链滴

枚举类的业务实践

作者：valarchie
原文链接：https://ld246.com/article/1571237898110
来源网站：链滴

业务场景

在进行业务开发时经常会有状态值的业务需要，例如一场考试有未开考、考试中、考试结束等状态或一年四季有春天、夏天、秋天、冬天等状态。从前台传参到我们的业务模型再到数据库，其实这些状是贯穿整个开发流程的。如果仅仅使用1、2、3、4来代表春夏秋冬的话，那代码的自解释能力就太差。那如何利用好枚举类来更优雅的编码呢?

枚举类需要满足的功能

1. 消除代码中的魔术数
2. 根据数值获取状态的名称
3. 根据数值获取状态
4. 获取数值和状态名称为键值对的Map作为前端筛选项。

第一种枚举类实现

```
public enum SeasonEnum {

    /**/
    SPRING(1,"春天"),
    SUMMER(2,"夏天"),
    AUTUMN(3,"秋天"),
    WINTER(4,"冬天");

    public final int id;
    public final String desc;

    SeasonEnum(int id, String desc) {
        this.id = id;
        this.desc = desc;
    }


    /**
     * 根据数值获取对应的枚举
     * @param id
     * @return
     */
    public static SeasonEnum getEnumById(int id) {
        for (SeasonEnum seasonEnum : SeasonEnum.values()) {
            if (seasonEnum.id == id) {
                return seasonEnum;
            }
        }
        return null;
    }


    /**
     * 根据数值获取对应的枚举描述
```

```
     * @param id
     * @return
     */
    public static String getEnumDescById(int id) {
        final String unknown = "未知";
        for (SeasonEnum seasonEnum : SeasonEnum.values()) {
            if (seasonEnum.id == id) {
                return seasonEnum.desc;
            }
        }
        return unknown;
    }


    /**
     * 获取数值-描述键值对map
     * @return
     */
    public static Map getEnumMap() {
        Map<Object, Object> map = new HashMap<>();
        for (SeasonEnum seasonEnum : SeasonEnum.values()) {
            map.put(seasonEnum.id, seasonEnum.desc);
        }
        return map;
    }

}
```

测试实现：

```
    int seasonType = 2;
    // 如果是夏天的话
    if (seasonType == SeasonEnum.SUMMER.id) {
        System.out.println("穿短袖！ ");
    }
    // 根据id获取对应枚举类
    System.out.println(SeasonEnum.getEnumById(seasonType));
    // 根据id获取对应枚举类描述
    System.out.println(SeasonEnum.getEnumDescById(seasonType));
    // 获取id-描述键值对map
    Map seasonMap = SeasonEnum.getEnumMap();
    System.out.println(seasonMap);
```

运行结果：

```
穿短袖!
SPRING
春天
{1=春天, 2=夏天, 3=秋天, 4=冬天}
```

通过通过枚举类，以及getEnumById()、getEnumDescById()、getEnumMap()方法我们实现了一始设置业务关于枚举类的需求。

但是当系统内的枚举类剧增的时候，所有枚举类都需要实现getEnumById()、getEnumDescById()、

etEnumMap()这三个方法。能不能将这三个方法都抽取出来呢?

第二种枚举类实现

因为enum枚举类其实也是一个Java类，一个相当于继承Enum的类，所以不能再通过继承来声明共的方法，而只能使用接口来声明共有的方法。

我们通过一个IEnum接口来声明共有的getId()、getDesc()方法。

```
public interface IEnum<V, D> {
    /**
     * 获取id
     * @return
     */
    V getId();
    /**
     * 获取描述
     * @return
     */
    D getDesc();
}
```

实现了IEnum接口的SeasonEnum枚举类。

```
public enum SeasonEnum implements IEnum {

    /**/
    SPRING(1,"春天"),
    SUMMER(2,"夏天"),
    AUTUMN(3,"秋天"),
    WINTER(4,"冬天");

    public final int id;
    public final String desc;

    SeasonEnum(int id, String desc) {
        this.id = id;
        this.desc = desc;
    }

    @Override
    public Object getId() {
        return id;
    }

    @Override
    public Object getDesc() {
        return desc;
    }
}
```

接着我们再根据一个枚举工具类EnumUtil来封装getEnumById()、getEnumDescById()、getEnumap()这三个方法。

```
public class EnumUtil {

    private static final String UNKNOW = "未知";

    /**
     * 根据id获取枚举类
     * @param id
     * @param type
     * @param <E>
     * @param <V>
     * @return
     */
    public static<E extends IEnum, V> E getEnumById(V id, Class<E> type) {
        if (!type.isEnum()) {
            throw new IllegalArgumentException("Type: " + type + " must be a enum");
        }

        for (E iEnum : type.getEnumConstants()) {
            if (iEnum.getId().equals(id)) {
                return iEnum;
            }
        }
        return null;
    }


    /**
     * 根据id获取对应枚举的描述
     * @param id
     * @param type
     * @param <E>
     * @param <V>
     * @return
     */
    public static <E, V> E getEnumDescById(V id, Class<? extends IEnum> type) {

        IEnum iEnum = getEnumById(id, type);

        if (iEnum != null) {
            return (E) iEnum.getDesc();
        } else {
            return (E) UNKNOW;
        }

    }


    /**
     * 获取枚举的id-描述键值对
     * @param type
     * @return
     */
    public static Map<Object, Object> getEnumMap(Class<? extends IEnum> type) {
```

```
        if (!type.isEnum()) {
            throw new IllegalArgumentException("Type: " + type + " must be a enum");
        }

        Map<Object, Object> map = new HashMap<>(type.getEnumConstants().length);

        for (IEnum iEnum : type.getEnumConstants()) {
            map.put(iEnum.getId(), iEnum.getDesc());
        }

        return map;
    }

}
```

测试实现：

```
  public static void main(String[] args) {

        int seasonType = 2;
        // 如果是夏天的话
        if (seasonType == SeasonEnum.SUMMER.id) {
            System.out.println("穿短袖！");
        }
        // 根据id获取对应枚举类
        System.out.println(EnumUtil.getEnumById(seasonType, SeasonEnum.class));
        // 根据id获取对应枚举类描述
        System.out.println(EnumUtil.getEnumDescById(seasonType, SeasonEnum.class));
        // 获取id-描述键值对map
        Map seasonMap = EnumUtil.getEnumMap(SeasonEnum.class);
        System.out.println(seasonMap);

        switch (EnumUtil.getEnumById(seasonType, SeasonEnum.class)) {

            case SUMMER:
                System.out.println("穿短袖！");
                break;
            case WINTER:
                System.out.println("穿羽绒服！");
                break;
            case SPRING:
            case AUTUMN:
                System.out.println("穿春秋装！");

            default:
                break;

        }


    }
```

运行结果：

穿短袖!
SUMMER
夏天
{1=春天, 2=夏天, 3=秋天, 4=冬天}
穿短袖!

通过IEnum接口以及EnumUtil的封装我们消除了之前Enum类当中冗余的代码，并较为良好的利用了举类的优点。

有其他好的枚举业务实现，恳请网友不吝赐教指点一下~

笔者个人心得，如有错误恳请网友评论指正。

转自我的个人博客 vc2x.com