JHipster 框架学习笔记

魏全福 2017.1.10

官网：https://jhipster.github.io/

1. 简介
2. JHipster的本质是一个基于Yeoman的代码生成器，其核心技术栈使用的是SpringBoot + AngularJS
3. JH就是很多框架、构建工具和其它工具的组合，如果先一个一个了解一下就好办了。yo是一个项目生成器，生成所有目录结构；gulp/grunt是前端框架（AngularJS、Bootstrap）构建工具；gradle/springboot是spring框架及组件构建工具；然后再加上loadblance、服务发现和注册以及各种插件。
4. 学习流程
5. 掌握以SpringBoot为核心的Spring生态系统(Spring Framework,Spring Data XXX,Spring Security....),熟读Spring官方各个子项的Refrencebook以及精品Spring书籍例如Springboot/Spring in action等;掌握AngularJS前端技术,熟读NG官方DOCS,练习官方的示例;熟悉Yeoman(Yo Gulp[Grunt在JHipster Release 2.27.0之前的生成器中作为生成可选项之一,Release 2.27.0之后,开发组就遗弃了Grunt的构建选项] Bower)前端构建工具
6. 去JHipster的Repository->[[JHipster · GitHub](//link.zhihu.com/?target=https%3A//github.com/jhipster)],下载jhipster/jhipster-sample-app-xxx(xxx代表所集成的Technology Stacks不一样,但核心都是SpringBoot+AngularJS),参照代码/模块 精读Gernerator Resources(这个阶段先熟悉生成的jhipster-sample-app-xxx)和官方的Refrencebook:[[JHipster - Generate your Spring Boot + AngularJS apps!](//link.zhihu.com/?target=http%3A//jhipster.github.io/)]
7. 一边学习,一边重构部分之前项目做小新项目(我想说真的真的非常快,非常好用),与此同时去Youtube等免费视频网站学习,常上Twitter看看了解开发组的最新资讯 ,遇到问题的时候在Gitter上提问,并帮助他人解决自己曾遇到的一些问题(Not Stack OverFlow,Gitter更加快捷方便,常驻开发组人员,能更效率的帮你解决问题),共同提高,这里提示下,做软件技术一定要有分享精神,协作精神
8. 进一步深入,可以学习Yeoman Generator,熟读官方的jhipster/generator-jhipster源码(这是用来生成代码的生成器源码,并非前面提到的jhipster/jhipster-sample-app-xxx,这是生成的代码),掌握How to Creating a module[[Creating a module](//link.zhihu.com/?target=http%3A//jhipster.github.io/modules/creating-a-module/)],以构建自己在项目中所需要的Modules[[Marketplace](//link.zhihu.com/?target=http%3A//jhipster.github.io/modules/marketplace/%23/list)],或者(再更深一步来说)根据自己的业务需求定制属于自己的JGernerator
9. 最后参加开发组的Webiner,理解领悟顶级开发人员的思想、观点,为什么要加入XXX为什么要这么架构;参与Jhipster项目的的讨论GoogleForum\_Jhipster[[https://groups.google.com/forum/?hl=en#!forum/jhipster-dev](//link.zhihu.com/?target=https%3A//groups.google.com/forum/%3Fhl%3Den%23%21forum/jhipster-dev)];参与Jhipster项目的BUG修复,问题谈论和方向规划GITHUB\_JHIPSTER\_ISSUSE[[Issues · jhipster/generator-jhipster · GitHub](//link.zhihu.com/?target=https%3A//github.com/jhipster/generator-jhipster/issues%3Fq%3Dis%253Aopen)];参与Jhispter项目的贡献Contributing[[generator-jhipster/CONTRIBUTING.md at master · jhipster/generator-jhipster · GitHub](//link.zhihu.com/?target=https%3A//github.com/jhipster/generator-jhipster/blob/master/CONTRIBUTING.md)]
10. JHipter安装
11. 安装Java 8，并配置好环境变量。
12. (可选)安装Java创建工具。
    1. 无论你选择使用Maven还是Gradle构建项目，都可以不安装，因为JHipster会自动帮你安装Maven或Gradle；
    2. 如果你不想用自动封装的工具，可以到官方网站下载安装；
13. 从官方网站下载安装Git，建议使用SourceTree工具。
14. 从官网安装Node.js，同时会自动安装npm，后续安装步骤会用到npm。
15. (建议)升级npm：npm install -g npm 。
16. 安装Yeoman：npm install -g yo 。
17. 安装Bower：npm install -g bower 。
18. 安装Gulp：npm install -g gulp-cli (如果你已经安装过Gulp，请运行npm rm -g gulp 命令，以防旧版本与 gulp-cli 冲突)。
19. 安装JHipster：npm install -g generator-jhipster 。
20. (可选)安装Yarn：npm install -g yarn如此就可以用yarn 标签yo jhipster –yarn 。
21. 创建Jhipster应用
22. 打开DOS窗口，进入要创建的项目所对应的空文件夹(testjhipster)。输入命令： yo jhipster 。
23. 接着出现交互问答，以便按照开发者要求去生成代码，实现应用的功能：
    1. What is the base name of your application?(testjhipster)  
       输入应用的基本名称，项目名，输入testjh；
    2. What is your default Java package name?(com.mycompany.myapp)   
       输入默认的Java包，输入com.hy.testjh；
    3. Which \*type\* of authentication would you like to use?(Use arrow keys)  
       选择那种身份验证类型，选默认的；
    4. Which \*type\* of database would you like to use?(Use arrow keys)   
       用什么类型的数据库？有三个选项：SQL、MongoDB、Cassandra。MongoDB是分布式文档存储数据库，Cassandra是开源分布式NoSQL数据库类型。我们选择常用的关系数据库类型SQL；
    5. Which \*production\* database would you like to use?(use arrow keys)   
       使用什么具体的数据库产品？我们选择默认的MySQL；
    6. which \*development\* database would you like to use?   
       开发时使用什么数据库？这里有三个选项，’H2 with disk-based persistence’、’H2 with in-memory persistence’和’MySQL’，第一个是将数据保存在运行内存中，重启服务器时，数据就会丢失。第二个是将数据保存在磁盘中，目前正在测试阶段，而且不能在window下正常使用。我们选择默认的MySQL。这个选择对应项目生成后的src/main/resources/config/application-dev.yml文件，在用MySQL数据库时可以在这个文件里修改相应的datasource参数；
    7. Do you want to use Hibernate 2nd level cache?   
       是否需要使用Hibernate二级缓存？根据自己需要，项目只是用来测试，因此我们选择NO；
    8. Do you want to use a search engine in your application?   
       是否需要使用一个搜索引擎。这个搜索引擎可以搜到关于jhipster的相关指南。我们选择默认的No；
    9. Do you want to use clustered HTTP sessions?   
       是否使用集群HTTP回话，不懂，没关系，选择默认的NO；
    10. Do you want to use WebSocket?  
        是否使用websocket？选择默认的No；
    11. Would you like to use Maven or Gradle for building the backend?  
        使用Maven或者Gradle构建后台。我们选择默认的Maven；
    12. Would you like to use　Grunt or Gulp.js for building the frontend?   
        使用默认的Gulp；
    13. Would you like to use the LibSass stylesheet preprocessor for your css?  
        是否使用LibSass作为css样式表处理器。我们输入NO；
    14. Would you like to enable translation support with Angular Translate?   
        是否使用Anjular提供的翻译支持？我们选择No；
    15. Which testing frameworks would you like to use?   
        测试框架的选择。我们选择默认；
24. 问题回答完毕，然后等待。直到结束，基本的jhipster项目testjhipster创建完毕，可以导入IDE。
25. 默认在创建完成后，系统自动运行 npm install && bower install 来加载需要的依赖，如果出现问题则需要手动执行这两条命令。
26. 提示：  
    在创建项目时，可以在命令后增加可选命令，如用 yo jhipster -–skip-install 来跳过自动安装依赖，默认false。  
    此外附加命令还有：  
    --help 打印生成器的选项和用法；  
    --skip-cache 不记录提示答案，默认false；  
    --skip-client 跳过client-side应用生成，因此你只有生成的springboot后台代码，默认false。与yo jhipster:server 命令结果相同；  
    --skip-server 跳过server-side应用生成，因此只会生成前端代码，默认false。与 yo jhipster:client 命令结果相同；  
    --skip-user-management 跳过用户管理生成，包括前端和后端，默认false；  
    --i18n 当跳过client生成后设置i18n是否可用，其它时刻没用，默认true；  
    --with-entities 如果存在实例就重新生成(用.jhipster中的配置信息) ，默认false；  
    --check-install 检查安装是否正确，默认true；
27. JHipster核心功能
28. 微服务
29. 创建应用  
    创建方法参考 四、创建JHipster应用
30. 创建实体
    1. 创建好应用后，你可能想创建实体，如Author或Book，对于每个实体，你需要以下条件(Jhipster创建项目时已经自动生成了以下内容)：
       1. 一个数据库表；
       2. 一个Liquibase change set；
       3. 一个Spring Date JPA 仓库；
       4. 一个能进行CRUD操作的Spring REST 控制器；
       5. AngularJS touter，控制器，服务；
       6. HTML显示；
       7. 综合测试，用来验证所有东西都按预期工作；
       8. 性能测试，用来查看所有东西是否平稳工作；

如果你已经有一些实体，想关联起来，那你就需要：

* + 1. 数据库外键；
    2. 管理这些关系的特殊javascript和HTML代码；
  1. **创建实体命令**：JHipster提供的实体生成器会创建需要的所有文件，并为每个实体提供CRUD前端。

yo jhipster:entity <entityName> --[options]

如需查看帮助信息，请执行：

yo jhipster:entity --help

支持的选项有：

--table-name <table\_name> 默认情况下，JHipster会自动生成一个与实体名相同的表，通过此选项可以重新设定表名；

--angular-suffix <suffix> 想让AngularJS使用自定义的前缀；

--regenerate 重新生成已存在的实体，并不会询问任何问题；

--skip-server 跳过服务端代码，只生成客户端代码；

--skip-client 跳过客户端代码，只生成服务端代码；

* 1. 如果你需要**批量创建实体**，最好是使用图形化工具：

JHipster UML 让你可以使用UML编辑器；

JDL Studio 是JHipster的在线生成实体和关联的工具，采用的是域特定语言；

* 1. **实例**：创建一个Author实体和Book实体：
* 执行 yo jhipster:entity author 命令，添加两个字段name (String类型)，birthDate (LocalDate属性)；
* 执行 yo jhipster:entity book 命令，添加四个字段title (String属性)，description (String属性)， publicationDate (LocalDate属性)，price (BigDecimal属性)，另外增加一个外键关联，名为book\_author，关联到author实体的name字段，为many-to-one类型；
* 上述步骤完成后，需要检查生成的代码：  
  运行生成的测试组建 mvn test 来测试Author和Book实体；  
  开启应用，如执行 mvn 命令，在页面中查看对应的实体，如果你添加了数据，就去数据库里查看对应的表。
  1. **改善生成的代码**：
* 自动生成的文件中只包含简单的增删改查操作，你可以根据自己项目的需要修改代码或数据库结构；此外，如果你需要处理复杂的业务逻辑，那你应该需要用JHipster提供的service生成器来添加一个Spring 的@Service类。
  1. **修改实体**：
* 用生成实体的命令来生成一个已存在的实体，就可以修改这个实体，如： yo jhipster:entity author ;
* 修改完成后，需要执行下列命令(如果修改文件时不需要需要询问，可移除 --force )：  
  mac & linux ： for f in `ls .jhipster`; do yo jhipster:entity ${f%.\*} --force ; done   
  windows ：for %f in (.jhipster/\*) do yo jhipster:entity %~nf --force

1. 创建DTO
2. 创建服务server
   1. 介绍
      1. 跟实体生成器相比，这是一个很简单的次生成器；
      2. 只需运行命令： yo jhipster:service bar 就可以生成一个名为bar的服务类；
      3. 运行后询问是否需要为该服务生成接口？我们选Y，生成接口；
   2. 一些疑问
      1. 为什么服务类不靠实体类生成？  
         答：Jhipster不建议生成没用的服务类，如果你只需要对实体进行增删改查操作，那你并不需要单独创建服务类；并且服务类应该涉及到多个实体库，因此不依赖实体创建服务类。
      2. 我们应该为服务类创建接口吗？  
         答：不。
      3. 为什么我们应该用事务去获取懒JPA关系？  
         答：
3. 管理关系
4. 国际化
5. 应用升级
6. 报错
7. 运行项目不报错，访问时网页中显示错误信息  
   解：cmd，cd到项目下，运行bower install，再运行gulp install ；
8. 运行gulp install命令后报错  
   Bower components directory does not exist at E:\testJHipster\src\main\webapp\bower\_components  
   解：手动创建bower\_components文件夹
9. 访问页面没内容，控制台报错  
   Uncaught ReferenceError: angular is not defined  
   bower\_components文件夹中没东西  
   原因：参照下一条；
10. 执行bower install命令时报错：  
    bower swagger-ui#2.1.5 ECMDERR Failed to execute "git ls-remote --tags --heads https://github.com/wordnik/swagger-ui.git", exit code of #128 fatal: unable to access 'https://github .com/wordnik/swagger-ui.git/': SSL read: error:00000000:lib(0):func(0):reason(0), errno 10054  
    原因：Github SSH配置问题；  
    解：<https://help.github.com/articles/generating-ssh-keys>
11. 在添加实体entity的关联关系后，运行项目报错：  
    Liquibase could not start correctly, your database is NOT ready: Validation Failed:  
    原因：liquibase的配置文件被改了，添加了一列，和原来的md5值不一致；  
    解：把添加的column移出来，重新建一个set；或修改数据库，手动添加列，把MD5值修改成新的；或使用no-liquibase配置文件，自己去管理数据库；
12. 参考引用
    * + 1. 官方文档：https://jhipster.github.io/
        2. 如何自学JHipster：https://www.zhihu.com/question/41219717/answer/102583263