的权利必须被授于所有该软件被分发的组织和个人，而不需要该组织或个人申请额外的许可。

### 8. 证书不可以对某个特殊产品有特别规定

赋予软件的权利不应该因为其在不同的分发环境下而有所不同。例如，它不能因为是作为某个特定软件的分包组件而有其他特殊的规定或者权利。如果这个程序是从另一个分包中被提取出来的，并遵循同样的许可证书被使用和分发，所有再分发的对象（组织或者个人）都享受和原发包同样的证书权利。

### 9. 证书不可以限制其他软件

证书不能对和该软件一起供应的其他软件进行限制。例如，证书不能要求和该开源软件一起布置在同一个媒介上的其他软件也必须开源。

### 10. 证书必须保持技术中立

证书不应该有条款针对特殊的技术或者接口风格。

# 二、常见证书介绍

## 2.1 BSD(original BSD license、FreeBSD license)

BSD开源协议是一个给予使用者很大自由的协议.基本上使用者可以“为所欲为”可以自由的使用,修改源代码,也可以将修改后的代码作为开源或者专有软件再发布.但“为所欲为”的前提当你发布使用了BSD协议的代码,或则以BSD协议代码为基础做二次开发自己的产品时,需要满足三个条件:

1. 如果再发布的产品中包含源代码,则在源代码中必须带有原来代码中的BSD协议.

2. 如果再发布的只是二进制类库/软件,则需要在类库/软件的文档和版权声明中包含原来代码中的BSD协议.

3. 不可以用开源代码的作者/机构名字和原来产品的名字做市场推广.

其实这几个规则约定的目的也只是达到一个目的:是他人的东西,别人以BSD开源了,你就不能不做任何声明而占为己有,更不能用他人的名义来做商业推广.你只对你自己的东西拥有绝对控制权.

举个例子,你用开源代码(A)修改或做其他增添之后,产生了产品B,这时候,你对B的控制由你自己决定,你可以用任何协议再开源,也可以闭源商业发布.但,因为如果B中包含了A或A的一部分(一点都不包含就不叫修改了),那你在B产品的版权声明中,必须有提到你有使用到 A ,并且附带上 A 的开源协议.而且不能做商业推广的时候将B 冠以原开源作者的名义以促进商业推广.

BSD代码鼓励代码共享,但需要尊重代码作者的著作权.BSD由于允许使用者修改和重新发布代码,也允许使用或在BSD代码上开发商业软件发布和销售,因此是对商业集成很友好的协议.而很多的公司企业在选用开源产品的时候都首选BSD协议,因为可以完全控制这些第三方的代码,在必要的时候可以修改或者二次开发.

**小结:商业软件可以使用，也可以修改使用BSD协议的代码。**

## 2.2 MIT

MIT 协议可能是几大开源协议中最宽松的一个，核心条款是：

该软件及其相关文档对所有人免费，可以任意处置，包括使用，复制，修改，合并，发表，分发，再授权，或者销售。唯一的限制是，软件中必须包含上述版权和许可提示。

这意味着：你可以自由使用，复制，修改，可以用于自己的项目。可以免费分发或用来盈利。唯一的限制是必须包含许可声明。MIT 协议是所有开源许可中最宽松的一个，除了必须包含许可声明外，再无任何限制。

**小结： 商业软件可以使用，也可以修改使用MIT协议的代码。**

## 2.3 ASL(Apache License， Version 2.0、Apache License， Version 1.1、Apache License， Version 1.0)

Apache License是著名的非盈利开源组织Apache采用的协议.该协议和BSD类似,同样鼓励代码共享和尊重原作者的著作权,同样允许代码修改,再发布(作为开源或商业软件).需要满足的条件也和BSD类似:

1. 需要给代码的用户一份Apache License

2. 如果你修改了代码,需要再被修改的文件中说明.

3. 在延伸的代码中(修改和有源代码衍生的代码中)需要带有原来代码中的协议,商标,专利声明和其他原来作者规定需要包含的说明.

4. 如果再发布的产品中包含一个Notice文件,则在Notice文件中需要带有Apache License.你可以在Notice中增加自己的许可,但不可以表现为对Apache License构成更改.

**小结：Apache License也是对商业应用友好的许可.使用者也可以在需要的时候修改代码来满足需要并作为开源或商业产品发布/销售.**

## 2.4 MPL开源协议(The Mozilla Public License)

MPL是The Mozilla Public License的简写，是1998年初Netscape的 Mozilla小组为其开源软件项目设计的软件许可证。MPL许可证出现的最重要原因就是，Netscape公司认为GPL许可证没有很好地平衡开发者对源代码的需求和他们利用源代码获得的利益。同著名的GPL许可证和BSD许可证相比，MPL在许多权利与义务的约定方面与它们相同（因为都是符合OSIA 认定的开源软件许可证）。但是，相比而言MPL还有以下几个显著的不同之处:

MPL虽然要求对于经MPL许可证发布的源代码的修改也要以MPL许可证的方式再许可出来，以保证其他人可以在MPL的条款下共享源代码。但是，在MPL 许可证中对“发布”的定义是“以源代码方式发布的文件”，这就意味着MPL允许一个企业在自己已有的源代码库上加一个接口，除了接口程序的源代码以MPL 许可证的形式对外许可外，源代码库中的源代码就可以不用MPL许可证的方式强制对外许可。这些，就为借鉴别人的源代码用做自己商业软件开发的行为留了一个 豁口。 MPL许可证第三条第7款中允许被许可人将经过MPL许可证获得的源代码同自己其他类型的代码混合得到自己的软件程序。 对软件专利的态度，MPL许可证不像GPL许可证那样明确表示反对软件专利，但是却明确要求源代码的提供者不能提供已经受专利保护的源代码（除非他本人是 专利权人，并书面向公众免费许可这些源代码），也不能在将这些源代码以开放源代码许可证形式许可后再去申请与这些源代码有关的专利。 对源代码的定义 而在MPL（1.1版本）许可证中，对源代码的定义是:“源代码指的是对作品进行修改最优先择 取的形式，它包括:所有模块的所有源程序，加上有关的接口的定义，加上控制可执行作品的安装和编译的‘原本’（原文为‘Script’），或者不是与初始 源代码显著不同的源代码就是被源代码贡献者选择的从公共领域可以得到的程序代码。” MPL许可证第3条有专门的一款是关于对源代码修改进行描述的规定，就是要求所有再发布者都得有一个专门的文件就对源代码程序修改的时间和修改的方式有描述。

**小结：商业软件可以使用，也可以修改MPL协议的代码，但修改后的代码版权归软件的发起者。**

## 2.5 CDDL（Common Development and Distribution License）

CDDL（Common Development and Distribution License，通用开发与销售许可）开源协议，是MPL（Mozilla Public License）的扩展协议，它允许公共版权使用，无专利费，并提供专利保护，可集成于商业软件中，允许自行发布许可。

**小结：商业软件可以使用，也可以修改CDDL协议的代码。**

## 2.6 CPL(Common Public License) version 1.0

CPL是IBM 提出的并通过了OSI(Open Source Initiative)批准的开源协议.主要用于一些IBM或跟IBM相关的开源软件/项目中.如很著名的Java开发环境 Eclipse 、RIA开发平台Open Laszlo等.

CPL也是一项对商业应用友好的协议.它允许 Recipients 对源码进行任意的使用、复制、分发、传播、展示、修改以及改后做闭源的二次商业发布,这点跟 BSD 很类似,也属于自由度比较高的开源协议.但是,需要遵循:

1. 当一个Contributors将源码的整体或部分再次开源发布的时候,必须继续遵循 CPL开源协议来发布,而不能改用其他协议发布.除非你得到了原“源码”Owner 的授权.

2. CPL协议下,你可以将源码不做任何修改来商业发布.但如果你要将修改后的源码其开源,而且当你再发布的是Object Code的时候,你必须声明它的Source Code 是可以获取的,而且要告知获取方法.

3. 当你需要将CPL下的源码作为一部分跟其他私有的源码混和着成为一个 Project 发布的时候,你可以将整个Project/Product 以私人的协议发布,但要声明哪一部分代码是CPL下的,而且声明那部分代码继续遵循CPL.  
4. 独立的模块(Separate Module),不需要开源.

**小结：商业软件可以使用，也可以修改Common协议的代码，但要承担代码产生的侵权责任。**

## 2.7 GPL(GNU General Public License)

我们很熟悉的[Linux](http://www.ha97.com/category/linux" \o "Linux)就是采用了GPL。GPL协议和BSD，Apache Licence等鼓励代码重用的许可很不一样。GPL的出发点是代码的开源/免费使用和引用/修改/衍生代码的开源/免费使用，但不允许修改后和衍生的代码做为闭源的商业软件发布和销售。这也就是为什么我们能用免费的各种[linux](http://www.ha97.com/tag/linux" \o "linux)，包括商业公司的linux和linux上各种各样的由个人，组织，以及商业软件公司开发的免费软件了。

GPL协议的主要内容是只要在一个软件中使用(”使用”指类库引用，修改后的代码或者衍生代码)GPL协议的产品，则该软件产品必须也采用GPL协议，既必须也是开源和免费。这就是所谓的”传染性”。GPL协议的产品作为一个单独的产品使用没有任何问题，还可以享受免费的优势。

GPL协议最主要的几个原则：

1、确保软件自始至终都以开放源代码形式发布，保护开发成果不被窃取用作商业发售。任何一套软 件，只要其中使用了受 GPL 协议保护的第三方软件的源程序，并向非开发人员发布时，软件本身也就自动成为受 GPL 保护并且约束的实体。也就是说，此时它必须开放源代码。

2、GPL 大致就是一个左侧版权（Copyleft，或译为“反版权”、“版权属左”、“版权所无”、“版责”等）的体现。你可以去掉所有原作的版权 信息，只要你保持开源，并且随源代码、二进制版附上 GPL 的许可证就行，让后人可以很明确地得知此软件的授权信息。GPL 精髓就是，只要使软件在完整开源 的情况下，尽可能使使用者得到自由发挥的空间，使软件得到更快更好的发展。

3、无论软件以何种形式发布，都必须同时附上源代码。例如在 Web 上提供下载，就必须在二进制版本（如果有的话）下载的同一个页面，清楚地提供源代码下载的链接。如果以光盘形式发布，就必须同时附上源文件的光盘。

4、开发或维护遵循 GPL 协议开发的软件的公司或个人，可以对使用者收取一定的服务费用。但还是一句老话——必须无偿提供软件的完整源代码，不得将源代码与服务做捆绑或任何变相捆绑销售。

其它细节如再发布的时候需要伴随GPL协议等和BSD/Apache等类似。

因为要完全开源代码，不适合公司商业用途。由于GPL严格要求使用了GPL类库的软件产品必须使用GPL协议，对于使用GPL协议的开源代码，商业软件或者对代码有保密要求的部门就不适合集成/采用作为类库和二次开发的基础。

**小结： 商业软件不能使用GPL协议的代码。**

## 2.8 LGPL(GNU Lesser General Public License)

LGPL是GPL的一个为主要为类库使用设计的开源协议。和GPL要求任何使用/修改/衍生之GPL类库的的软件必须采用GPL协议不同。LGPL允许商业软件通过类库引用(link)方式使用LGPL类库而不需要开源商业软件的代码。这使得采用LGPL协议的开源代码可以被商业软件作为类库引用并发布和销售。但是如果修改LGPL协议的代码或者衍生，则所有修改的代码，涉及修改部分的额外代码和衍生的代码都必须采用LGPL协议。因此LGPL协议的开源代码很适合作为第三方类库被商业软件引用，但不适合希望以LGPL协议代码为基础，通过修改和衍生的方式做二次开发的商业软件采用。

GPL/LGPL都保障原作者的知识产权，避免有人利用开源代码复制并开发类似的产品

**小结： 商业软件可以使用，但不能修改LGPL协议的代码。**

## 2.9 AGPL(GNU Affero General Public License)

AGPL是GPL的一个补充, 在GPL的基础上加了一些限制。AGPL这个协议的制定是为了避免一个GPL/LGPL协议中的漏洞，称之为 Web Service Loopwhole。这主要是由于 GPL是针对传统的软件分发模式的商业模式(以微软为代表), 如果你使用的GPL的代码作为基础完成你自己的软件，如果你要分发你的软件，你的软件必须也是GPL的。随着以Google为代表的软件  
作为服务的互联网公司的兴起，它们的“不分发软件，为客户提供网络服务”的商业模式就不受GPL协议的约束，所以Google公司在构筑他的搜索引擎的时候可以随心所欲的拿现有的GPL协议的开源代码，无需开源他的修改成果。AGPL协议在GPL协议的基础上加上了这个约束。

**小结： 商业软件不能使用AGPL协议的代码。**

## **3.0 总结**：

按照使用条件的不同，开源软件许可证可以分为三类（严苛程度递减）

1． 使用该开源软件的代码再散布（redistribute）时，源码也必须以相同许可证公开。

代表许可类型：GPL， AGPL。

2． 使用该开源软件的代码并且对开源代码有所修改后再散布时，源码必须以相同许可证公开。

　　代表许可类型：LGPL， CPL，CDDL， CPL，MPL等

　　3． 使用该开源软件的代码（包括修改）再散布（redistribute）时，没有特殊限制，只需要明记许可。

代表许可类型：ASL， BSD，MIT等

# 三、开源产品介绍

## 3.1 Android

Android是一个基于Linux内核的操作系统，主要为触屏移动设备，如智能手机和平板电脑，设计。最初有Android Inc. 开发，由谷歌提供经济支持并与2005被谷歌收购。2007年，Android正式问世。同时问世的，还有一个叫Open Handset Alliance(开源手持联盟)的联盟组织，一些硬件、软件和电信巨头公司建立的致力于建设移动设备的开源标准的组织。第一个面向大众的安装Android系统的只能手机是HTC Dream，在2008年末发布。

Android的源码由谷歌基于Apache证书开源。这个宽松的证书保证了软件可以被设备生产商、电信供应商和有热情的开发者自由地修改和分发。大多Android设备发包同时包括开源部分和专利部分。2013年7月，Android成为拥有应用最多的平台，在Google Play商店中，有一百万个应用被提供下载，同时总下载次数突破了500亿。2013年4月到5月的一份开发者调查报告显示，Android是开发者用得最多的平台，它被71%的移动开发者使用。

Android受到那些需要已完成、低成本和高自定义操作系统的科技公司青睐，用于开发高科技产品。尽管主要为智能手机和平板设备设备，它同样被用于电视、游戏机、数码相机和其他电子设备中。Android的开源特性鼓励大批的社区开发者和爱好者用开源代码来作为社区驱动项目的基础，为高级用户增加新的特性，或将Android带进其他原本跑在其他系统上的设备中。

在2012年5月，Android成为最流行的移动操作系统，有最多的安装基数，并且是很多国家(包括美国)的市场领跑者。2013年第三季度，Android的全球市场份额是81.3%，世上最高。 2013年9月，10亿Android设备被激活。

## 3.2 Spring Framework

Spring框架是一个基于JAVA平台的，开源的项目框架和IoC(控制反转，Inversion of Control)容器。这个框架的核心特性可以被任何的JAVA程序应用，但它为WEB应用做的拓展需要基于JAVAEE平台。尽管该框架没有强加一些特殊的代码模型，它却成为EJB(Enterprise JavaBean)模型的有力替代者或补充。

初版由Rod Johnson编写，并与他撰写的Expert One-on-One J2EE Design and Development一书一起在2002年10月发布。第一个正式版在2003年6月，以Apache2.0证书发布。第一个标志性版本1.0，在2004年3月发布。在04年9月和05年3月又先后发布了另2个标志性版本。Spring 1.2.6在06年赢得了Jolt Productivity Award和 Jax Innovation Award 两项大奖。Spring 2.0 在06年10月发布，然后2.5 在07年11月， 3.0在09年12月， 3.1在11年12月，3.2.5在13年11月发布。最新的4.0发布于13年12月。

Spring的一些设计理念被JAVA开发者广泛认可，包括IoC控制反转，AOP(面向层的编程，Aspect-Oriented Programming)等。这些理念都逐步被JAVAEE标准吸收，成为一种JAVA规范。

四、开源社区--Github介绍

Github是一个网站式的为软件开发提供基于GIT的版本控制和团队协作的项目保存系统。它同时提供有偿的私人库服务和免费的开源库服务。最初由Tom Preston-Werner, Chris Wanstrath及PJ Hyett在08年建立。

它是现在全球最大的开源社区。截至2013年12月23日，官方统计超过1千万个工作库坐落于Github上。

页面主要分为3部分：

1. 内容展示区：

根据选择不同的功能TAB，在该区域展示不同的内容详情。

1. 功能TAB及按钮区：

2.1 Code： 内容展示区会展示源码文件结构和内容。

2.2 Issue: 类似JIRA系统，其他用户可以就该库内容进行提问和BUG报告，该库的管理员可以进行解答或修复BUG后关闭Issue。

2.3 Pull Request： 当用户想要参与到某个开源项目的开发中，可以先复制该项目到自己的库中，做了相应修改后，可以提出Pull Request，系统会自动匹配出修正版和原版的区别再加上修正作者的注释提交给原项目管理员，由原项目管理员确认是否要将改动融合入原项目中。

2.4 Wiki： 关于该工作库中项目的介绍，功能介绍，使用说明等任何相关的说明都可以在wiki中呈现出来。

2.5 Pulse&Graphs&Network： 一些关于该工作库状态的图表统计，包括最近融合的Pull Request和待审核的Pull Request统计，Issue统计，贡献者活跃度统计，代码提交统计等；

2.6 Clone&Download：Github提供三种源码下载方式：1. 基于Https的git clone；2. 基于GITHUB客户端的clone； 3. ZIP包下载。

1. 右上按钮：Watch， Star， Fork

3.1 Watch: 关注此库，此库有任何的提交或者融合，ISSUE变动都会通过邮件通知到关注者。

3.2 Star：类似社交网站的“点赞”。

3.3 Fork：将项目复制到自己的库中。

# 五、NGBF 第三方开源引用

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **项目** | **版本** | **官网地址** | **用途** | **许可证书** |
| Spring | 2.5.6 | <http://www.springsource.org/> | IOC框架 | ASL 2.0 |
| Hibernate | 3.3.2 | <http://www.hibernate.org/> | 数据持久层(ORM)框架 | LGPL v2.1 |
| Mybatis | 3.2.2 | <http://ibatis.apache.org/> | 数据持久层(ORM)框架 | ASL 2.0 |
| C3p0 | 0.9.1.2 | <http://www.c3p0.se/> | JDBC连接池 | LGPLv2 |
| Primefaces | 3.5 | <http://www.primefaces.org/> | 支持JSF规范的UI框架 | ASL 2.0 |
| POI | 3.9 | <http://poi.apache.org/> | 微软XLS、XLSX、DOC、DOCX文档读写操作类库 | ASL 2.0 |
| DOM4J | 1.6 | <http://www.dom4j.org/> | XML文档读写操作类库 | BSD-style |
| Log4j | 1.2.14 | <http://logging.apache.org/log4j/2.x/> | 日志操作包 | Apache License Version 2.0 |
| Java Mail | 1.4 | <http://www.oracle.com/technetwork/java/javamail/index.html> | 邮件发送类库 | CDDL and GPLv2+CE |
| Quartz | 1.8.6 | <http://www.quartz-scheduler.org/> | 调度框架 | ASL 2.0 |
| Rome | 1.0 | <https://java.net/projects/rome/pages/Home> | RSS解析类库 | ASL 2.0 |
| expressionJ | 0.9.1 | <http://sourceforge.net/projects/expressionj/> | 简单算术表达式类库 | LGPLv2.0 |
| MINA | 2.0.6 | <http://mina.apache.org/> | NIO数据通讯框架 | Apache License 2.0 |
| Oscache | 2.3 | <http://www.opensymphony.com/oscache/> | 缓存框架 | Apache Software License |
| Xmemcached | 1.2.6.2 | <https://code.google.com/p/xmemcached/> | memcached客户端 | ASL 2.0 |
| Camel | 2.9.1 | <http://camel.apache.org/> | ESB框架 | ASL 2.0 |
| ActiveMQ | 5.3.0 | <http://activemq.apache.org/> | 消息服务中间件 | ASL 2.0 |
| Jackrabbit | 1.5.5 | <http://jackrabbit.apache.org/> | JCR内容管理框架 | ASL 2.0 |
| JasperReports | 5.0.1 | <http://community.jaspersoft.com/project/jasperreports-library> | 报表引擎 | LGPL |
| CAS | 3.3.3 | <http://www.jasig.org/cas/> | SSO单点登录 | [CAS License](http://www.jasig.org/cas/license) |
| Velocity | 1.7 | <http://velocity.apache.org/> | 模板引擎 | ASL 2.0 |
| ZooKeeper | 3.4.5 | <http://zookeeper.apache.org/> | 分布式应用协调服务框架 | ASL 2.0 |

# 六、总结

## 6.1 开源的必然性

通过前面对开源历史的介绍，不难发现。开源是一个行业走向成熟的必然产物。一个行业起步时，掌握行业核心技术的只有几个寡头公司，而这几个寡头公司靠技术专利就可以垄断该行业，而该行业的下游生产商则很大程度上要依附这些寡头公司，并支付给寡头公司高额的专利接触费。随着行业慢慢扩张做大，更多企业进入市场，他们现在不愿意都支付高额的专利接触费给寡头，便会建立开源联盟，来共享一些基础信息和技术。而正是由于这些基础信息的共享，使得更多的企业有条件进入这个市场，对现有的技术进行再改进。随着市场的变大，和迭代的加快，一个行业才会进入健康发展的轨道。

## 6.2 开源代码的类型和开源的收益

查看大量开源项目后，不难发现，开源代码大多提供的是基础或者框架层面的服务。它们不是只为某一个特定产品服务。这些代码有着非常强的复用性。

开源基础或框架代码，除了降低他人改进代码的成本以外，也给代码带来了更多的使用者。更多的使用者，同时也意味着更多的代码审核，代码扩展，代码稳定性的提升。而这些对于基础或者框架层面的代码，则有着非常重要的意义。

同时，优秀的开源项目，随着参与者的增加，慢慢会形成产业生态环境，甚至达到一定量级时，该项目的一些基础理念会成为产业标准，例如之前介绍的Spring。

## 6.3 开源项目的盈利模式

大部分的开源项目，通过开源，不停地进行快速迭代更新，同时积累用户，当形成一定规模后，一些用户会希望该项目更快速的发展，从而对该项目进行投资。比较著名的包括Linux社区，Android社区等，都由很多软件硬件巨头每年大量的对其进行投资，辅助其发展。

另外，随着用户的增加，生态环境的做大，开源组可以找到更多的盈利机会。例如，有偿的技术培训，定制级开发等。

## 6.4 开源社区的建设

总结Github的成功。主要有以下几个因素：

1. 简约。其帐号的创建，直接在主页上填入三个输入框，分别为昵称，邮箱和密码，即可完成。
2. 社交功能的引入。关注项目或者跟随某个人，都可以让用户更好的跟踪项目进展。
3. Fork & Pull 模式的引入。所有人都可以通过Fork把开源项目保存到自己的库中，而不需要得到原库管理员的认可。而当自己进行了修改后，可以通过Pull Request来进行和原库融合的请求。
4. JIRA ISSUE模式的引入。ISSUE的简单引入，不但提高了项目问题管理能力，也增加了项目社区的沟通效率。
5. WIKI的融入。一个好的开源项目，离不开好的文档管理，而在开源社区中有一个WIKI来管理文档，则使其更直接高效。

在国内，由于网络的特殊限制，在访问国外的开源社区时，有诸多不便。所以国内也诞生了很多开源社区网站，但是，似乎很少有像Github这么功能全面的社区网站。如果能借鉴Github的成功经验，做个国内版的，应该也是很有意义的事情。

# 参考资料

1. Open Source Wiki: <http://en.wikipedia.org/wiki/Open_source>
2. Open Source Definition: <http://opensource.org/definition>
3. Android Wiki: <http://en.wikipedia.org/wiki/Android_(operating_system)>
4. Spring Framework: <http://en.wikipedia.org/wiki/Spring_Framework>
5. NGBF: <http://ngbf/portal>
6. Github: <https://www.github.com>

1. 为美国国防等研究计划署开发的世界上第一个运营的封包交换网络，它是全球互联网的始祖。 [↑](#footnote-ref-0)
2. 由IBM使用者自发组织的，用来交流和分享开发语言，操作系统，数据库系统，及IBM企业应用的用户体验等技术信息的论坛。 [↑](#footnote-ref-1)
3. Open Source Definition. 开源定义。 [↑](#footnote-ref-2)