Custom Validation trong Struts 2

Bạn có thể định nghĩa trình logic riêng cho Validation của bạn (custom validation) trong Struts 2 bởi triển khai **Validateable Interface** trong lớp Action. **Workflow Interceptor**được sử dụng để lấy thông tin về các thông điệp lỗi (error message) được định nghĩa trong lớp action.

Workflow Interceptor trong Struts 2

Workflow Interceptor kiểm tra xem có hay không bất cứ error nào. Nó không thực hiện bất cứ trình validation nào. Nó được áp dụng khi lớp action triển khai Validateable Interface. Tham số mặc định là input cho Interceptor này để xác định result để được triệu hồi cho action hoặc field error. Workflow Interceptor được tìm thấy trong defaultStack vì thế bạn không cần xác định nó một cách tường minh.

Chỉ có một tham số được định nghĩa cho Workflow Interceptor, đó là:

inputResultName: xác định result name để được trả về nếu field error hoặc action error được tìm thấy. Theo mặc định nó được thiết lập là input.

Giới thiệu Validatebale Interface

Validatebale Interface phải được triển khai để thực hiện trình logic cho validation trong lớp Action. Nó chỉ chứa một phương thức là validate() mà phải được ghi đè trong lớp action để định nghĩa trình logic cho validation. Cú pháp cho phương thức validate là:

```
public void validate();
```

Giới thiệu ValidationAware Interface

ValidationAware Interface có thể chấp nhận các thông điệp lỗi ở cấp độ trường hoặc cấp độ lớp action. Các thông điệp ở cấp độ trường (field level) được giữ trong Map và các thông điệp cấp độ lớp Action (action class level) được giữ trong Collection. Nó nên được triển khai bởi lớp Action để thêm bất cứ thông điệp lỗi nào.

Các phương thức của ValidationAware Interface

- **void addFieldError(String fieldName,String errorMessage)**: thêm thông điệp lỗi cho field đã cho.
- **void addActionError(String errorMessage)**: Thêm một thông điệp lỗi action level cho action này.

- **void addActionMessage(String message)**: Thêm một thông điệp action level cho action này.
- **void setFieldErrors(Map<String,List<String>> map)**: Thiết lập một collection của các thông điệp lỗi cho field.
- **void setActionErrors(Collection<String> errorMessages)**: Thiết lập một collection của các thông điệp lỗi cho action này.
- **void setActionMessages(Collection<String> messages)**: Thiết lập một collection của các thông điệp cho action này.
- **boolean hasErrors()**: Kiểm tra xem nếu có bất cứ field error hoặc action error nào.
- **boolean hasFieldErrors()**:Kiểm tra xem nếu có bất cứ field error nào.
- **boolean hasActionErrors()**:Kiểm tra xem nếu có bất cứ action error nào.
- **boolean hasActionMessages()**: Kiểm tra xem nếu có bất cứ thông điệp ở action level nào.
- **Map<String,List<String>> getFieldErrors()**: Trả về tất cả thông điệp lỗi ở cấp độ trường.
- **Collection<String> getActionErrors()**:Trả về tất cả thông điệp lỗi ở cấp độ action.
- **Collection<String> getActionMessages()**:Trả về tất cả thông điệp ở cấp độ action.

Ghi chú: Lớp ActionSupport triển khai Validateable và ValidationAware interface, vì thế chúng ta có thể kế thừa lớp ActionSupport để định nghĩa trình logic cho validation và các thông điệp lỗi.

Các bước để thực hiện Custom Validation trong Struts 2

Bạn theo các bước sau:

- Tạo form để nhận input từ người dùng.
- Định nghĩa trình logic cho validation trong lớp action bởi kế thừa lớp ActionSupport và ghi đè phương thức validate.
- Định nghĩa result cho thông điệp lỗi trong struts.xml file.

Ví dụ để thực hiện Custom Validation trong Struts 2

Trong ví dụ này, chúng ta tạo 4 page:

Tạo index.jsp

Cho input từ người dùng. Nó nhận name, password, và email id từ người dùng.

```
<%@ taglib uri="/struts-tags" prefix="s" %>
<s:form action="register">
<s:textfield name="name" label="Name"></s:textfield>
<s:password name="password" label="Password"></s:password>
<s:submit value="register"></s:submit>
</s:form>
```

Tạo lớp action

:

Lớp này kế thừa lớp ActionSupport và ghi đè phương thức validate.

RegisterAction.java

```
package com.vietjack;
import com.opensymphony.xwork2.ActionSupport;

public class RegisterAction extends ActionSupport{
private String name,password;
public void validate() {
         if(name.length()<1)
                  addFieldError("name","Name khong duoc de trong");
         if(password.length()<6)
                  addFieldError("password","Password phai lon hon 5");
}

//phuong thuc getter va setter

public String execute(){
//Thuc hien trinh logic o day
```

```
        return "success";
}
}
```

Định nghĩa input result trong struts.xml

Được triệu hồi nếu có bất cứ thông điệp lỗi nào được tìm thấy trong lớp Action.

struts.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts
 Configuration 2.1//EN" "http://struts.apache.org/dtds/struts-2.1.dtd">
<struts>

<package name="default" extends="struts-default">
<action name="register" class="com.vietjack.RegisterAction">
<result>welcome.jsp</result>
<result name="input">index.jsp</result>
</action>
</package>
</struts>
```

Tạo thành phần view

JSP file đơn giản này hiển thị thông tin của người dùng.

welcome.jsp

```
<%@ taglib uri="/struts-tags" prefix="s" %>
Name:<s:property value="name"/><br/>
Password:<s:property value="password"/><br/>
```

Định nghĩa thông điệp lỗi cấp độ action

Thông điệp lỗi action level làm việc cho toàn bộ form. Bạn có thể định nghĩa nó bởi phương thức **addActionError()** của ValidationAware Interface trong phương thức validate(). Ví dụ:

```
package com.vietjack;
```

```
import com.opensymphony.xwork2.ActionSupport;
public class RegisterAction extends ActionSupport{
private String name,password,email;
public void validate() {
         if(name.trim().length()<1 || password.trim().length()<1){
                  addActionError("Field khong duoc de trong");
         }
}

//phuong thuc getter va setter

public String execute(){
         return "success";
}
}
```

Bây giờ, bạn cần sử dụng thẻ **actionerror** trong index.jsp để hiển thị thông điệp lỗi cấp độ action.

index.jsp

```
<%@ taglib uri="/struts-tags" prefix="s" %>
<s:actionerror/>
<s:form action="register">
<s:textfield name="name" label="Name"></s:textfield>
<s:password name="password" label="Password"></s:password>
<s:textfield name="email" label="Email Id"></s:textfield>
<s:submit value="register"></s:submit>
</s:form>
```