**Nodejs**

# 安装

## Windows

安装 ：<https://nodejs.org/> 可以安装 node 和 npm

## CentOS

安装 ：<https://nodejs.org/> 可以安装 node 和 npm

## mnp

<https://www.npmjs.com/>

## WebStrome

下载 ： WebStorm 11.0.3.rar 或 WebStorm-2017.1.tar.gz

然后用docker 见 ： <https://hub.docker.com/r/woailuoli993/jblse/>

|  |
| --- |
| Full Description jetbrains license server on docker 悄悄的干活，打枪的不要。 感谢[@ilanyus](http://idea.lanyus.com/)发布的认证服务及认证工具。认证方法 （在使用后面的的步骤搭建好服务后） http://qiniu.heyuhua.com/jblse_show.png 认证jetbrains系列ide时点选license, 并填写http://127.0.0.1:20701. 如果要提供服务给团队使用，记得把127.0.0.1 替换为 相应的外网ip。 临时认证搭建 docker run -it --rm -p 20701:20701 woailuoli993/jblse:latest version 0.2.0 增加了自定义认证用户名参数。 usage  1. docker pull woailuoli993/jblse:0.2.0  * 使用默认认证用户名 * docker run -d --name="jblse" -p 20701:20701 woailuoli993/jblse:0.2.0 * 使用自定义用户名 * docker run -d --name="jblse" -p 20701:20701 woailuoli993/jblse:0.2.0 -u coldplay   以上命令建立了一个认证用户名为coldplay的认证服务。  如果不想用这个端口， -p 后面的 第一个20701 也可以修改成别的。 |

## Web 框架

各种框架介绍：

<http://www.cnblogs.com/lhb25/p/10-best-node-js-mvc-frameworks.html>

<http://ourjs.com/detail/15%E4%B8%AA%E6%9C%80%E5%A5%BD%E7%94%A8%E7%9A%84node-js%E5%90%8E%E7%AB%AF%E6%A1%86%E6%9E%B6>

<http://www.techweb.com.cn/network/system/2015-12-11/2240069.shtml>

<http://www.csdn.net/article/2014-03-25/2818964-web-application-frameworks-for-node-js>

## Express

<https://www.npmjs.com/package/express-generator>

npm i -g express-generator

<https://github.com/expressjs/generator>

中文站点： <http://www.expressjs.com.cn/>

## AngularJS

英文网站： <https://angularjs.org/>

中文网站： <https://angular.cn/>

NgNice：　站点：<http://ngnice.com/>　NgNice 站点是由一批 Angular.js 爱好者发起的，致力于打造一个 Angular.js 的学习和经验分享平台　　<https://github.com/angular-cn/ng-nice>

## 框架选择

|  |
| --- |
| [**Node.js**](http://lib.csdn.net/base/nodejs)非常适用于Web开发，但是现在无论是一个网站，还是Web App都已经成为包括很多不同部分，如前端、**[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "_blank)**、业务模块、功能模块等等的大型项目，使用Node.js从零开始进行Web开发，也许大中型团队能够胜任，但对于个人和小型团队来说是不现实的。这时候框架就成为Web开发利器，对于个人开发来说几乎是必不可少。那么如何选择Node.js Web开发框架呢？  首先，我们必须要弄清楚的是，我们需要的是—— 程序 or 框架？ 程序是已经成型的应用，你需要的是为它搭建环境、添加配置，然后就可以运行起来；框架则是应用的骨架，你需要为它添加数据模型、业务逻辑，它才能成为应用，开始提供服务。  事实上，对于Web开发来说，程序和框架的区别正越来越模糊，比如几乎妇孺皆知的Wordpress，它是一个博客程序，但它丰富的插件以及高度的自定义能够支持很大程度上的二次开发，在这点上它比起一些**[PHP](http://lib.csdn.net/base/php" \o "PHP知识库" \t "_blank)**框架也并不逊色。我个人认为，如果重心在于提供服务而不是掌握技术，有WordPress这样的程序是没有必要使用框架的。  可惜的是，由于Nodejs还很年轻，目前还没有WordPress这样的程序，因此目前在Node.js开发里，如果想做出自己想要的作品，框架是必然的选择。如果是某些特定类型的应用，可以尝试一些开源的程序，比如要用Nodejs做博客，有Hexo、Ghost等。 Node.js Web框架有哪些？ Node.js里的Web框架分为API框架和Web应用框架。前者能够开发出RESTful的API，后者也能开发出RESTful API，但还包括模板、渲染等为前端所准备的功能。  API框架的使用场景是为跨平台应用提供统一的数据模型，而渲染由前端/客户端自行解决。目前比较知名的API框架有   * **restify**（[文档](http://mcavage.github.io/node-restify/)、[Github](https://github.com/mcavage/node-restify)、[NPM](https://www.npmjs.org/package/restify" \t "_blank)） * **ActionHero.js**（[官网](http://actionherojs.com/)、[Github](https://github.com/evantahler/actionhero)、[NPM](https://www.npmjs.org/package/actionhero" \t "_blank)） * **LoopBack**（[官网](http://strongloop.com/node-js/loopback/)、[Github](https://github.com/strongloop/loopback)、[NPM](https://www.npmjs.org/package/loopback" \t "_blank)） * **Fortune.js**（[官网](http://fortunejs.com/)、[Github](https://github.com/daliwali/fortune)、[NPM](https://www.npmjs.org/package/fortune" \t "_blank)） * **~~Frisby~~**（[官网](http://frisbyjs.com/)、[Github](https://github.com/vlucas/frisby)、[NPM](https://www.npmjs.org/package/frisby" \t "_blank)）经提醒修正，这是一个用于测试RESTful API的框架，并不是API框架。   Web应用框架顾名思义，就是为了打造Web应用所开发的框架。这里有两种风格的Web应用框架。  一个是Sinatra风格，另一个是Rails风格。Sinatra和Rails都是Ruby语言的Web框架，后者的影响力更大也更为知名。这里简单的解释一下两种风格是什么意思。  Sinatra风格是指高度可配置，注重开发的自由度。代表性的Nodejs Web框架有：   * **Express**（[官网](http://expressjs.com/)、[Github](https://github.com/strongloop/express)、[NPM](http://npmjs.org/express" \t "_blank)）TJ大神开发，Node.js官方推荐 * **hapi**（[官网](http://hapijs.com/)、[Github](http://github.com/hapijs/hapi" \t "_blank)、[NPM](https://www.npmjs.org/package/hapi" \t "_blank)） * **koa.js**（[官网](http://koajs.com/)、[Github](https://github.com/koajs/koa)、[NPM](https://www.npmjs.org/package/koa" \t "_blank)） * **flaliron**（[官网](http://flatironjs.org/)、[Github](https://github.com/flatiron/flatiron)、[NPM](https://www.npmjs.org/package/flatiron" \t "_blank)） * **total.js**（[官网](http://www.totaljs.com/)、[Github](https://github.com/totaljs/framework)、[NPM](https://npmjs.org/package/total.js" \t "_blank)） * **locomotive**（[官网](http://locomotivejs.org/)、[Github](https://github.com/jaredhanson/locomotive)、[NPM](https://www.npmjs.org/package/locomotive" \t "_blank)）   Rails风格则是指不重复自己和约定优于配置，以及严格遵循MVC结构开发。代表性的框架有：   * **Sails.js**（[官网](http://sailsjs.org/)、[Github](https://github.com/balderdashy/sails/)、[NPM](https://www.npmjs.org/package/sails" \t "_blank)） * **geddy**（[官网](http://geddyjs.org/)、[Github](https://github.com/geddy/geddy)、[NPM](https://www.npmjs.org/package/geddy" \t "_blank)） * **CompoundJS**（[官网](http://compoundjs.com/)、[Github](https://github.com/1602/compound)、[NPM](https://www.npmjs.org/package/compound" \t "_blank)） 原railswayjs   这两种风格无所谓谁优谁劣，全凭使用者的偏好。  而在这两种Web框架之外，还有更大型的框架，即全栈框架，其中的代表是MEAN。 MEAN？ MEAN指**[MongoDB](http://lib.csdn.net/base/mongodb" \o "MongoDB知识库" \t "_blank)**+Express+Angular.js+Node.js，这一组合包括运行环境、数据库、Web框架和前端引擎。被称为全栈框架（Full-stack framework）。这其中除了Node.js之外，每一个都是可替换的，目标是创建从前端到后端，全部使用**[JavaScript](http://lib.csdn.net/base/javascript" \o "JavaScript知识库" \t "_blank)**的Web应用。  由于这一框架的完善性，有人将其称为LAMP的接班人。LAMP即PHP的典型运行环境，**[Linux](http://lib.csdn.net/base/linux" \o "Linux知识库" \t "_blank)**+Apache+[**MySQL**](http://lib.csdn.net/base/mysql)+PHP，被大量的用于各种虚拟主机上。  MEAN看似庞大，但事实上要构建完整的现代化Web应用，特别是SPA（单页面应用），这几个组件都是难以缺少的，并且，其中每一项几乎都是目前情况下的最佳选择，因此用于学习和重头开始打造新的Web应用是非常合适的。但由于实际业务的独特性，很可能要替换其中的组件，比如用Mysql来替换MongoDB，因此，学习其中的原理和**[架构](http://lib.csdn.net/base/architecture" \o "大型网站架构知识库" \t "_blank)**，打造自己的类MEAN框架也是一种选择。  作为个人和小团队来说，全栈框架MEAN基本上足够了，但目前大多数全栈框架还包含一项特性，那就是实时，拥有实时功能的框架我们又称为实时框架。 实时框架好吗？ 实时框架（Real-time framework）指包含了webSocket的双向通信功能，能够在服务器和客户端做到实时通信的框架。  服务端和客户端自由通信的需求一直都在，但由于HTTP协议本身的局限性，因此催生了Comet等变通的方法，但即使这样也离实时相距甚远。而当Node.js兴起后，另一个**[HTML5](http://lib.csdn.net/base/html5" \o "HTML5知识库" \t "_blank)**技术webSocket也渐渐成熟，人们突然发现，实时通信一下子变得触手可及，于是webSocket技术在Node.js中得到大量的应用，其中最为知名的模块就是socket.io，而各种全栈框架也纷纷加入实时特性来应对更广阔的开发需求。  目前有代表性的实时框架有：   * **Meteor**（[官网](http://meteor.com/)、[Github](https://github.com/meteor/meteor)、[NPM](https://www.npmjs.org/package/meteor" \t "_blank)） * **MEAN.io**（[官网](http://mean.io/)、[Github](https://github.com/linnovate/mean)、[NPM](https://www.npmjs.org/package/meanio" \t "_blank)） * **Derby**（[官网](http://derbyjs.com/)、[Github](https://github.com/derbyjs/derby)、[NPM](https://www.npmjs.org/package/derby" \t "_blank)） * **SocketStream**（[官网](http://socketstream.org/)、[Github](http://github.com/socketstream/socketstream)、[NPM](https://npmjs.org/package/socketstream" \t "_blank)）   不过说实话，目前能看到的实时通信的应用场景其实不多，其中大多集中于聊天室、to-do、实时图表、在线游戏等领域。其他领域使用实时特性不但没必要，而且是对服务器资源的浪费。因此目前是否要采用实时框架，要看具体的项目而定。  以上基本就是Node.js Web框架的现状了，相信看到这里，对于选择何种框架读者已经心里有数了吧。最后再介绍一个容易搞混的概念，和解释一下我的选择。 YEOMAN？ 第一次见到这个词，我还以为它和MEAN有什么联系。事实上，它们是截然不同的两个东西。YEOMAN由YO（脚手架）、grunt（构建工具）、bower（包管理器），它代表的是一种工作流，与框架开发的思维方式完全不同。具体的介绍可见[这里](http://blog.fens.me/nodejs-yeoman-intro/" \o "Yeoman自动构建js项目" \t "_blank)。  YEOMAN能够和框架达到类似的目的，都是为构建一个Web应用做好准备，但是要不要采用YEOMAN，则是见仁见智。我个人的看法是，学习YEOMAN本身就需要不少时间，并且有一定的学习门槛。至少在目前，使用框架开发还是相对经济的，而如果以后YEOMAN这种模式推广开来，再来学习也不迟，更何况有一定的Node.js项目经验之后再来学习YEOMAN要轻松很多。  事实上，我还是很认可YEOMAN这种Generator+package Manager的模式的，这是因为Node.js本身崇尚**微模块**的概念，即无论是多么小的功能，都将它们模块化，甚至大的模块也要拆分成小的模块，然后通过搭积木的方式来构建应用。这样能够彻底的解耦，对于不容易调试的Javascript来说，也有助于定位和修复应用中的问题。Generator就是这种理念催生下的产物，通过选择不同的配置和选项，将积木搭起来。不过对于这种模式目前大家也还处于实验当中，不急于进行实际应用。 为什么我选择了Hackathon Starter？ 在我的个人项目中，使用的是Hackathon Starter，一个[Node.js Web应用脚手架](http://idlelife.org/archives/491" \t "_blank)。  我使用它的原因是，要求高度可配置，同时又讨厌写一些配置的代码，因此它对于我来说是很好的选择。一些全栈框架对我来说，封装过多，将原生的Node.js/Express API隐藏掉了，要使用还需要一定的学习成本。而Express这样的框架又太过简洁，在实际的项目中使用还需要大量的插件和配置，而这些在Hackathon Starter中都已经帮我们做好了，同时还有一些示例代码以供学习，对于新人来说非常友好，可以避免过多的挫折感。  上面一段可以看做是免费为[Hackathon Starter](https://github.com/sahat/hackathon-starter)做的广告吧，开源项目需要宣传和布道才能让更多人所关注。  最后，本文里的框架大多来源于[nodeframework](http://nodeframework.com/)网站，本文可以看做是该站的注释版，在扫清我自己的一些疑惑的同时，也希望对读者有所帮助。 |

# 全栈

## MEAN.IO

|  |
| --- |
| http://mean.io/wp-content/uploads/sites/164/2016/08/banner-top-ninja-1.png  网站：<http://mean.io/>  代码：<https://github.com/linnovate/mean>  从属于：<http://www.linnovate.net/>  学习： <http://learn.mean.io/> **<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<**  MEAN =（Mongo DB，Express，Angular 和 Node.js）    $ sudo npm install -g mean-cli   $ mean init yourNewApp |

## MEAN.JS

|  |
| --- |
| MEAN.JS  网站： <http://meanjs.org/>  代码： <https://github.com/meanjs/mean>  文档见：<http://meanjs.org/generator.html>  先安装 [yo scaffolding tool](https://github.com/yeoman/yo) 【**yo 即 Yeoman 见**<http://yeoman.io/>】  $ npm install -g yo  再安装 MEAN.JS generator  $ npm install -g generator-meanjs |

## Angular Full-Stack Generator

网站： <https://angular-fullstack.github.io/>

代码： <https://github.com/angular-fullstack/generator-angular-fullstack>

|  |
| --- |
| 先安装yo  $ npm install --global yo generator-angular-fullstack gulp-cli  建立项目  $ mkdir example && cd $\_  $ yo angular-fullstack  ...  启动服务  $ gulp serve  得到结果  Home page screenshot |

## 相关文章

[**Nodejs之MEAN栈开发（一）---- 路由与控制器**](http://www.cnblogs.com/stoneniqiu/p/5538109.html)

[**Nodejs之MEAN栈开发（二）----视图与模型**](http://www.cnblogs.com/stoneniqiu/p/5551019.html)

[**Nodejs之MEAN栈开发（三）---- 使用Mongoose创建模型及API**](http://www.cnblogs.com/stoneniqiu/p/5556669.html)

[**Nodejs之MEAN栈开发（四）---- form验证及图片上传**](http://www.cnblogs.com/stoneniqiu/p/5613823.html)

[**Nodejs之MEAN栈开发（五）---- Angular入门与页面改造**](http://www.cnblogs.com/stoneniqiu/p/5641721.html)

[**Nodejs之MEAN栈开发（六）---- 用Angular创建单页应用(上)**](http://www.cnblogs.com/stoneniqiu/p/5659158.html)

[**Nodejs之MEAN栈开发（七）---- 用Angular创建单页应用(下)**](http://www.cnblogs.com/stoneniqiu/p/5669419.html)

[**Nodejs之MEAN栈开发（八）---- 用户认证与会话管理详解**](http://www.cnblogs.com/stoneniqiu/p/5690731.html)

MEAN: AngularJS + NodeJS的REST API开发教程 <http://www.jdon.com/idea/nodejs/web-app-with-angularjs-and-rest-api-with-node.html>

# 开源实例

## NgNice：Angular.js 学习和经验分享平台

网站： <http://ngnice.com/>

源代码： <https://github.com/angular-cn/ng-nice>

## Nodeclub：使用 Node.js 和 MongoDB 开发的社区系统

网站： <https://cnodejs.org/>

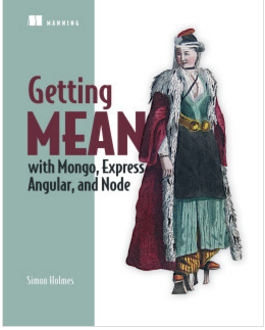
源代码：<https://github.com/cnodejs/nodeclub>

## Imprint - User Authentication with MEAN

网站： <https://boiling-harbor-43090.herokuapp.com/>

源代码：<https://github.com/lushen/imprint>

书：Getting MEAN with Mongo, Express, Angular, and Node <https://www.manning.com/books/getting-mean-with-mongo-express-angular-and-node>



## Mongodb、Express、Nodejs与Angularjs结合的MEAN架构指南系列之CRUD综合DEMO实例

网站：<http://www.upsnail.com/?p=670>

源代码：<https://github.com/lxf/MEAN>

## 一个JS全栈构建的博客应用，MEAN架构

网站： <http://114.215.164.12:3000/> 【无法访问】

源代码： <https://github.com/icyse/mean-blog>

## Blog Engine

网站：<http://zackyang.com/>

源代码：<https://github.com/TossShinHwa/CMS>