



JHipster 是什么？

JHipster 是一个 Yeoman 的代码生成器。用来创建基于 Spring Boot + AngularJS 的应用，提供完全热加载的 Java 和 JavaScript 代码。



就如上图所示那样，Yeoman + Spring Boot + AngularJS = JHipster

基于的技术堆栈

Spring Boot: 能建成独立 Spring 的应用程式

Spring Security: 标准业内的授权和认证

AngularJS: JavaScript 的 MVC 框架客户端

Bootstrap: 来自 Twitter，是目前最受欢迎的前端框架

REST APIs: 基于 Http 协议实现资源操作

Liquid Database: 数据库源代码版本控制

CSS3 + 动画

HTML5: 移动开发主导 (Mobil First)

Full internationalization support: 支持完善的国际化文字

Web Socket: 允许用户在浏览器中实现双向通信，实现数据的及时推送

嵌入式 tomcat, jetty 及 undertow

项目工具选择

自动配置依赖资源: Maven 或 Gradle

验证类型: Cookie type, JWT 和 OAuth2

数据库: SQL 及 NOSQL (MySQL, Postgres, H2)

高速缓存: EhCache or Hazelcast
自动化重复任务: 实时编辑: GULP 及 Grunt
各类工具: Yeoman, npm (nodejs), BrowserSync

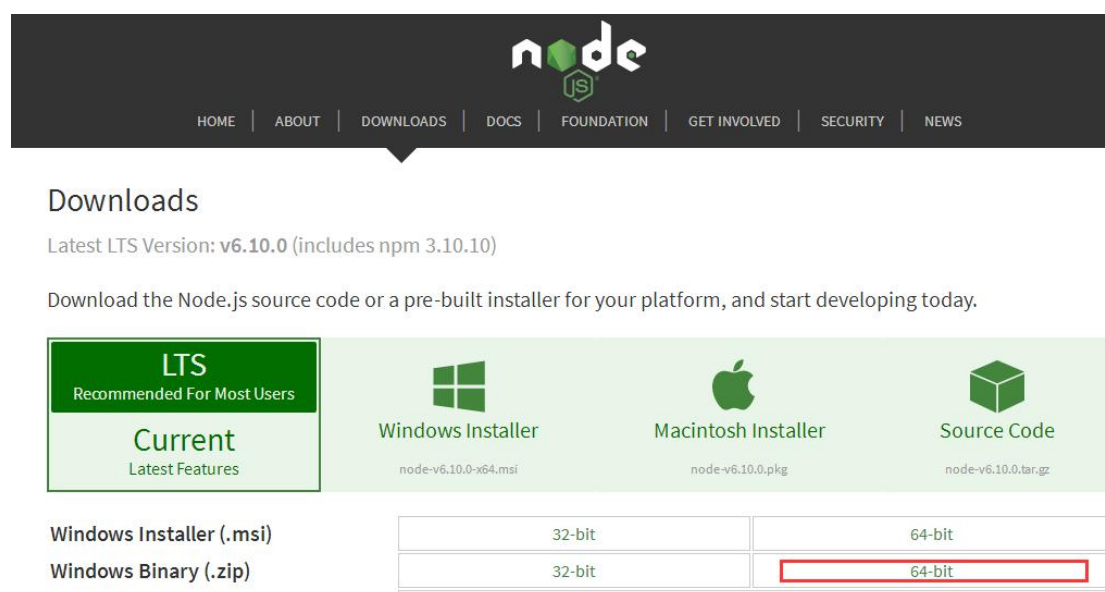
更多关于 JHipster 介绍及使用请参见: <https://jhipster.github.io/>

准备环境

Java 8 SDK
Maven 或 Gradle
nodeJs
PhantomJS
Mysql 或 Postgres (Mysql Workbench or pgAdmin3)
STS ide, Eclipse, IntelliJ IDEA, 其中一个

第一步: 安装 nodeJS

nodeJS 下载地址: <https://nodejs.org/en/download/>



node


HOME | ABOUT | DOWNLOADS | DOCS | FOUNDATION | GET INVOLVED | SECURITY | NEWS


Downloads


Latest LTS Version: **v6.10.0** (includes npm 3.10.10)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS
Recommended For Most Users
Current
Latest Features


Windows Installer
node-v6.10.0-x64.msi


Macintosh Installer
node-v6.10.0.pkg

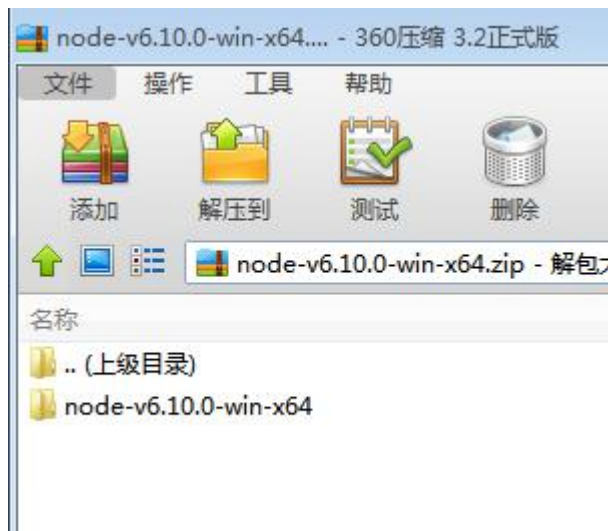

Source Code
node-v6.10.0.tar.gz

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit

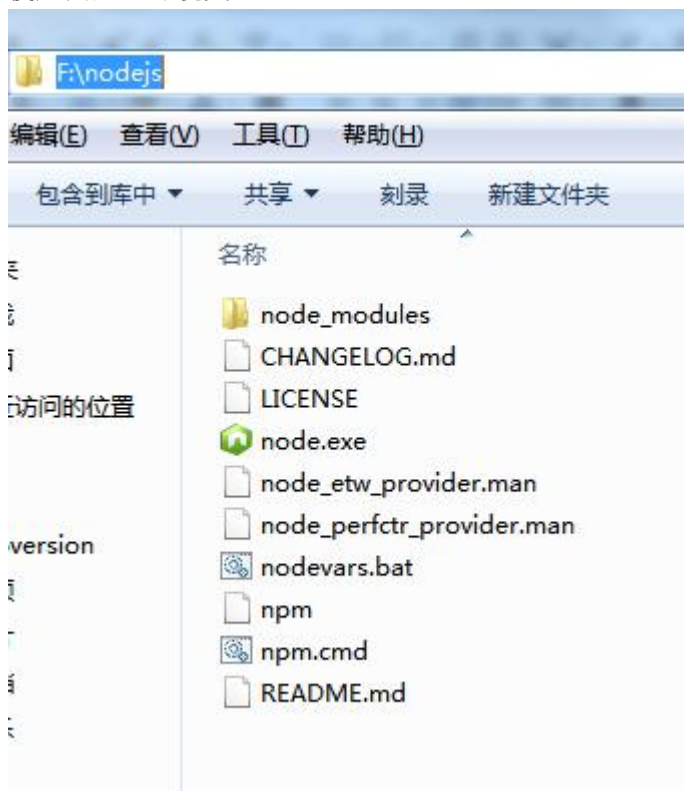
参照自己的操作系统下载对应的安装文件, 我下载的是 windows 64 位 zip 压缩包。



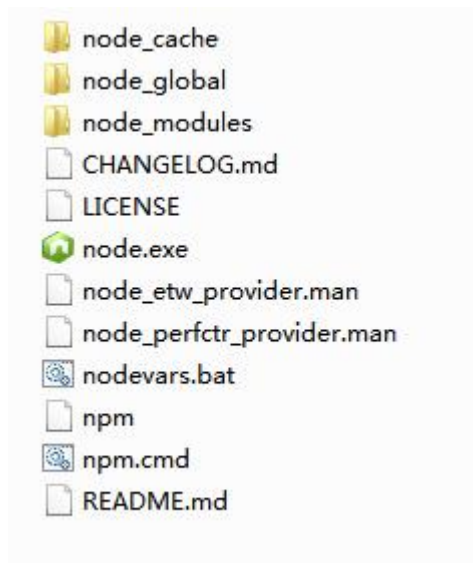
下载后的文件如上图。



将压缩包文件夹解压至磁盘任意位置，并将 node-v6.10.0-win-x64 文件夹更名为 nodejs，方便后面配置环境变量。



然后在目录下创建两个文件夹，分别是 node_global 和 node_cache，用于将 npm 安装的模块放置到全局统一的目录位置。(关于 nodejs 和 npm 请自行百度了解，这里不做过多描述)

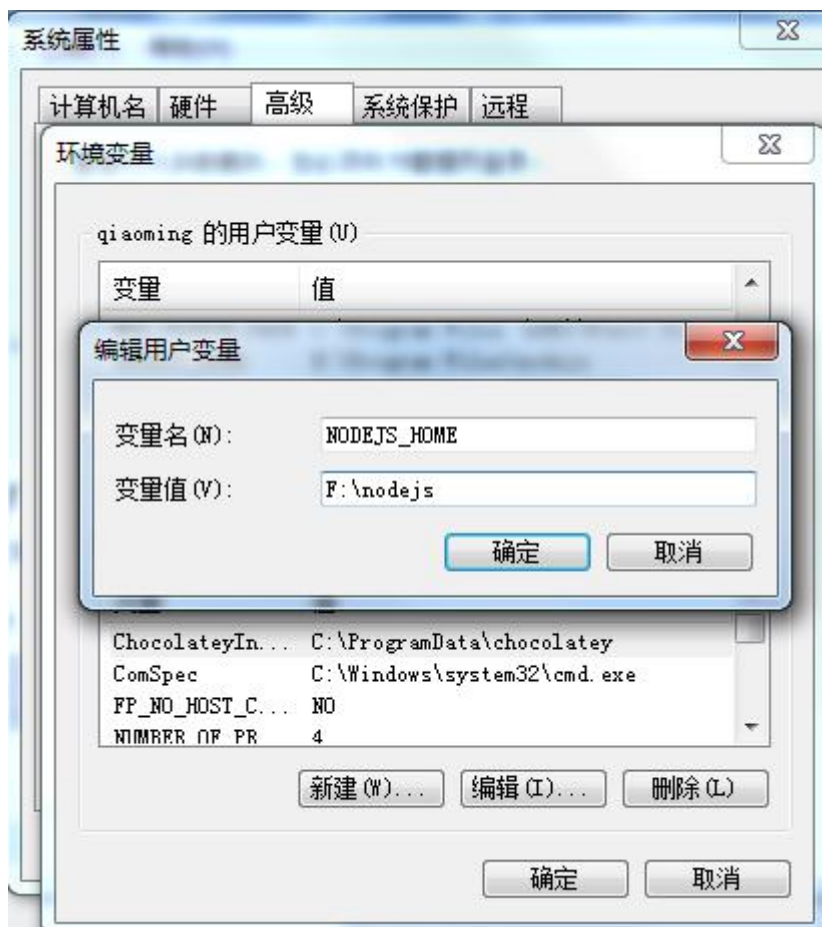


修整后的效果如上图。

第二步：配置系统环境变量

配置系统环境变量，用于命令行(cmd)操作。

我的电脑右键-》属性-》高级系统设置-》高级选项卡 环境变量



新建环境变量：

变量名: NODEJS_HOME

变量值: F:\nodejs

(变量值对应自己的 nodejs 目录)

然后编辑环境变量 Path:

追加变量值: %NODEJS_HOME%\;%NODEJS_HOME%\node_global;

这样就可以在命令行操作了。

打开命令行工具, 输入 `node -v` 和 `npm -v` 查看环境变量是否配置正确, 正确的话应该提示对应的版本号。



```
C:\Users\qiaoming>node -v
v6.10.0

C:\Users\qiaoming>npm -v
3.10.10

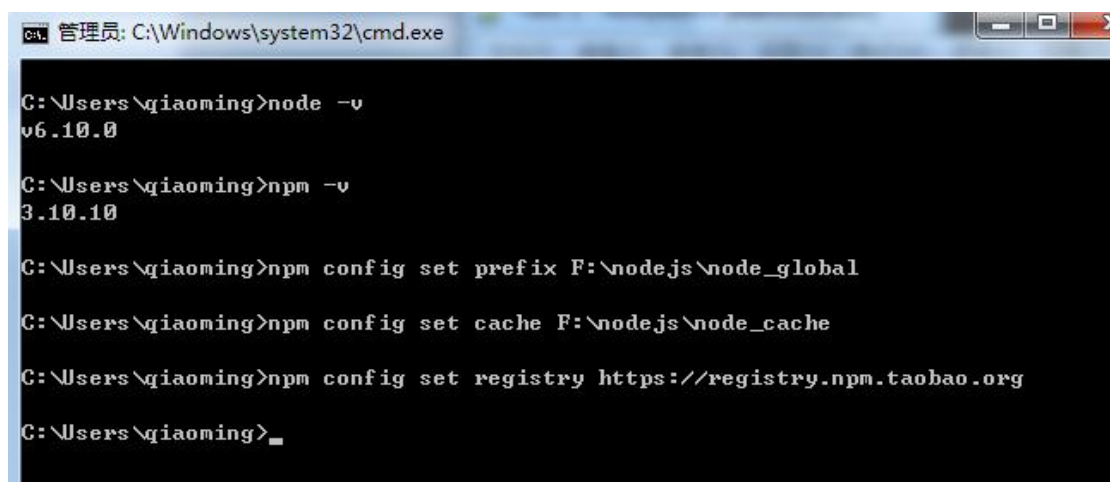
C:\Users\qiaoming>
```

然后依次输入如下命令, npm 模块包全局安装位置和缓存位置以及资源库镜像 (淘宝的镜像下载包比较快)。

`npm config set prefix F:\nodejs\node_global`

`npm config set cache F:\nodejs\node_cache`

`npm config set registry https://registry.npm.taobao.org`



```
C:\Users\qiaoming>node -v
v6.10.0

C:\Users\qiaoming>npm -v
3.10.10

C:\Users\qiaoming>npm config set prefix F:\nodejs\node_global

C:\Users\qiaoming>npm config set cache F:\nodejs\node_cache

C:\Users\qiaoming>npm config set registry https://registry.npm.taobao.org

C:\Users\qiaoming>
```

然后安装 Yo、Bower 和 Gulp。

```
npm install -g yo bower gulp-cli
```

继续安装 phantomjs。

```
npm install phantomjs -g
```

继续安装 jhipster，等待安装结束。

```
npm install -g generator-jhipster
```

至此，环境已经搭建完毕，下面使用 jhipster 生成一个可运行的示例项目。

在磁盘任意位置创建文件夹 jhWork（名称随便，只用来存放 jhipster 生成的项目代码）

然后在 jhWork 下创建 demo 作为示例项目的代码存放目录。

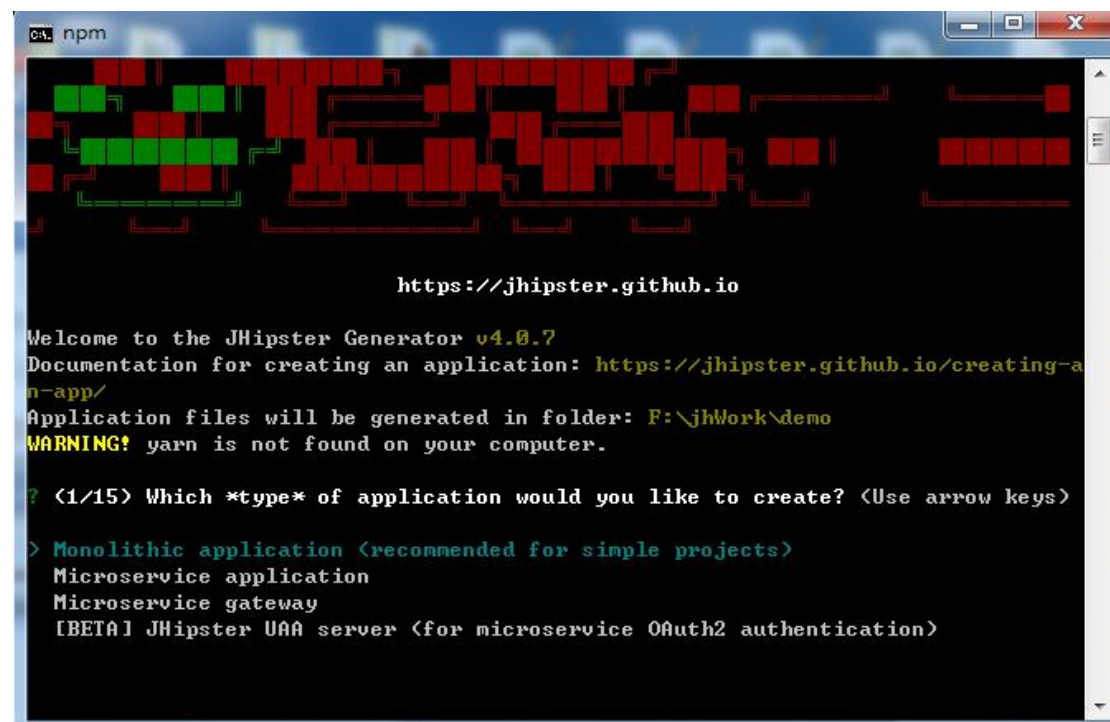
第三步：使用 jhipster 生成一个示例项目

回到命令行界面，进入 demo 目录。

```
C:\Users\qiaoming>f :  
f : \>cd jhWork/demo  
f : \jhWork\demo>
```

输入命令，然后会显示下图，这就是生成项目的选择提示页面：

yo jhipster



第一个是选择项目类型，选默认第一个，综合类型项目。

```
? <2/15> What is the base name of your application? demo
```

第二个是项目的名称，直接回车使用目录默认名称 demo。


```
? <3/15> Would you like to install other generators from the JHipster Marketplace? No
```

第三个是询问是否安装 jhipster 集市，直接回车，默认不安装。

```
? <4/15> What is your default Java package name? com.mycompany.myapp
```

第四个询问默认的包名称，直接回车使用默认包名。

```
? <5/15> Which *type* of authentication would you like to use? HTTP Session Authentication <stateful, default Spring Security mechanism>
```

第五个询问使用的验证方式，直接回车。

```
? <6/15> Which *type* of database would you like to use? SQL <H2, MySQL, MariaDB, PostgreSQL, Oracle, MSSQL>
```

第六个询问使用的数据库类型，使用 sql 类型数据库，直接回车。

```
? <7/15> Which *production* database would you like to use? MySQL
```

第七个询问使用的生产环境数据库，使用 mysql，直接回车。

```
? <8/15> Which *development* database would you like to use?  
H2 with disk-based persistence  
> H2 with in-memory persistence  
MySQL
```

第八个询问开发环境数据库，我们使用 H2 内存(in-memory)数据库方便快速搭建，回车下一步。

```
? <9/15> Do you want to use Hibernate 2nd level cache? No
```

第九个询问 Hibernate 使用的二级缓存，选择 NO，不使用，下一步。

```
? <10/15> Would you like to use Maven or Gradle for building the backend? Maven
```

第十个询问构建工具，选择 Maven，回车。

```
? <11/15> Which other technologies would you like to use? <Press <space> to select, <a> to toggle all, <i> to inverse selection>
```

第十一个选择其他技术，直接回车下一步。

```
? <12/15> Which *Framework* would you like to use for the client? AngularJS 1.x
```

第十二个，选择 AngularJS 版本，使用 1.x，直接回车。

```
? <13/15> Would you like to use the LibSass stylesheet preprocessor for your CSS? No
```

第十三个询问是否用 sass 处理 css 样式表，No，直接回车。

```
? <14/15> Would you like to enable internationalization support? Yes  
? Please choose the native language of the application? <Use arrow keys>  
Armenian  
Catalan  
> Chinese <Simplified>  
Chinese <Traditional>  
Czech  
Danish  
Dutch
```

第十四个询问是否国际化支持，YES，直接回车，选择 chinese<Simplified>中文简体。

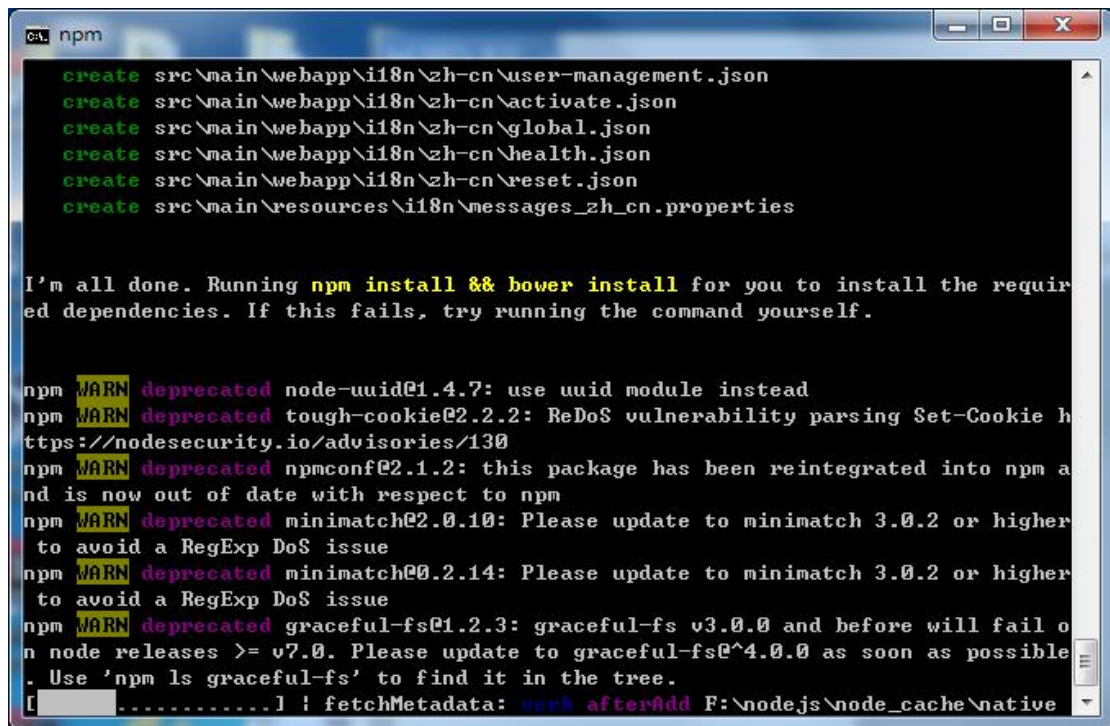
```
? Please choose additional languages to install <Press <space> to select, <a> to toggle all, <i> to inverse selection>
>< > Armenian
< > Catalan
< > Chinese <Traditional>
< > Czech
< > Danish
< > Dutch
< > English
```

然后会询问额外的国际化支持，直接回车。

```
<15/15> Besides JUnit and Karma, which testing frameworks would you like to use? <Press <space> to select, <a> to toggle all, <i> to inverse selection>
>< > Gatling
< > Cucumber
< > Protractor
```

直接回车。

设置完毕，接下来程序会创建基础文件和下载依赖的 jar 包。



```
ca. npm
create src\main\webapp\i18n\zh-cn\user-management.json
create src\main\webapp\i18n\zh-cn\activate.json
create src\main\webapp\i18n\zh-cn\global.json
create src\main\webapp\i18n\zh-cn\health.json
create src\main\webapp\i18n\zh-cn\reset.json
create src\main\resources\i18n\messages_zh-cn.properties

I'm all done. Running npm install && bower install for you to install the required dependencies. If this fails, try running the command yourself.

npm WARN deprecated node-uuid@1.4.7: use uuid module instead
npm WARN deprecated tough-cookie@2.2.2: ReDoS vulnerability parsing Set-Cookie header
https://nodesecurity.io/advisories/130
npm WARN deprecated npmconf@2.1.2: this package has been reintegrated into npm and is now out of date with respect to npm
npm WARN deprecated minimatch@2.0.10: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
npm WARN deprecated minimatch@0.2.14: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
npm WARN deprecated graceful-fs@1.2.3: graceful-fs v3.0.0 and before will fail on node releases >= v7.0. Please update to graceful-fs@^4.0.0 as soon as possible. Use 'npm ls graceful-fs' to find it in the tree.
[.....] ! fetchMetadata: work after add F:\nodejs\node_cache\native
```

安装完毕后，会显示如下信息：


```
C:\Windows\system32\cmd.exe

Client application generated successfully.

Inject your front end dependencies into your source code:
gulp inject

Generate the AngularJS constants:
gulp ngconstant:dev

Or do all of the above:
gulp install

[21:03:25] Using gulpfile F:\jhWork\demo\gulpfile.js
[21:03:25] Starting 'install'...
[21:03:25] Starting 'inject:test'...
[21:03:25] Starting 'inject:vendor'...
[21:03:25] Starting 'ngconstant:dev'...
[21:03:25] Finished 'install' after 157 ms
[21:03:25] Finished 'ngconstant:dev' after 125 ms
[21:03:25] gulp-inject 22 files into karma.conf.js.
[21:03:25] gulp-inject 23 files into index.html.
[21:03:25] Finished 'inject:test' after 200 ms
[21:03:25] Finished 'inject:vendor' after 163 ms
[21:03:25] Starting 'inject:dep'...
[21:03:25] Finished 'inject:dep' after 5.45 μs
[21:03:25] Starting 'copy:languages'...
[21:03:25] Finished 'copy:languages' after 4.2 ms
[21:03:25] Starting 'inject:app'...
[21:03:25] gulp-inject 101 files into index.html.
[21:03:25] Finished 'inject:app' after 183 ms
[21:03:25] Starting 'inject:troubleshoot'...
[21:03:25] gulp-inject 1 files into index.html.
[21:03:25] Finished 'inject:troubleshoot' after 4.85 ms

F:\jhWork\demo>
```

第四步：运行示例项目

在命令行输入：

```
mvn spring-boot:run
```

启动项目。

```
ca. 管理员: C:\Windows\system32\cmd.exe - mvn spring-boot:run

sConfiguration : Initializing Metrics JMX reporting
2017-03-09 21:10:38.315 INFO 9624 --- [ restartedMain] c.mycompany.myapp.conf
g.WebConfigurer : Web application configuration, using profiles: swagger
2017-03-09 21:10:38.317 DEBUG 9624 --- [ restartedMain] c.mycompany.myapp.conf
g.WebConfigurer : Initializing Metrics registries
2017-03-09 21:10:38.321 DEBUG 9624 --- [ restartedMain] c.mycompany.myapp.conf
g.WebConfigurer : Registering Metrics Filter
2017-03-09 21:10:38.322 DEBUG 9624 --- [ restartedMain] c.mycompany.myapp.conf
g.WebConfigurer : Registering Metrics Servlet
2017-03-09 21:10:38.325 DEBUG 9624 --- [ restartedMain] c.mycompany.myapp.conf
g.WebConfigurer : Initialize H2 console
2017-03-09 21:10:38.326 INFO 9624 --- [ restartedMain] c.mycompany.myapp.conf
g.WebConfigurer : Web application fully configured
2017-03-09 21:10:38.606 DEBUG 9624 --- [ restartedMain] c.m.myapp.config.Databa
seConfiguration : Configuring Liquibase
2017-03-09 21:10:38.673 WARN 9624 --- [demo-Executor-1] i.g.j.c.liquibase.Async
SpringLiquibase : Starting Liquibase asynchronously, your database might not b
e ready at startup!
2017-03-09 21:10:42.548 DEBUG 9624 --- [demo-Executor-1] i.g.j.c.liquibase.Async
SpringLiquibase : Started Liquibase in 3873 ms
2017-03-09 21:10:48.731 DEBUG 9624 --- [ restartedMain] i.g.j.c.apidoc.SwaggerC
onfiguration : Starting Swagger
2017-03-09 21:10:48.737 DEBUG 9624 --- [ restartedMain] i.g.j.c.apidoc.SwaggerC
onfiguration : Started Swagger in 6 ms
2017-03-09 21:10:50.033 INFO 9624 --- [ restartedMain] com.mycompany.myapp.Dem
oApp : Started DemoApp in 20.902 seconds (JVM running for 21.348)
2017-03-09 21:10:50.035 INFO 9624 --- [ restartedMain] com.mycompany.myapp.Dem
oApp :

-----
Application 'demo' is running! Access URLs:
Local:      http://localhost:8080
External:   http://192.5.0.224:8080
Profile(s): [swagger, dev]
-----
```

在浏览器访问地址: <http://localhost:8080/>





点击登录，默认账号和密码是：（首页信息提示已经告诉我们的）
管理员（账号="admin"和密码="admin"）
普通用户（账号="user"和密码="user"）

登录后可以点点菜单看看都有什么。



目前为止，我们已经成功运行了一个 jhipster 应用。
下面我们在程序中添加一个实体，用来操作增删改查以及分页功能。

第五步：添加一个实体模型

在命令行输入：
`yo jhipster:entity`

```
? Do you want to add a field to your entity? Yes
? What is the name of your field? parkName
? What is the type of your field? String
? Do you want to add validation rules to your field? No

===== Park =====
Fields
parkName <String>

Generating field #2

? Do you want to add a field to your entity? <Y/n>
```

第一个询问是否添加一个属性到你的实体，选择 YES，回车
 然后是属性名称。
 然后是属性类型。
 然后是是否验证。
 然后会回到一开始继续询问是否添加一个属性。
 我们只添加一个实体一个属性，选择 NO 回车。

```
? Do you want to add a relationship to another entity? No
```

询问是否添加另一个实体关系，No 回车。

```
? Do you want to use a Data Transfer Object (DTO)? <Use arrow keys>
> No, use the entity directly
[BETA] Yes, generate a DTO with MapStruct
```

询问是否使用 DTO 模型，直接回车。

```
? Do you want to use separate service class for your business logic? <Use arrow keys>
> No, the REST controller should use the repository directly
Yes, generate a separate service class
Yes, generate a separate service interface and implementation
```

不知道什么意思，直接回车。

```
? Do you want pagination on your entity? <Use arrow keys>
> No
Yes, with a simple pager
Yes, with pagination links
Yes, with infinite scroll
```

这个是询问是否实现分页效果。
 我选的第三个，是比较丰富的分页效果。

```
Everything is configured, generating the entity...

create .\hipster\Park.json
create src\main\resources\config\liquibase\changelog\20170310022025_added_entity_Park.xml
create src\main\java\com\mycompany\myapp\domain\Park.java
create src\main\java\com\mycompany\myapp\repository\ParkRepository.java
create src\main\java\com\mycompany\myapp\web\rest\ParkResource.java
create src\test\java\com\mycompany\myapp\web\rest\ParkResourceIntTest.java
conflict src\main\resources\config\liquibase\master.xml
? Overwrite src\main\resources\config\liquibase\master.xml? <Ynaxdh> _
```


回车后如上图，会生成相关文件，然后询问是否重写 master，应该就是实体对应的主页面资源吧。直接回车。

```
create .jhipster\Park.json
create src\main\resources\config\liquibase\changelog\20170310022025_added_entity_Park.xml
create src\main\java\com\mycompany\myapp\domain\Park.java
create src\main\java\com\mycompany\myapp\repository\ParkRepository.java
create src\main\java\com\mycompany\myapp\web\rest\ParkResource.java
create src\test\java\com\mycompany\myapp\web\rest\ParkResourceIntTest.java
conflict src\main\resources\config\liquibase\master.xml
? Overwrite src\main\resources\config\liquibase\master.xml? overwrite
force src\main\resources\config\liquibase\master.xml
create src\main\webapp\app\entities\park\parks.html
create src\main\webapp\app\entities\park\park-detail.html
create src\main\webapp\app\entities\park\park-dialog.html
create src\main\webapp\app\entities\park\park-delete-dialog.html
create src\main\webapp\app\entities\park\park.state.js
create src\main\webapp\app\entities\park\park.controller.js
create src\main\webapp\app\entities\park\park-dialog.controller.js
create src\main\webapp\app\entities\park\park-delete-dialog.controller.js
create src\main\webapp\app\entities\park\park-detail.controller.js
create src\main\webapp\app\entities\park\park.service.js
create src\test\javascript\spec\app\entities\park\park-detail.controller.spec.js
conflict src\main\webapp\app\layouts\navbar\navbar.html
? Overwrite src\main\webapp\app\layouts\navbar\navbar.html? <Ynaxdh>
```

询问是否重写导航条，选择是，直接回车。

```
conflict src\main\webapp\app\layouts\navbar\navbar.html
? Overwrite src\main\webapp\app\layouts\navbar\navbar.html? overwrite
force src\main\webapp\app\layouts\navbar\navbar.html
create src\main\webapp\i18n\zh-cn\park.json
conflict src\main\webapp\i18n\zh-cn\global.json
? Overwrite src\main\webapp\i18n\zh-cn\global.json? <Ynaxdh>
```

询问是否重写 global，直接回车。

```
conflict src\main\webapp\app\layouts\navbar\navbar.html
? Overwrite src\main\webapp\app\layouts\navbar\navbar.html? overwrite
force src\main\webapp\app\layouts\navbar\navbar.html
create src\main\webapp\i18n\zh-cn\park.json
conflict src\main\webapp\i18n\zh-cn\global.json
? Overwrite src\main\webapp\i18n\zh-cn\global.json? overwrite
force src\main\webapp\i18n\zh-cn\global.json

Running 'gulp inject' to add JavaScript to index.html

[10:24:28] Using gulpfile F:\jhWork\demo\gulpfile.js
[10:24:28] Starting 'inject:app'...
[10:24:28] gulp-inject 107 files into index.html.
[10:24:28] Finished 'inject:app' after 297 ms

F:\jhWork\demo>
```

现在我们重启之前的服务（ctrl+c 停掉服务）


```
:1.5.1.RELEASE:run <default-cli> on project demo: Could not exec java: Application finished with exit code: 1 -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoExecutionException
终止批处理操作吗(Y/N)?
```

输入 `y` 结束服务进程。

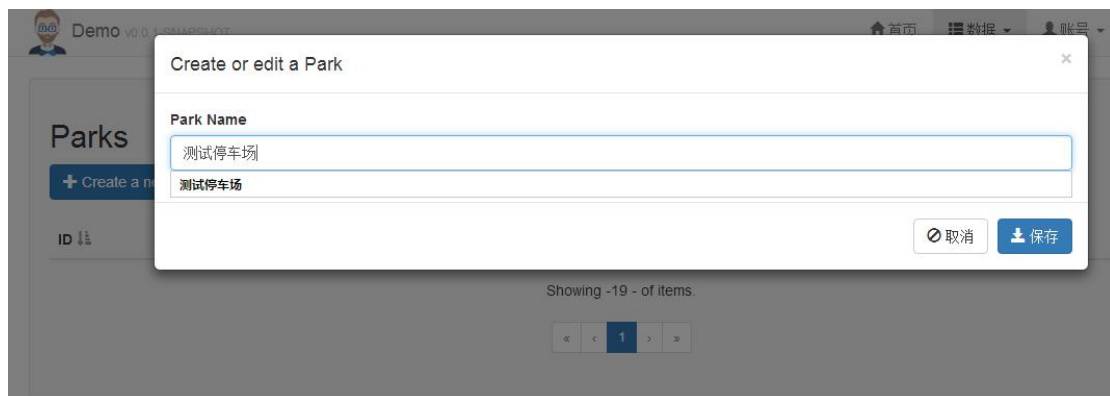
`mvn spring-boot:run`

重新启动项目。

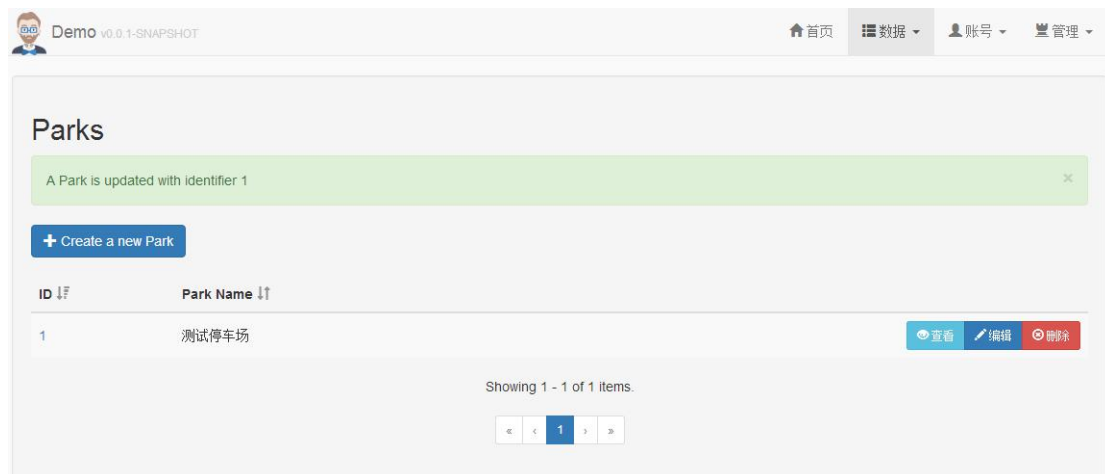
然后刷新 `http://localhost:8080` 并登陆。



可以看到已经有个 `park` 菜单，点击进去。



点击 `Create a new Park`



基本入门教程到此结束，其他参考的资源：

<http://yzijun.iteye.com/category/357163>

<http://www.zuidaima.com/share/2898005074136064.htm>

<http://www.cnblogs.com/wuya/p/jhipster-microservice-spring-cloud-demo.html>

JHipster 中文社区 QQ 群：58612944