CreatingCustomValidators

Creating a custom validator

Explains how to write and integrate custom validators for Tapestry 4 and 5

Tapestry 5.0

Author: Sebastian Hennebrueder, Date: August 2009 There are 3 steps:

1. Create a validator
2. Create a resource file for validation messages
3. Register both in your application

Create a validator

The following validator check if the value is not 'foo' because foo is really a bad value. It expects a String and no configuration parameter. The `MinLength` validator uses a configuration parameter to define the minimum length. The validation error messages has the key `foo-not-allowed`. You might check the source code of the existing validators for further inspiration.

```
import org.apache.tapestry5.validator.AbstractValidator;
// ... some more imports here
public class FooValidator extends AbstractValidator<Void, String> {
        public FooValidator() {
                super(null, String.class, "foo-not-allowed");
        }

        public void validate(Field field, Void constraintValue, MessageFormatter formatter, String value)
                        throws ValidationException {
                if ("foo". equals(value))
                        throw new ValidationException(buildMessage(formatter, field, constraintValue));
        }

        private String buildMessage(MessageFormatter formatter, Field field, Void constraintValue) {
                return formatter.format(constraintValue, field.getLabel());
        }

        public void render(Field field, Void constraintValue, MessageFormatter formatter, MarkupWriter writer,
                                                                                FormSupport formSupport) {
                formSupport.addValidation(field, "foo", buildMessage(formatter, field, constraintValue), null);
        }
}
```

Create a resource file for validation messages

File ValidationMessages.properties

```
foo-not-allowed=Foo is not allowed
```

Register both in your application

A Tapestry application is configured by a AppModule class (See the Tapestry Guide). In the `AppModule` class add the following methods:

```
  public static void contributeValidationMessagesSource(OrderedConfiguration<String> configuration) {
    configuration.add("Default", "de/lalilуna/example/components/ValidationMessages", "before:*");
  }

  public static void contributeFieldValidatorSource(MappedConfiguration<String, Validator> configuration) {
    configuration.add("foo", new FooValidator());
  }
```

That's it.

Tapestry 4.0

this is a revised copy of a posing to the users mailing list by Scott F. Walter

1. implementing a validator

Create a class that extends `org.apache.tapestry.form.validator.BaseValidator`
(note, this is **not** `org.apache.tapestry.valid.BaseValidator`)

Override the `validate` method.

If you want a parameter to be passed into your validator:

1. create a constructor that takes a String
2. add a (non abstract) property (setXXX, getXXX) where XXX will be the name of your validator. E.g. if the validator is called

`mustBe`, use `getMustBe()`, `setMustBe(String)`. Later on you will reference the validator using: `mustBe=tapestry`

Here is an example:

```
package dummy;

import org.apache.tapestry.form.IFormComponent;
import org.apache.tapestry.form.ValidationMessages;
import org.apache.tapestry.form.validator.BaseValidator;
import org.apache.tapestry.valid.ValidatorException;

/**
 * @author scott/ron
 */
public class MustBeValidator extends BaseValidator {

   private String _mustBe;

   public MustBeValidator() {
     super();
   }

   public MustBeValidator(String string) {
     super(string);
   }

   public void setMustBe(String s) {
     _mustBe = s;
   }

   public String getMustBe() {
     return _mustBe;
   }

   public void validate(IFormComponent field,
             ValidationMessages messages,
             java.lang.Object object)
             throws ValidatorException {
             if (!equal(object,_mustBe) {
           throw new ValidatorException(messages.formatValidationMessage(getMessage(), "invalid-format", null));
       }
   }

   private boolean equal(Object o1, Object o2) {
     if (o1 == null && o2 == null)
       return true;
     if (o1 == null || o2 == null)
       return false;
     return o1.equals(o2);
   }
}
```

configuration

Provide a contribution to tapestry.form.validator.Validators in hivemodule.xml (stored in your WEB-INF directory)

{{{<?xml version="1.0"?>
<module id="myApplicationName" version="1.0.0">
<contribution configuration-id="tapestry.form.validator.Validators">
<validator name="mustBe" configurable="true" class="dummy.MustBeValidator"/>
</contribution>
</module>
}}}

Note: If your validator does not take any parameters set configurable to "false" here

Using

Inside the specification file, use the new validator as follows:

```
<component id="firstNameField" type="TextField">
  <binding name="value" value="ognl:person.firstName"/>
  <binding name="validators" value="required,minLength=3,mustBe=Scott"/>
  <binding name="displayName" value="First Name"/>
</component>
```