A few words on white space control

General Description

The control of leading and trailing white spaces is always a problem regarding XML document processing. You always have to decide where you want to accept white spaces and how many.

For UJAC, I have decided to accept at most one leading/trailing white space. In the past I tried to automatically decide where to accept these single white spaces and where not. I have then found out that this decision heavily depends on the way you're using the tags. Now it's possible for the user to take more control over them.

I have added an attribute called *trim-body* to all font defining, all condition tags and the *phrase* tag. I've decided to support white space control for them, because these tags are the the most commonly used ones,

to manipulate the textual output of a surrounding higher level tag such as *cell*, *paragraph* or *phrase*.

The default behaviour for font tags is **not to trim** the leading or trailing white space, for all other tags the default is to trim them.

Here some examples:

```
<paragraph>
    Test
    <b>
        Document
    </b>
    <i>
        italic
    </i>
    <u>
        underline
    </u>
    .
</paragraph>
```

leads to

Test **Document** *italic* underline .

The reason for the big spaces between the words is the fact that the one accepted leading and trailing white spaces is not trimmed off. The next example solves this problem by specifying the attribute *trim-body* for the font manipulating tags.

```
<paragraph>
    Test
    <b trim-body="true">
        Document
    </b>
    <i trim-body="true">
        italic
    </i>
    <u trim-body="true">
        underline
    </u>
    .
</paragraph>
```

leads to

Test **Document** *italic* underline.

Explicite Whitespaces

As I mentioned before, UJAC only allowed one single leading or trailing white space. The same rule is applied for sequences of inner white spaces. These ones are also reduced to one single space. These rules are very simmilar to the HTML processing rules. In principal I'm content with these rules but what's about extra spaces, you may want to insert or line breaks? For this problem I have added support for some common escape sequences. You may have seen them already in programming languages like Java or C(++) or in Unix or Linux environments.

'\n': Adds a new line.
'\ ': Adds a explicite space.
'\t': Adds a tabulator, which is a short cut for four explicite spaces in a row.

Here some examples:

```
<paragraph>
    \tTest\n
    <b>Document</b>
</paragraph>
```

leads to

Test
Document

```
<paragraph>
    Numbers:
    <foreach loop-variable="i"
             sequence="1,2,3,4,5">
        \t${i}
        ->
        <switch value="">
            <case value="1">
                one
            </case>
            <case value="2">
                two
            </case>
            <case value="3">
                three
            </case>
            <case value="4">
                four
            </case>
            <case value="5">
                five
            </case>
        </switch>
        \n
    </foreach>
</paragraph>
```

leads to

Numbers:
1 -> one
2 -> two
3 -> three
4 -> four
5 -> five

Line Feeds

There are two ways to add line feeds in UJAC. One way is to add the special character
\n
Another way is to use the line feed tag:
**
**
The line feed special character is intended to work just in textual content. The tag is intended to work almost everywhere (document, phrase, paragraph, cell, column-text).

Usage Examples

Usage at <document> level:
Paragraph in document

content phrase after line feed tag (
)

Usage at <cell> level:

Paragraph in cell content phrase after line feed tag ( )	Plain text in cell content after two line feed tags ( )	Plain text in cell content after two line feed using the special character ('\n')

Usage at <column-text> level:

Paragraph in <column-text>

content phrase after line feed tag (
)

Plain text in <column-text>

Line Wrap Control

The 'no-wrap' attribute of paragraphs can be used to force the contents of a phrase to be displayed in one single line. Using this flag has one negative side effect: It removes all decorations, defined inside the phrase.

Usage Examples

Usage at <document> level:
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
+421 934 111 222

Usage at <cell> level:

'no-wrap' attribute not used aaaaaaaaaaaaaaaaaaaaaaaaaaaaa +421 **934** 111 222	'no-wrap' attribute used aaaaaaaaaaaaaaaaaaaaaaaaaaaaa +421 934 111 222
phone numbers: +421 111 111, +421 222 222, +421 333 333, +421 444 444, +421 555 555, +421 666 666,	phone numbers: +421 111 111, +421 222 222, +421 333 333, +421 444 444, +421 555 555, +421 666 666,

Usage at <column-text> level:

'no-wrap' attribute not used

aaaaaaaaaaaaaaaaaaaaaaaaaaaaa +421 **934** 111
222

'no-wrap' attribute used

aaaaaaaaaaaaaaaaaaaaaaaaaaaaa
+421 934 111 222

Dynamic evaluation

Usage with templates
Texts in UJAC print can be dynamically evaluated in two ways: First of all the template interpreter can be used for this. The second chance is to use conditional XML tags (<if>/<ellse>, <foreach> ...).

Template interpreter test:

The whitespace handling with template code is much more difficult than with static text.

Example with conditional tags:

The whitespace handling with template code is much more difficult than with static text.

Conditional table cell contents:

a (0) b (0) **c** (0) d (0) e (0) f (0) g (0) h (0) i (0) j (0) k (0) l (0) m (0) n (0) o (0) p (0) q (0) r (0) s (0) t (0) u (0) v (0) w (0) x (0) y (0) last z (0)	a (1) b (1) **c** (1) d (1) e (1) f (1) g (1) h (1) i (1) j (1) k (1) l (1) m (1) n (1) o (1) p (1) q (1) r (1) s (1) t (1) u (1) v (1) w (1) x (1) y (1) last z (1)	a (2) b (2) **c** (2) d (2) e (2) f (2) g (2) h (2) i (2) j (2) k (2) l (2) m (2) n (2) o (2) p (2) q (2) r (2) s (2) t (2) u (2) v (2) w (2) x (2) y (2) last z (2)	a (3) b (3) **c** (3) d (3) e (3) f (3) g (3) h (3) i (3) j (3) k (3) l (3) m (3) n (3) o (3) p (3) q (3) r (3) s (3) t (3) u (3) v (3) w (3) x (3) y (3) last z (3)
a (4) b (4) **c** (4) d (4) e (4) f (4) g (4) h (4) i (4) j (4) k (4) l (4) m (4) n (4) o (4) p (4) q (4) r (4) s (4) t (4) u (4) v (4) w (4) x (4) y (4) last z (4)	a (5) b (5) **c** (5) d (5) e (5) f (5) g (5) h (5) i (5) j (5) k (5) l (5) m (5) n (5) o (5) p (5) q (5) r (5) s (5) t (5) u (5) v (5) w (5) x (5) y (5) last z (5)	a (6) b (6) **c** (6) d (6) e (6) f (6) g (6) h (6) i (6) j (6) k (6) l (6) m (6) n (6) o (6) p (6) q (6) r (6) s (6) t (6) u (6) v (6) w (6) x (6) y (6) last z (6)	a (7) b (7) **c** (7) d (7) e (7) f (7) g (7) h (7) i (7) j (7) k (7) l (7) m (7) n (7) o (7) p (7) q (7) r (7) s (7) t (7) u (7) v (7) w (7) x (7) y (7) last z (7)