Usage of Edit Masks

This chapter covers the following topics:

- General Information
- Data Types with Edit Masks
- Characters Used in Edit Masks
- Specifying Edit Masks in Layouts
- Static versus Dynamic Validation

General Information

Natural for Ajax supports a subset of the Natural edit mask concept, in order to support output formatting and input validation of numeric fields.

If an edit mask is specified for a numeric field, the field content is rendered according to the edit mask. Input to this field is validated against a regular expression that corresponds to this edit mask. The regular expression is automatically generated from the edit mask, so that by defining the edit mask, the user controls at design time the formatting of the field as well as the validation.

During the output formatting and input validation for numeric fields with edit masks, the Natural parameters `DC`, `THSEPCH` and `EMFM` are supported.

Data Types with Edit Masks

In all controls that support the property `datatype`, an edit mask can be specified for the following data types:

- `N` *`n.n`*
- `P` *`n.n`*
- `int`
- `float`
- `xs:short`
- `xs:decimal`

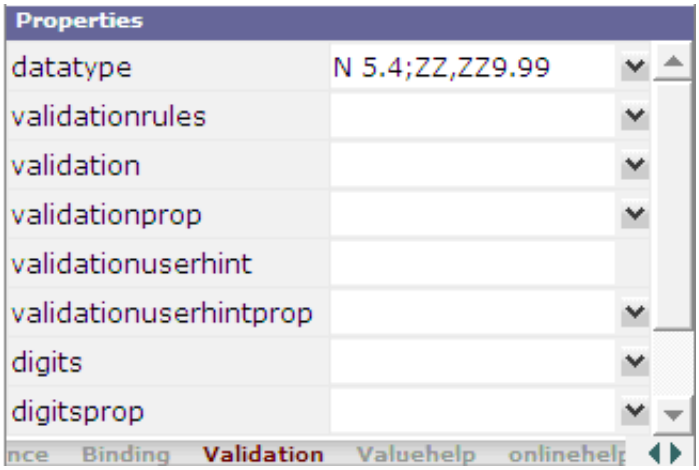
Characters Used in Edit Masks

The following characters can be used in edit masks:

Character	Function
9	Decimal digit or digit.
Z	Zero-suppressed digit.
+	Preceding/following sign.
-	Preceding/following sign if the value is negative.
,	Thousands separator character. **Note:** The actual character used as the thousands separator can be either a comma (,) or a period (.). This depends on the setting of the parameter THSEPCH in Natural.
.	Decimal separator character. **Note:** The actual character used as the decimal separator can be either a period (.) or a comma (,). This depends on the setting of the parameter DC in Natural.

Specifying Edit Masks in Layouts

An edit mask is added to a numeric data type in the following way (the edit mask is separated from the data type by a semicolon):



Application Designer generates a regular expression from the edit mask and adds it as a value for the `validation` property. This regular expression is used to validate the value on the client side when the user changes the field. This is done in the same way as if the user had specified the regular expression directly in the `validation` property.

The generated regular expression can be seen in the generation protocol. For the above example, it is shown as follows:

Id	Tag	Type	Message
20	itr		
21	label		name/N 5.4;ZZ,ZZ9.99 width/150
22	field		valueprop/thdecsepfield width/200 flush/server datatype/N 5.4;ZZ,ZZ9.99
		Info	Validation added [0-9,.]{1,6}[.,]{1}+[0-9]{1,2}

The validation takes place *after* the modification of a field value, that is: when the focus has left the field. This means that the validation does not take place with every key pressed. The user can enter invalid values as long as the input focus is in the field.

Input Rendering

For `ZZ,ZZ9.99` of the above example, the user can enter a valid value even without entering the thousands separator character:

Since the value itself is correct, it is accepted. Ideally, the value should be rendered according to the precise edit mask when the focus of the field is left. This can be easily achieved by setting the `flush` property to "server". This will trigger a server roundtrip and since output rendering takes place with each server roundtrip, the rendering is done automatically.

Input Completion

Application Designer does some input completion, for example, when working with `float` data types. An example for this is adding "0" decimal digits.

For the above input, Application Designer will do the following input completion:

Static versus Dynamic Validation

At design time, the characters that will actually be used for `DC` and `THSEPCH` are not yet known. Therefore, when the regular expression for validation is already generated at design time, both characters must be considered as possible `DC` or `THSEPCH` characters. In consequence, the validation is more tolerant than it should be.

To achieve a more stringent validation, the regular expression can also be dynamically generated at runtime. This is achieved by appending the literal "dynamic" to the edit mask, separated by a semicolon. In this case, the runtime settings of DC and THSEPCH are evaluated during the validation. Example:

```
datatype="N 5.4;+ZZ,999.99;dynamic"
```

If the parameter EMFM is to be evaluated, the validation expressions must be generated dynamically, because at design time, it is not yet known whether the application will run in free mode or not. Statically generated validations always behave as in free mode. That is: they do not force the user to enter plus or minus signs or the thousands separator character.