# Linux java环境安装

<https://www.cnblogs.com/fswhq/p/10713429.html>

# Springboot项目部署

1. Linux安装好java环境jdk；
2. 关闭防火墙（或开放8080端口）；

service firewalld stop

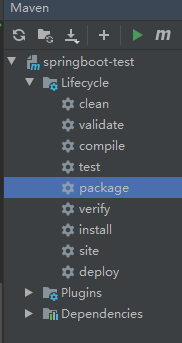
开放8080端口:

vim /etc/sysconfig/iptables

-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 8080 -j ACCEPT

service iptables restart

3）pom.xml文件，packaging写成jar，点击maven打jar包



4）将jar包使用ftp传输到服务器。

5）执行java -jar 包名启动项目

# springboot多模块构建

参考：<https://blog.csdn.net/zcf980/article/details/83040029>

## 搭建思路：

先创建一个 Spring Initializr工程 demo 作为 父工程， 然后在父工程再建四个子 Module (demo-base, demo-demo-dao, demo-service)，其实他们就是四个普通的Spring Initializr工程。

## 开始搭建：

1.创建一个父工程maven项目spring-demo，打包方式pom，删除不必要的文件(下面项目相同),只保留(.idea文件夹 , 和项目 pom 文件, 以及一个 \*.iml 文件 )，

1. 创建maven项目demo-base子模块，可以引进必要的依赖（mybatis,mysql驱动）

在pom文件中声明父工程，删除<relativePath/>：

<parent>  
 <groupId>com.chq</groupId>  
 <artifactId>springboot-demo</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
</parent>

在父工程spring-demo的pom文件中添加声明：

<modules>  
 <module>demo-base</module>  
</modules>

1. 创建demo-dao, demo-service，创建方法一样，同样创建一个maven项目，然后同demo-base一样在pom文件中声明父工程，在父工程spring-demo的pom文件中添加声明。
2. 创建demo-web 模块，引入web依赖，其他一样声明.....。
3. 保留demo-web的启动类和配置文件，其他项目的启动类和配置文件都删除。

除了demo-web，其他模块pom文件中的<build>删除掉，不需要打jar包。

1. 在demo-dao中添加demo-base 的依赖信息，在demo-service中添加 demo-dao 的依赖信息， 在 demo-web 中添加 demo-service 的依赖信息, 依赖信息添加到 各 pom 文件下的 <dependencies> </dependencies>节点下。

例如demo-dao：

<dependency>  
 <groupId>com.chq</groupId>  
 <artifactId>demo-base</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
</dependency>

1. 编写测试代码

**demo-dao:**

public class Department implements Serializable {  
 Integer id;  
 String departmentName;  
  
 public Integer getId() {  
 return id;  
 }  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
  
 public String getDepartmentName() {  
 return departmentName;  
 }  
  
 public void setDepartmentName(String departmentName) {  
 this.departmentName = departmentName;  
 }  
}

@Repository  
@Mapper  
public interface DepartmentMapper {  
 List<Department> selectAll();  
}

**demo-service:**

public interface DepartmentService {  
 List<Department> getAll();  
}

@Service  
public class DepartmentServiceImpl implements DepartmentService{  
  
 @Autowired  
 private DepartmentMapper departmentMapper;  
  
 @Override  
 public List<Department> getAll() {  
 return departmentMapper.selectAll();  
 }  
}

**demo-web:**

@RestController  
public class DepartmentController {  
  
 @Autowired  
 private DepartmentService departmentService;  
  
 @RequestMapping("getAll")  
 public List<Department> getAll(){  
 List<Department> list = departmentService.getAll();  
 return list;  
 }  
}

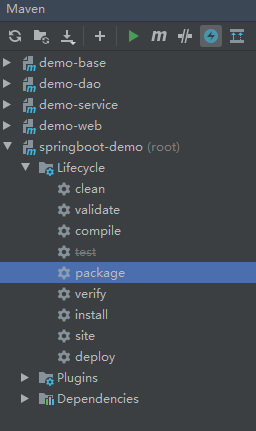
1. 在demo-web配置数据源

spring:  
 datasource:  
 username: root  
 password:  
 url: jdbc:mysql://127.0.0.1:3306/laravel?serverTimezone=UTC  
 driver-class-name: com.mysql.cj.jdbc.Driver  
mybatis:  
 mapper-locations: classpath:mapper/\*.xml  
 configuration:  
 map-underscore-to-camel-case: true

1. 将demo-web的启动类移到chq包下，因为springboot只会扫描启动类包名下
2. 运行项目，访问http://localhost:8080/getAll

## 打包部署

运行maven打包



将demo-web>target下的jar包传到服务器，运行：Java -jar 包名

# Springboot整合elasticsearch

## 环境准备

### ElasticSearch安装

要求JDK1.8！

下载：<https://www.elastic.co/>

以windows为例。版本7.8.0，下载解压即用

双击/bin/elasticsearch.bat运行。

访问：<http://localhost:9200/>

其他环境查看<https://www.elastic.co/cn/downloads/elasticsearch>下

Package Managers安装方式

### Kibana安装

Kibana是一个针对Elasticsearch的开源分析及可视化平台，用来搜索、查看交互存储在Elasticsearch索引中的数据。

在同个网站上下载，需要版本要和 Es 一致！解压即用。

双击/bin/kibana.bat运行

访问：<http://localhost:5601>

### 可视化界面

要求node环境

1、下载地址：<https://github.com/mobz/elasticsearch-head/>

2、启动

npm install

npm run start

3、连接测试发现，存在跨域问题：配置es（\config\elasticsearch.yml）

http.cors.enabled: true

http.cors.allow-origin: "\*"

4、重启es服务器，然后再次连接

## 案例

1.导入依赖声明版本：

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-elasticsearch</artifactId>  
</dependency>

<properties>  
 <elasticsearch.version>7.8.0</elasticsearch.version>  
</properties>

一定要保证 导入的依赖和es 版本一致

2.配置es

@Configuration  
public class ElasticsearchConfig {  
 @Bean  
 RestHighLevelClient restHighLevelClient() {  
 RestHighLevelClient client = new RestHighLevelClient(RestClient.*builder*(new HttpHost("127.0.0.1",9200,"http")));  
 return client;  
 }  
}

3.批量导入示例

List<Content> contents = htmlParseUtl.parseJD(); //要导入的数据  
BulkRequest bulkRequest = new BulkRequest();  
bulkRequest.timeout("2m");  
for (Content content: contents) {  
 bulkRequest.add(new IndexRequest("goods").source(JSON.*toJSONString*(content), XContentType.*JSON*));  
}  
  
BulkResponse bulk = restHighLevelClient.bulk(bulkRequest, RequestOptions.*DEFAULT*);  
System.*out*.println(bulk.hasFailures());

搜索示例：

SearchRequest searchRequest = new SearchRequest("goods");  
SearchSourceBuilder sourceBuilder = new SearchSourceBuilder();  
  
//分页  
sourceBuilder.from(0);  
sourceBuilder.size(5);  
  
//精准匹配  
TermQueryBuilder termQueryBuilder = QueryBuilders.*termQuery*("title","java");  
sourceBuilder.query(termQueryBuilder);  
sourceBuilder.timeout(new TimeValue(60, TimeUnit.*SECONDS*));  
  
//执行搜索  
searchRequest.source(sourceBuilder);  
SearchResponse searchResponse = restHighLevelClient.search(searchRequest,RequestOptions.*DEFAULT*);  
//解析结果  
for (SearchHit documentFields:searchResponse.getHits().getHits()){  
 System.*out*.println(documentFields.getSourceAsMap());  
}

更多操作查看：

<https://www.elastic.co/guide/en/elasticsearch/client/java-rest/current/index.html>

推荐使用：Java High Level REST Client

# Springboot爬虫

<!-- https://mvnrepository.com/artifact/org.jsoup/jsoup -->  
<dependency>  
 <groupId>org.jsoup</groupId>  
 <artifactId>jsoup</artifactId>  
 <version>1.10.2</version>  
</dependency>

@Data  
public class Content {  
 private String title;  
 private String img;  
 private String price;  
}

public ArrayList<Content> parseJD() throws IOException {  
 String url = "https://search.jd.com/Search?keyword=java";  
  
 Document document = Jsoup.*parse*(new URL(url),30000);  
 Element element = document.getElementById("J\_goodsList");  
 Elements element1s = element.getElementsByTag("li");  
  
 ArrayList<Content> goodsList = new ArrayList<>();  
  
 for (Element el:element1s){  
 String img = el.getElementsByTag("img").eq(0).attr("source-data-lazy-img");  
 String price = el.getElementsByClass("p-price").eq(0).text();  
 String title = el.getElementsByClass("p-name").eq(0).text();  
  
 Content content = new Content();  
 content.setImg(img);  
 content.setPrice(price);  
 content.setTitle(title);  
 goodsList.add(content);  
 }  
 return goodsList;  
}