Rule Input Field

Rule Input Field

The rule input field is the most powerful and complex one of all the input fields offered by this panel. In its most simple incarnation it looks and works like a regular text input field. There is also only an incremental increase of the complexity in the specification for this case. However, it is unlikely that you would use it for such a purpose. The real power of this input field comes from the fact that rules can be applied to it that control many aspects of its look as well as overt and covert operation.

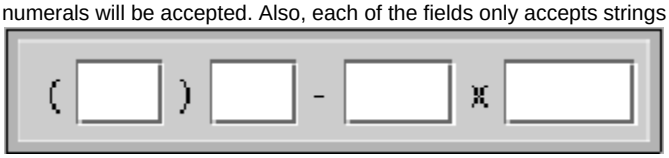
Layout and Input Rules

The basic nature of this input field is that of a text input field and as mentioned before, in its most simple incarnation that is what it looks like and how it operates. However, the layout of the field can be defined in such a way that there are multiple logically interconnected text input fields, adorned with multiple labels. Further more, each of these fields can be instructed to restrict the type of input that will be accepted. Now you might ask what this could be useful for. As an answer, let me present a few examples that show how this feature can be used. Before i do this however, i would like to describe the specification syntax, so that the examples can be presented together with the specifications that make them work in a meaningful way.

The actual specification of the layout, the labels and the type of input each field accepts all happens in a single string with the layout attribute. First let us have a look at the specification format for a single field. This format consists of a triplet of information, separated by two colons ':'. A typical field spec would look like this: N:4:4, where the first item is a key that specifies the type of input this particular field will accept - numeric input in the example. The second item is an integer number that specifies the physical width of the field, this is the same as in the with of any regular text field. Therefore the field in the example will provide space to display four characters. The third item specifies the editing length of the string or in other words, the maximum length of the string that will be accepted by the field. In the layout string you can list as may fields as you need, each with its own set of limitations. In addition you can add text at the front, the end and in between the fields. The various entities must be separated by white space. The behavior of this field is such that when the editing length of a field has been reached, the cursor automatically moves on to the next field. Also, when the backspace key is used to delete characters and the beginning of a field has been reached, the cursor automatically moves on to the previous field. So let us have a look a some examples.

Phone Number

The following specification will produce a pre formatted input field to accept a US phone number with in-house extension. Even though the pattern is formatted into number groups as customary, complete with parentheses '(' and dash '-', entering the number is as simple as typing all the digits. There is no need to advance using the tab key or to enter formatting characters. Because the fields only allow numeric entry, there is a much reduced chance for entering erroneous information. "(N:3:3) N:3:3 - N:4:4 x N:5:5". Each of the fields uses the 'N' key, indicating that only numerals will be accepted. Also, each of the fields only accepts strings of the same length as the physical width of the field.



E-Mail address

This specification creates a pattern that is useful for entering an e-mail address "AN:15:U @ AN:10:40 . A:4:4". Even though the first field is only fifteen characters wide it will accept a string of unlimited length, because the 'U' identifier is used for the edit length. The second field is a bit more restrictive by only accepting a string up to forty characters long.

IP address

It might not be uncommon to require entering of an IP address. The following simple specification will produce the necessary input field. All fields are the same, allowing just three digits of numerical entry. "N:3:3 . N:3:3 . N:3:3 . N:3:3"

Serial Number or Key Code

If you ship your product with a CD key code or serial number and require this information for registration, you might want to ask the customer to transcribe that number from the CD label, so that it is later on accessible to your application. As this is always an error prone operation, the predefined pattern with the easy editing support and restriction of accepted data helps to reduce transcription errors "H:4:4 - N:6:6 - N:3:3". This particular specification will produce three fields, the first accepting four hexadecimal, the second six numerical and the third three numerical digits.

Limitations

Even though the above examples all use single character labels between fields, there is no restriction on the length of these labels. In addition, it is possible to place label text in front of the first field and after the last field and the text can even contain spaces. The only limitation in this regard is the fact that all white space in the text will be reduced to a single space on the display. This means that it is not possible to use multiple spaces or tabs in the text.

The following table lists and describes all the keys that can be used in the specification string.

Key	Meaning	Description
N	numeric	The field will accept only numerals.
H	hexadecimal	The field will accept only hexadecimal numerals, that is all numbers from 0-F.
A	alphabetic	The field will accept only alphabetic characters. Numerals and punctuation marks will not be accepted.
AN	alpha-numeric	The field will accept alphabetic characters and numerals but no punctuation marks.
O	open	The filed will accept any input, without restriction.
U	unlimited	This key is only legal for specifying the editing length of a fields. If used, the field imposes no length restriction on the text entered.

Using Default Values

Like all other input fields the rule input field can also be pre-filled with data and as usual, this is accomplished thought the set attribute. As you might expect, the details of setting this field are rather on the complicated side. In fact you can set each sub field individually and you can leave some of the fields blank in the process. The set specification for all sub fields is given in a single string. Each field is addressed by its index number, with the count starting at 0. The index is followed by a colon ':' and then by the content of the field. The string "0:1234 1:af415 3:awer" would fill the first subfield with 1234, the second one withaf415 and the fourth with awer. The third subfield would stay blank and so would any additional fields that might follow.

The individual field specs must be separated with spaces. Spaces within the pre-fill values are not allowed, otherwise the result is undefined.

Output Format

The user input from all subfields is combined into one single value and used to replace the variable associated with the field. You can make a number of choices when it comes to the way how the subfield content is combined. This is done with the resultFormat and separator attributes. TheresultFormat attribute can take the following values:

Value	Meaning
plainString	The content of all subfields is simply concatenated into one long string.
displayFormat	The content of all subfields and all labels (as displayed) is concatenated into one long string.
specialSeparator	The content of all subfields is concatenated into one string, using the string specified with the separator attribute to separate the content of the subfields.
processed	The content is processed by Java code that you supply before replacing the variable. How to do this is described below.

Example

Here's some short example to illustrate this:

```
<field type="rule" variable="address">
  <description align="left" id="address.label"/>
  <spec id="url.address.label" txt="Connection:" layout="O:15:U : N:5:5"
resultFormat="displayFormat"/>
  <validator
class="com.izforge.izpack.panels.userinput.validator.RegularExpressionValidator"
id="address.fail">
    <param name="pattern"
value="\b.*\:(6553[0-5]|655[0-2]\d|65[0-4]\d{2}|6[0-4]\d{3}|[1-5]\d{4}|[1-9]\d{0,3})\b
"/>
  </validator>
</field>
```

or a bit more robust validation by checking whether the address is online:

```
<field type="rule" variable="address">
  <description align="left" id="address.label"/>
  <spec id="url.address.label" txt="Connection:" layout="O:15:U : N:5:5"
resultFormat="displayFormat"/>
  <validator
class="com.izforge.izpack.panels.userinput.validator.HostAddressValidator"
id="address.fail">
</field>
```

Note:

Coming with IzPack 5.0 (introduced in 5.0.0-RC2) it is not necessary to use the `<spec .. set="..."/>` attribute any longer. During creating the appropriate UserInputPanel with the rule field, there will be at first filled in the the values from the assigned variables, which are parsed and divided among the single edit field of a compound rule field. If the assigned variable isn't set, the value of the 'set' attribute is used like before. This fits better the behavior of other input fields on UserInputPanel, for example of `type="text"`and seems to be more natural (default values are only fallbycks of missing content).