liuxi--ganbare

liuxi1024

失业了，开始找工作了
2010-02-08 通过*Firefox*插件
>>更多闲聊

设计模式--使用java内置的观察者模式（Obse ... | 基础巩固--jsp中cookie使用

2010-01-29

设计模式--spring源码中使用策略模式（Strategy Pattern）

文章分类:软件开发管理

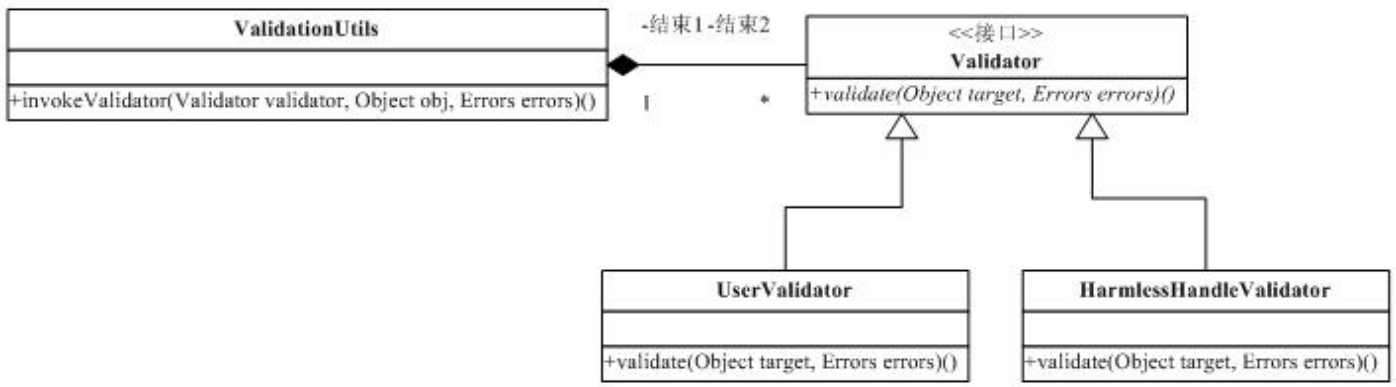
策略模式（Strategy Pattern）中体现了两个非常基本的面向对象设计的基本原则：封装变化的概念；编程中使用接口，而不是对接口实现。策略模式的定义如下：

定义一组算法，将每个算法都封装起来，并且使它们之间可以互换。策略模式使这些算法在客户端调用它们的时候能够互不影响地变化。

策略模式使开发人员能够开发出由许多可替换的部分组成的软件，并且各个部分之间是弱连接的关系。弱连接的特性使软件具有更强的可扩展性，易于维护；更重要的是，它大大提高了软件的可重用性。

下面使用spring中源码说明策略模式(spring validation)

1、UML图说明



说明：UserValidator、HarmlessHandleValidator 分别为两个行为策略，实现不同的算法。

2、类和接口代码

Class: org.springframework.validation.ValidationUtils 验证工具类

Java代码

```
1.  public static void invokeValidator(Validator validator, Object obj, Errors errors) {
2.                  Assert.notNull(validator, "Validator must not be null");
3.                  Assert.notNull(errors, "Errors object must not be null");
```

storm1127 blueram

博客分类

其他分类

最近加入圈子

存档

最新评论

```
4.                  if (logger.isDebugEnabled()) {
5.                          logger.debug("Invoking validator [" + validator + "]");
6.                  }
7.                  if (obj != null && !validator.supports(obj.getClass())) {
8.                          throw new IllegalArgumentException(
9.                                          "Validator [" + validator.getClass() + "] does not sup
    port [" + obj.getClass() + "]");
10.                 }
11.                 validator.validate(obj, errors);
12.                 if (logger.isDebugEnabled()) {
13.                         if (errors.hasErrors()) {
14.                                 logger.debug("Validator found " + errors.getErrorCount() + " e
    rrors");
15.                         }
16.                         else {
17.                                 logger.debug("Validator found no errors");
18.                         }
19.                 }
20.         }
```

Interface:org.springframework.validation.Validator

Java代码

```
1.  public interface Validator {
2.          boolean supports(Class clazz);
3.          void validate(Object target, Errors errors);
4.
5.  }
```

Class:UserValidator

Java代码

```
1.  public class UserValidator implements Validator {
2.
3.          @Override
4.          public boolean supports(Class clazz) {
5.                  return User.class.equals(clazz);
6.          }
7.
8.          @Override
9.          public void validate(Object target, Errors errors) {
10.                 User user = (User) target;
11.                 if (!StringUtils.hasLength(user.getUsername())) {
```

利用DOM4J创建XML格式字 ...

谢谢！！顶！

-- by hanzhu

评论排行榜

- 利用DOM4J创建XML格式字串
- spring中使用Quartz完成时序调度工作
- 数据采集--连接Access数据库
- 设计模式--外观模式(Facade Pattern)
- jdbc操作数据库的步骤

[什么是RSS?]

```
12.                     errors.rejectValue("username", "", "登录编码必须填写！");
13.             }
14.             if (!StringUtils.hasLength(user.getPassword())) {
15.                     errors.rejectValue("password", "", "登录密码必须填写！");
16.             }
17.             if (!StringUtils.hasLength(user.getName())) {
18.                     errors.rejectValue("name", "", "用户姓名必须填写！");
19.             }
20.         }
21.
22. }
```

Class:HarmlessHandleValidator

Java代码

```
1.  public class HarmlessHandleValidator implements Validator {
2.
3.          @Override
4.          public boolean supports(Class clazz) {
5.                  return HarmlessHandle.class.equals(clazz);
6.          }
7.
8.          @Override
9.          public void validate(Object target, Errors errors) {
10.                 HarmlessHandle harmlessHandle = (HarmlessHandle) target;
11.                 if (!StringUtils.hasLength(harmlessHandle.getHandleNo())) {
12.                         errors.rejectValue("handleNo", "", "编码必须填写！");
13.                 }
14.                 if (!StringUtils.hasLength(harmlessHandle.getHandleName())) {
15.                         errors.rejectValue("handleName", "", "名称必须填写！");
16.                 }
17.         }
18.
19. }
```

调用各自的validate策略

Java代码

```
1.  ValidationUtils.invokeValidator(new UserValidator(), user, errors);
```

3、策略模式的本质：少用继承，多用组合

over^^

查看图片附件

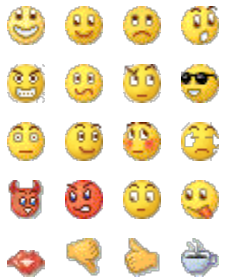
设计模式--使用java内置的观察者模式（Obse ... | 基础巩固--jsp中cookie使用

10:59 | 浏览 (34) | 评论 (0) | 分类: 设计模式 | 相关推荐

评论

发表评论

表情图标



B I U Quote Code List Img URL Flash Table

字体颜色: 标准 字体大小: 标准 对齐: 标准

提示：选择您需要装饰的文字, 按上列按钮即可添加上相应的标签

您还没有登录，请登录后发表评论(快捷键 Alt+S / Ctrl+Enter) 提交