# 2018-4-23

1. Javax.validation ----- bean validation

|- http://tanlan.iteye.com/blog/1099523

|-@NotNull(message = "The name of employee can not be null")

@Size(min = 1,max = 10,message="The size of employee's name must between 1 and 10")

private String name;

ValidatorFactory vf = Validation.buildDefaultValidatorFactory();

Validator validator = vf.getValidator();

Set<ConstraintViolation<Employee>> set = validator.validate(employee);

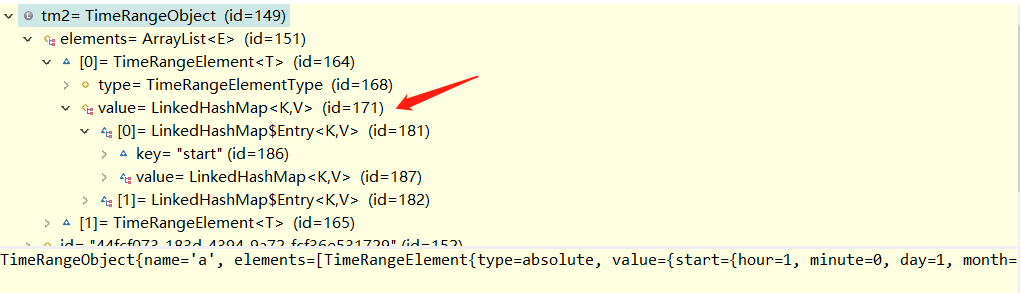
for (ConstraintViolation<Employee> constraintViolation : set) {

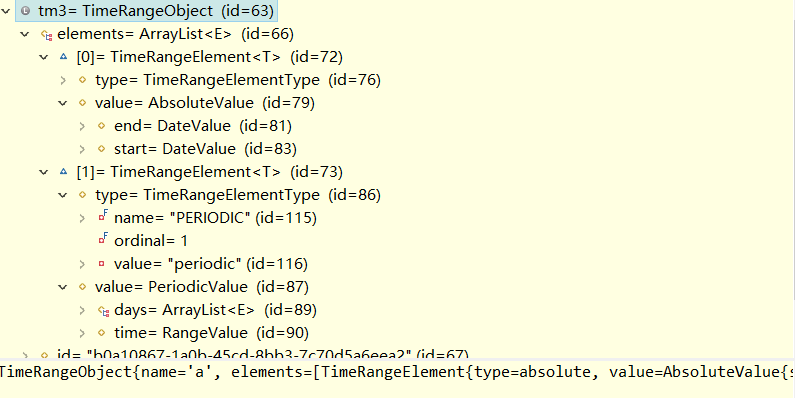
System.out.println(constraintViolation.getMessage());

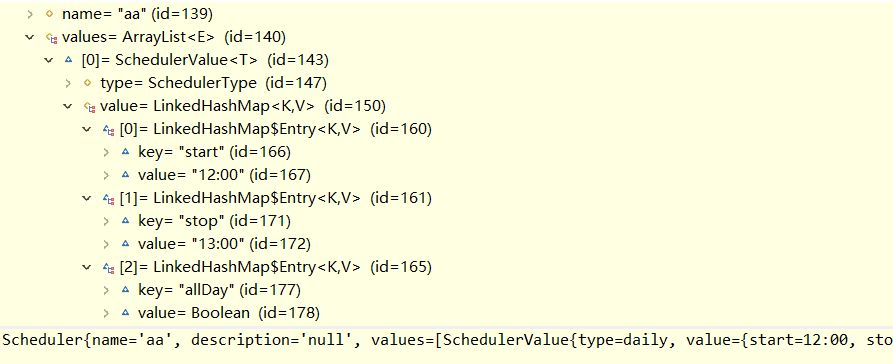
} //使用message用来定位非valid的属性

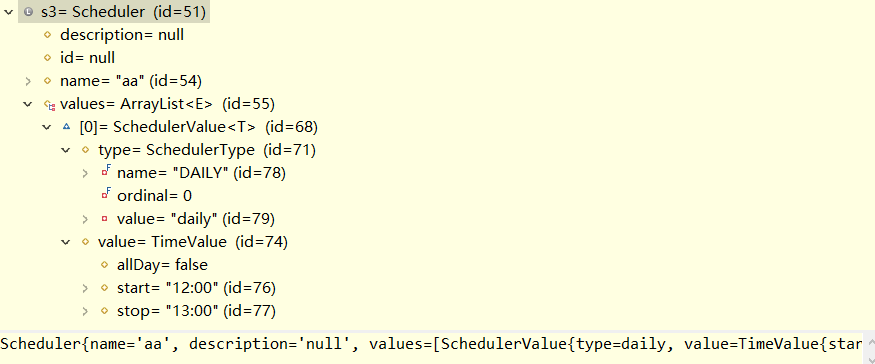
Bean Validation 规范对 Java Bean 的验证流程如下：

在实际使用中调用 Validator.validate(JavaBeanInstance) 方法后，Bean Validation 会查找在 JavaBeanInstance上所有的约束声明，对每一个约束调用对应的约束验证器进行验证，最后的结果由约束验证器的 isValid 方法产生，如果该方法返回 true，则约束验证成功，否则验证失败。验证失败的约束将产生约束违规对象（ConstraintViolation 的实例）并放到约束违规列表中。验证完成后所有的验证失败信息均能在该列表中查找并输出。









1. Json数据

|- <https://www.json.org/json-zh.html>

|- http://www.hankcs.com/program/java-object-to-json-demo.html

# 2018-4-24

1. SpringBoot

|-https://c.runoob.com/front-end/53 在线JSON转换工具

|- @Async //表明该方法是一个异步方法，如果在类级别，该类中所有方法为异步

|- @RestController注解相当于@ResponseBody ＋ @Controller合在一起的作用

|- @CrossOrigin(origins = "\*")应用于类上，代表该类中所有方法可以处理任意域上的请求？

1. 未完成事项

|- 分析策略开通的流程 http://oa.sky-cloud.net:8090/pages/viewpage.action?pageId=9536399

|- 对前后端数据交互的学习(后端数据的转换 在NAP平台上，181，进去后右上角的?按钮，API文档

|- Spring三件套的复习。。。

# 2018-4-25

1. @ApiOperation不是spring自带的注解是swagger里的 ，Swagger是当前最好用的Restful API文档生成的开源项目，时swagger-ui还可以测试spring restful风格的接口功能

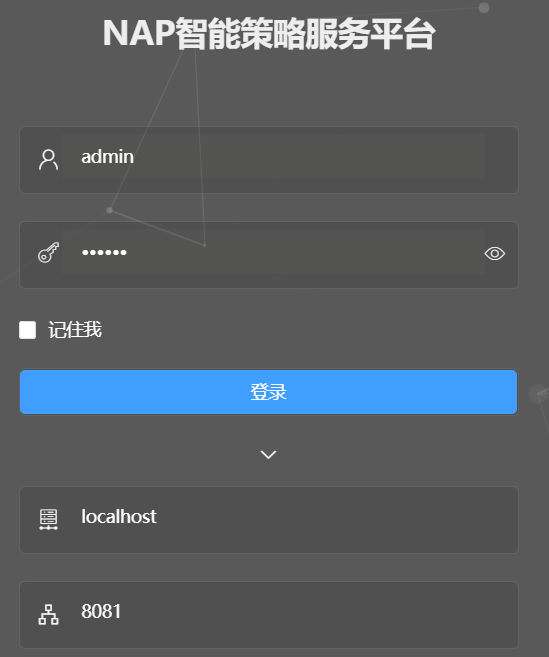
获取nap-web-alpha仓库，全程GIT上操作 切换到nap-web-alpha (feature/NAP-1092) 分支上面

|- npm config set registry <https://registry.npm.taobao.org> ---修改registry

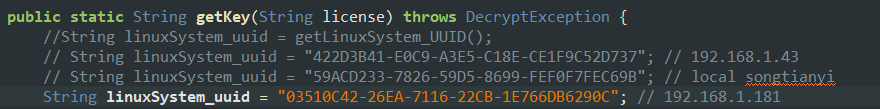
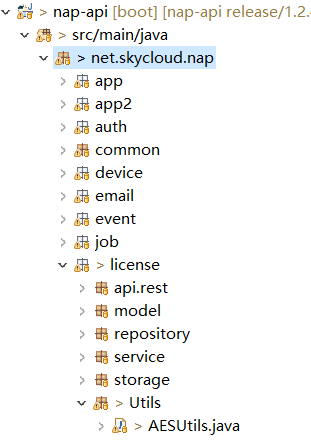
|- npm config list 确认registry配置信息

|- npm install

|- npm run dev

|- 在产品登录界面可以使用本地代码的逻辑

|-只需要在License的获取逻辑下，更改uuid，就可以连接远程的docker仓库镜像，使用es和postgresql



1. 在一台机器上将jar包复制到另一台机器 这一步其实在第二步之前就要完成，在Xshell上操作

|-scp build/libs/nap-api.xxx.jar [root@192.168.1.100:/data/builder/nap-api/nap-api.jar](mailto:root@192.168.1.100:/data/builder/nap-api/nap-api.jar) 第一个路径是灵活变化的

|- 登录到远程，上面的192.168.1.100

|- 切换到第二个路径后面，先把依赖的包给确定好，复制重写都可以。

|- docker build -t IMAGE\_NAME:VERSION . 按照规范写NAME和version(自主命名)，可以用命令查看已有的镜像

|- docker images

|- systemctl restart nap.service 重启服务

# 2018-4-26

1. Gradle

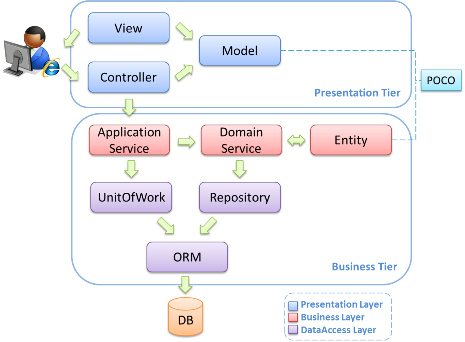
|- http://www.cnblogs.com/davenkin/p/gradle-learning-3.html

1. 在开发期间我们用Annotation注解，这样在一定程度上不仅可以省去对XML配置文件的维护，而且大大的提高了开发效率，缩短了开发周期。

开发后期，项目功能完成，我们可以将Annotation配置转换为XML配置，禁用Annotation即可。这样做的理由是如果项目上线，我们需要修改相关代码的配置，直接改XML、properties配置文件即可。这样就不需要开发人员找到相应的代码修改源代码、重新编译打包发布。而xml的配置是可以直接修改的，不需要重新编译，只需重启下你的服务器即可。

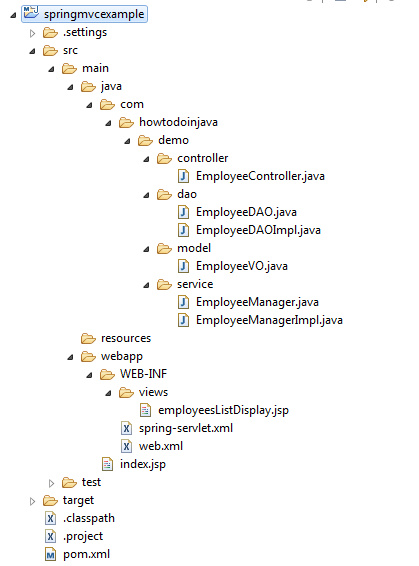
1. MVC主要结构是：Model,View,Controller，模型通常为pojo对象。MVC主要是在三层框架中的表示层。

|-https://blog.csdn.net/xiao\_\_gui/article/details/46803193

通常一个WEB应用是3层结构，数据-业务-表示

DAO负责与底层的数据库存储进行交互。

Service负责与DAO进行交互，主要是业务逻辑



1. Spring

|- https://www.journaldev.com/

# 2018-4-27

1. 幂等性

|-每一个有志与分布式领域的程序员都应该熟悉它的设计思想。

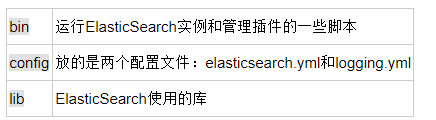
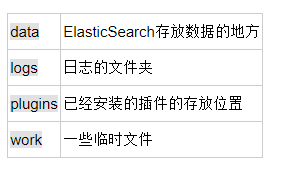
1. Elasticsearch的部署

|-一般不要去动它的配置，不存在什么“神奇的配置”去优化elasticsearch，默认配置0.0.0.0地址

|- 要优化也是优化它的数据

|- 参考nap-docs里面的build/下的部署方案

|- <http://dockone.io/article/3655>



1. postgreSQL

|- docker restart postgres 重启 postgre的container

|-参考nap-docs里面的build/下的部署方案

|- /srv/nap/postgresql/data/postgresql.conf 在文件中查找关键字listen，设置监听端口

|-/srv/nap/postgresql/data/pg\_hba.conf 更改ipv4的host地址为0.0.0.0/0,代表所有IP都能链接

|- <http://blog.51cto.com/corasql/1910485>

|- <https://www.elastic.co/guide/cn/elasticsearch/guide/current/important-configuration-changes.html>

|- <http://www.ruanyifeng.com/blog/2017/08/elasticsearch.html> es入门

# TYPE DATABASE USER ADDRESS METHOD

# "local" is for Unix domain socket connections only

local all all trust

# IPv4 local connections:

host all all 0.0.0.0/0 trust

# IPv6 local connections:

host all all ::1/128 trust

# Allow replication connections from localhost, by a user with the

# replication privilege.

local replication all trust

host replication all 127.0.0.1/32 trust

host replication all ::1/128 trust

1. ACL执行顺序

# 2018-4-28

10.1.1.0/24

10.2.2.0/24

172.16.1.0/24

172.16.22.0/24

192.168.24.0/8

192.168.10.0/24