JSF 2 Exercises: Validation Part 2

1. Make a Website registration page where a user signs up by choosing a username and password. Have the user enter the password twice. Enforce that the password exists, the password is at least 6 characters long, and that the second password entry matches the first. This process is slightly easier if you use h:messages to put all the error messages at the top, but it is not too hard to put the error messages next to the fields: just give the form an id and then use "formId:fieldId" as the first argument to context.addMessage.

2. For the password, reject "password", "123456", "12345678", "qwerty", and "abc123". You could fold this test into the manual validation code from the first problem, but imagine that you might want to enforce this rule for more than one form, so instead use a custom validator method or component. Note that if you are putting error messages next to the fields, there is no need to specify the field ID in the validation method: it will automatically be associated with whichever field used "validator=" to specify the custom validation method.

3. Make a form where you nominate an employee for an award. The form should collect only a name (non-empty), and the confirmation should simply show the name. Make your app support at least one language in addition to English. You can take the language from the browser settings as in the Properties exercises, or you can add in controls to let the user choose the language, as in the lecture on handling GUI events, or you could even choose the Locale at random, as you did in one of your exercises on using Properties. Any of these approaches is fine; the point here is to be sure that the validation error message reflects the current language. If you don't speak any other languages, use translate.google.com or babelfish.com. You could also just use dummy values like "Name (FRENCH):".

4. Make a form where a user can sign up to be on a high-volume spam email list (yay!). Have the user enter an email address and two favorite URLs (so that the spammers can customize the emails based on the user's interests). Make sure the email addresses and URLs are in valid formats, and be sure that the two URLs are not the same. For checking that the URLs are different, use the MyFaces commons tag, not manual validation, and use "ne" (not equals) as the operator. Also, for the second URL you need to check *both* that it is a valid URL *and* that it is different than the first URL. So, you cannot use the validatorMessage attribute of h:inputText. Instead, use the message attribute of the MyFaces tags themselves, like this:

```
<mcv:validateBlah message="..."/>
```