# 自制注解实现验证框架

## 1 环境

- java 8
- SpringBoot 2.0.3.RELEASE
- Maven 3.6.1

## 2 搭建环境

- idea快速创建Spring项目，勾选`Spring Web`模块
- 相关pom依赖

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

## 3 项目代码

### 实体类

UserBean

```
package com.lyb.mvc.pojo;


import com.lyb.mvc.anno.Enumration;

import javax.validation.constraints.Max;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class UserBean {
    private Integer uid;

    @NotBlank
```

```
    private String uname;

    @Enumration(options = {"男","女"})
    private String sex;

    @Min(value = 1)
    @Max(value = 100)
    @NotNull
    private Integer age;

    @NotBlank
    @Enumration(value = "sf")
    private String address;


}
```

- 这是一些相关注解的介绍，但是现在我们只是使用了
    - @NotNull：不能为null，但可以为empty
    - @NotEmpty：不能为null，而且长度必须大于0
    - @NotBlank：只能作用在String上，不能为null，而且调用trim()后，长度必须大于0
    - @Null 被注释的元素必须为null
    - @NotNull 备注是的元素不能null
    - @AssertTrue 被注释的元素必须为true
    - @AssertFalse 被注释的元素必须是false
    - @Min(value) 被注释的元素必须是一个数字，其值必须大于等于指定的最小值
    - @Max(value) 被注释的元素必须是一个数字，其值必须小于等于指定的最大值
    - @DecimalMin(value) 被注释的元素必须是一个数字，其值必须大于等于指定的最小值
    - @DecimalMax(value) 被注释的元素必须是一个数字，其值必须小于等于指定的最大值
    - @Size(max,min) 被注释的元素的大小必须在指定的范围内
    - @Digits(integer,fraction) 被注释的元素必须是一个数字，其值必须在可接受的范围内
    - @Past 被注释的元素必须是一个过去的日期
    - @Future 被注释的元素必须是一个将来的日期
    - @Pattern(value) 被注释的元素必须符合指定的正则表达式
    - @Email 被注释的元素必须是电子邮箱地址
    - @Length 备注是的字符串的大小必须在指定的范围内
    - @NotEmpty 被注释的字符串必须非空
    - @Range 被注释的元素必须在合理的范围内

ResultBean

```
package com.lyb.mvc.pojo;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class ResultBean<T> {
    private boolean success;

    private String msg;

    private T data;
```

```
    public static <T> ResultBean<T> getSuccess(T data) {
        ResultBean<T> bean = new ResultBean<T>();
        bean.setData(data);
        bean.setSuccess(true);
        bean.setMsg("success");
        return bean;
    }

    public static <T> ResultBean<T> getFaild(String msg, T data) {
        ResultBean<T> bean = new ResultBean<>();
        bean.setData(data);
        bean.setSuccess(false);
        bean.setMsg(msg);
        return bean;
    }


}
```

## 自定义注解

Enumration

```
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface Enumration {
    String[] options() default {};
    String value() default "";
}
```

## DAO

```
package com.lyb.mvc.dao;

import com.lyb.mvc.pojo.UserBean;
import org.springframework.stereotype.Repository;

import java.util.*;

@Repository
public class UserDao {

    private static Map<Integer, UserBean> userMap = new HashMap<>();

    public boolean addUser(UserBean user) {
        userMap.put(user.getUid(), user);
        System.out.println("添加用户成功, username:" + user.getUname());
        return true;
    }
```

```
    public static List<String> getEnumList(String enumName){
        //如果传入的是 sf 就返回 省份的List
        if ("sf".equalsIgnoreCase(enumName)){
            return getSFList();
        }
        //都不符合就返回空的list
        return Collections.EMPTY_LIST;
    }

    public static List<String> getSFList(){
        //按道理是从数据库获取的
        List<String> SFlist = new ArrayList<>();
        SFlist.add("北京");
        SFlist.add("上海");
        SFlist.add("深圳");
        SFlist.add("广州");
        SFlist.add("广西");
        SFlist.add("新疆");
        SFlist.add("西藏");
        SFlist.add("海南");
        return SFlist;
    }


}
```

## Service

UserService

```
@Service
public class UserService {

    @Autowired
    private UserDao userDao;

    public boolean addUser(UserBean user) {
        Eutil.validate(user);
        return userDao.addUser(user);
    }

}
```

## Utils

Eutil

```
package com.lyb.mvc.utils;
```

```java
import com.lyb.mvc.anno.Enumration;
import com.lyb.mvc.dao.UserDao;
import org.springframework.util.CollectionUtils;

import javax.validation.constraints.*;
import java.lang.reflect.Field;
import java.util.Arrays;
import java.util.List;

public class Eutil {
    public static boolean isNvlOrBlank(Object obj) {
        if (obj == null || obj.toString().trim().length() == 0) {
            return true;
        }
        return false;
    }

    public static void checkRequireParam(String name, Object... strs) {
        for (Object str : strs) {
            if (isNvlOrBlank(str)) {
                throw new RuntimeException(String.format("param %s cannot
empty.", name));
            }
        }
    }

    public static <T> void validate(T obj) {
        Field[] fields = obj.getClass().getDeclaredFields();
        try {
            for (Field field : fields) {
                NotBlank blankAnno = field.getAnnotation(NotBlank.class);
                NotNull nullAnno = field.getAnnotation(NotNull.class);
                Enumration enumAnno = field.getAnnotation(Enumration.class);
                Size sizeAnno = field.getAnnotation(Size.class);
                //获取属性值
                field.setAccessible(true);
                //获取界面传过来的值
                Object value = field.get(obj);
                //实体类中的属性的名称
                String name = field.getName();
                System.out.println(name);

                //判断 NotBlank 注解
                if (blankAnno != null) {
                    checkRequireParam(name, value);
                }
                //判断 NotNull 注解
                if (nullAnno != null) {
                    if (value == null) {
                        throw new RuntimeException(String.format("param %s
cannot be null", name));
                    }
                }
                if (sizeAnno != null) {
                    //为Integer类型
                    if (value instanceof Integer) {
                        int tmpval = (Integer) value;
                        if (tmpval < sizeAnno.min()) {
```

```
                            throw new RuntimeException(String.format("param %s
is too small.", name));
                        } else if (tmpval > sizeAnno.max()) {
                            throw new RuntimeException(String.format("param %s
is too large.", name));
                        }
                    } else {
                        throw new RuntimeException(String.format("The value type
of parameter %s is not correct.", name));
                    }
                }
                if (enumAnno != null) {
                    String[] options = enumAnno.options();
                    List<String> list = null;
                    //先判断是options还是value
                    if (options.length == 0) {
                        //不是options，就获取value值
                        String enumName = enumAnno.value();
                        //看看注解上的 value是否为空
                        if (!isNvlOrBlank(enumName)) {
                            //若不是空，就通过注解上的 value 获取相对应的list
                            list = UserDao.getEnumList(enumName);
                        }
                    } else {
                        //若为options，就把options作为list
                        list = Arrays.asList(options);
                    }
                    //对list进行非空判断，为空就不用进行下去了
                    if (CollectionUtils.isEmpty(list)) {
                        continue;
                    }
                    //强制转换为String
                    String val = getStr(value);
                    if (!list.contains(val)) {
                        throw new RuntimeException(String.format("param %s value
not exists.", name));
                    }
                }

            }
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    public static String getStr(Object obj) {
        return obj.toString();
    }
}
```

### Controller

UserController

```
@Controller
@RequestMapping("/user")
public class UserController {

    @Autowired
    private UserService userService;

    @PostMapping
    @RequestMapping("/add")
    @ResponseBody
    public ResultBean<Boolean> addUser(@RequestBody UserBean user) {
        return ResultBean.getSuccess(userService.addUser(user));
    }
}
```

## 4 测试

- 使用postman进行测试