Validation

Summary

There are pros and cons for considering validation as business logic, and Spring offers a design for validation (and data binding) that does not exclude either one of them. Specifically validation should not be tied to the web tier, should be easy to localize and it should be possible to plug in any validator available. Considering the above, Spring has come up with a Validator interface that is both basic ands eminently usable in every layer of an application.
External validators such as Jakarta Commons Validator or Valang can be used at Spring framework.
For how to use Jakarta Commons Validator with Spring Modules, see Use Commons Validator at Spring Framework.

Description

Spring features a Validator interface that you can use to validate objects. The Validator interface works using an Errors object so that while validating, validators can report validation failures to the Errors object.

```
package com.easycompany.domain;

public class Department {

        private String deptid;    //Department ID
        private String deptname;   //Department name
        private String superdeptid;   //Upper department ID
        private String superdeptname; //Upper department name
        private String depth;   //Department level
        private String description;   //Department description

        //setter/getter of above properties
}
```

Realization of Validator

Method of interface org.springframework.validation.Validator is as follows:

- boolean supports(Class clazz) : can Validator support for given object(clazz)?
- void validate(Object target, Errors errors) : perform validation for given object(target). If validation error occurs, related information is saved in the given Errors object.

Above two methods should be implemented when creating the implementation Validator class.

Let's create Department DepartmentValidator for checking validation.
Validation condition is that department name(deptname) property should have value and the length of input value of department description property should be over 10.

```
package com.easycompany.validator;

import org.springframework.validation.Errors;
import org.springframework.validation.Validator;
import com.easycompany.domain.Department;

public class DepartmentValidator implements Validator {

        public boolean supports(Class clazz) {
                return Department.class.isAssignableFrom(clazz);
        }

        public void validate(Object target, Errors errors) {
                Department department = (Department)target;

                if(isEmptyOrWhitespace(department.getDeptname())){ //does department name
property exist?
                        errors.rejectValue("deptname", "required");
                }

                if(department.getDescription()==null ||
department.getDescription().length()<10){ //value length of department description property iis over
10?
                        errors.rejectValue("description", "lengthsize", new Object[]{10},
"description's length must be larger than 10.");
                }
        }

        public boolean isEmptyOrWhitespace(String value){
                if(value==null || value.trim().length() == 0){
                        return true;
                }else{
                        return false;
                }
        }
}
```

If validation fails like above code, rejectValue method of Errors interface is executed. See here for details on Errors interface.

errors.rejectValue("deptname", "required");
⇒ An error occurs when validating deptname property and the related message key is "required".

errors.rejectValue("description", "lengthsize", new Object[]{10}, "description's length must be larger than 10.");
⇒ An error occurs in validating description property. Related message key is "lengthsize" and the argument to be delivered to the message is 10.
If relevant message key does not exist, it means using the message "description's length must be larger than 10.".

The spring provides the utility class of ValidationUtils for validation check.
The area of checking whether department name property(deptname) value is null or white space can be created using rejectIfEmptyOrWhitespace method of ValidationUtils.

```
package com.easycompany.validator;

import org.springframework.validation.Errors;
import org.springframework.validation.Validator;
import org.springframework.validation.ValidationUtils;
import com.easycompany.domain.Department;

public class DepartmentValidator implements Validator {

        public boolean supports(Class clazz) {
                return Department.class.isAssignableFrom(clazz);
        }

        public void validate(Object target, Errors errors) {

                Department department = (Department)target;

                ValidationUtils.rejectIfEmptyOrWhitespace(errors, "deptname", "required");
```

```
                    if(department.getDescription()==null ||
department.getDescription().length()<10){ //Department description property has input value length
over 10?
                              errors.rejectValue("description", "lengthsize", new Object[]{10},
"description's length must be larger than 10.");
                    }
          }
}
```

Error Message Configuration

Configure the message on the message keys, “required” and “lengthsize” on the message property file.

```
required=required input value.
Should enter lengthsize={0} letters or over.
```

Validation in Controller

Apply the code that runs validation on the controller.
If you want to validate validity on the instance of sending the form, add the validation code on the onSubmit method in the com.easycompany.controller.annotation.UpdateDepartmentController.

```
package com.easycompany.controller.annotation;
...
import org.springframework.validation.BindingResult;
import com.easycompany.validator.DepartmentValidator;

@Controller
public class UpdateDepartmentController {

          //If the user finishes data modification and presses the Save button, it calls the service (DB)
taking charge of saving.
          //If it is successfully saved, move to the department list page and if there is an error, move to
input form page again.
          @RequestMapping(value = "/updateDepartment.do", method = RequestMethod.POST)
          public String onSubmit(@ModelAttribute("department") Department department,
BindingResult bindingResult) {

                    //validation code
                    new DepartmentValidator().validate(department, bindingResult); //perform
validation.
                    if(bindingResult.hasErrors()){ //if there is a validation error,
                              return "modifydepartment";   //move to this page.
                    }

                    try {
                              departmentService.updateDepartment(department);
                              return "redirect:/departmentList.do?depth=1";
                    } catch (Exception e) {
                              e.printStackTrace();
                              return "modifydepartment";
                    }
          }
}
```

JSP

Another mechanism for registering property editors with the Spring container is to create and use a PropertyEditorRegistrar.

/easycompany/webapp/jsp/annotation/modifydepartment.jsp

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
...
<form:form commandName="department">
<table>
...
		<tr>
			<th>department name</th>
			<td><form:input path="deptname" size="20"/><form:errors path="deptname"
/></td>
		</tr>
...
		<tr>
			<th>description</th>
			<td><form:textarea path="description" rows="10" cols="40"/><form:errors
path="description" /></td>
		</tr>
</table>
</form:form>
...
```

TEST

Clear the name value, click on store after inserting department descriptions less than 10 characters, return to department data update page, and following error message will be printed.

Reference

- Spring Framework API Documentation 2.5.6