Tapestry5HowToAddValidators

Here is a short description how to add validators to Tapestry 5. In this example a validator is added to check whether the contents of a text field can be parsed as an integer. Changes and corrections welcome.

First define a validation class. This validator does not take an argument (as in min=5 e.g.), so the first template parameter is `Void`. As no conversion to a specific type of the contents is required, use `Object` as the second parameter.

The message-key is specified in the constructor, and is `validate-int`. The name of the validator function is `int` as specified in the 'addValidation' call in the `render` method.

A sample class is

```
public class ValidateInt extends AbstractValidator<Void, Object> {

	public ValidateInt() {
		super(null, Object.class, "validate-int");
	}

	@Override
	public void render(Field field, Void constraintValue,
			MessageFormatter formatter, MarkupWriter writer,
			FormSupport formSupport) {
		formSupport.addValidation(field, "int", formatter.format(field.getLabel()), null);
	}

	@Override
	public void validate(Field field, Void constraintValue,
			MessageFormatter formatter, Object value) throws ValidationException {
		if (value != null) {
			try {
				Integer.parseInt(value.toString());
			} catch (NumberFormatException ex) {
				throw new ValidationException(formatter.format(field.getLabel()));
			}
		}
	}

}
```

Specify the message in a properties file, e.g. `org/example/ValidationMessages.properties`

```
validate-int=You must provide a whole number for %s.
```

These need to be registered in `AppModule.java` for your application as follows:

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

	public void contributeValidationMessagesSource(
			OrderedConfiguration<String> configuration) {
		configuration.add("example","org/example/ValidationMessages");
	}
```

Not done yet, as the javascript to do client-side validation is still required. Create a javascript file `validators.js` with e.g.

```
Tapestry.Validator.int = function(field, message) {
	Tapestry.addValidator(field, true, function(value, event) {
		if (isNaN(Number(value)) || value.indexOf(".")!=-1)
			event.recordError(message);
	 });
}
```

This javascript file needs to be included in all pages/components where the new validator is to be used (Is there a better way?)

Once this is done you can add the validator to your text fields as in

```
<t:textfield t:id="select" value="selector" validate="required,int,min=0"/>
```