FirstSpirit™
Your Content Integration Platform

FirstSpirit™ Office

FirstSpirit Version 4.2

Version	**1.2**
Status	**RELEASED**
Date	**2009-07-14**
Department	FS-Core
Author/ Authors	A. Pfeiler, B. Gutknecht
Copyright	2009 e-Spirit AG
File name	OFFI42EN_FirstSpirit_Modules_Office

e-Spirit AG

Barcelonaweg 14
44269 Dortmund | Germany

T +49 231. 286 61-30
F +49 231. 286 61-59

info@e-spirit.de
www.e-spirit.de

e-Spirit^AG

Table of contents

1 Introduction

The FirstSpirit™ Office module enables text passages from Word documents or complete Word documents to be imported into FirstSpirit JavaClient and worked on further there. Not only pure text, but also formatting, paragraph breaks/carriage return, pictures, links and tables can be imported.

The module translates the information originating from the Word document into information which can be processed and displayed in FirstSpirit. Specifically, to this end Meta information from the Word document (e.g. formatting) is mapped with format and link templates in FirstSpirit. This is possible as the standard document exchange format for MS Office applications is HTML. The module uses individually definable rulesets, which contain details about which HTML information from the Word document is to be assigned to which format or link template in FirstSpirit.

1.1 Overview of the functions

The Office function can be used in the

- DOM Editor (CMS_INPUT_DOM) and
- DOM table (CMS_INPUT_DOMTABLE)

input components.

Unless specified otherwise, the whole of this document mainly relates to both input components. Alternatively, they are also jointly named "DOM input components" in the following.

The following information from Word documents is taken into account on importing into FirstSpirit:

- Continuous text
- Character formatting (**bold**, *italics*, underlined and combinations thereof)
- Paragraph formatting (e.g. headings)
- Paragraph breaks and "soft" line breaks (<RETURN> or <SHIFT> + <RETURN>)
- Lists (with number or symbols (bullets))
- Pictures (including automatic upload into Media Store)
- Links

- Tables (including merging, cell colour, cell alignment and cell format)

1.2 Topics covered in this document

Chapter 2: Describes installation of the "FirstSpirit Office" module on the server and installation of the project component in a project (from page 10).

Chapter 3: Explains the server and project-wide configuration options provided by the module (from Page 12).

Chapter 4: Uses examples to describe the syntax used for XML definition of the rulesets, with which content from Word documents can be reproduced in the DOM Editor (CMS_INPUT_DOM input component) or in the DOM Table (CMS_INPUT_DOMTABLE input component) (from Page 16).

Chapter 5: The DOM Editor and / or the DOM Table must be adjusted by the template developer so that the module can be used in JavaClient (from Page 42).

Chapter 6: This chapter is primarily intended for FirstSpirit editors and illustrates how the module can be used in JavaClient for editorial purposes. To this end, by way of example, the Word test document supplied is imported step by step and the effect in the DOM Editor and DOM Table as well as in the HTML output is shown. This chapter is also interesting for development of the XML rulesets (from Page 48).

1.3 Procedure for introducing the module

1.3.1 Use of the rulesets and templates provided

The default configuration of the Office module contains two rulesets which can be used immediately:

- “Project local Import Ruleset“: ruleset for converting content from Word documents with “static“ links (for projects working with static links, up to FirstSpirit version 4.2 inclusive)
- “Project local Import Ruleset (generic link)“: ruleset for converting content from Word documents with “generic“ links (for projects working with generic links, for further information see FirstSpirit Online Documentation, section “Template development” / “Link templates” / “Generic Link Editors”)
- “Standard (text only import)“: ruleset for content which is not to be converted

The rules defined in the global ruleset can be used to import the also available Word test document into the DOM Editor or DOM table with all its formatting, links and pictures.

In order to reproduce the formatting from Word, on the one hand the format templates included in the standard scope of supply of FirstSpirit are used, e.g. **bold** (format template abbreviation (quicktag) "b"), italic (format template abbreviation (quicktag) "i") or underline (format template abbreviation (quicktag) "u").

On the other hand, additional format templates required will be included with the following properties in a demo project. This is not yet (as of 06/2009) currently available.

- **H1-5**: Section format templates for level one to five headings:
  - Tag: “h1“ to “h5“
  - Style: „Bold“
  - Sizes: "20", “18“, “16“, “15“, “13“
  - Conversion: “Unicode to HTML4"
  - HTML channel: `<h1>$CMS_VALUE(#content)$</h1>`
- **Note**: Section format template for notes in red font colour:
  - Tag: “note “
  - Colour: “#FF0000“
  - Conversion: “Unicode to HTML4"
  - HTML channel: `<font color="red">$CMS_VALUE(#content)$ </font>`
- **Important**: Character format template for notes in red font colour:
  - Tag: “important “
  - Colour: “#FF0000“
  - Conversion: “Unicode to HTML4"
  - HTML channel: `<font color="red">$CMS_VALUE(#content)$ </font>`
- **s**: Character format template for strike through text:
  - Tag: “s “
  - Style: „Strikeout“
  - Conversion: “Unicode to HTML4"
  - HTML channel: `<s>$CMS_VALUE(#content)$</s>`

In addition, special **link templates** are required for importing links and pictures. Those required for importing the Word test document are located in the "Mithras Energy" demo project, under the "Link Template" node in the Template Store, under the "Continuous text link (external)" and "Continuous text link (internal)" link configurations:

- **textlinkinternal.standard**: Link template for reproducing imported pictures
- **textlinkexternal.standard**: Link template for reproducing external links

Note in case of using generic links: If these link templates have been converted to “generic links“ (context menu “Extras“ / “Convert link template“), their unique names are **textlinkinternal** and **textlinkexternal**. In this case the default ruleset “Project local Import Ruleset (generic link)“ must be selected when importing content into the DOM-Editor (see Chapter 6.2 page 50 and Figure 6-2).

The appearance of the format and link templates used in JavaClient and for output on the website can be adjusted to the needs of the respective project. For further information on the configuration options, see FirstSpirit Online Documentation, "Template Development", Chapter: "Format Templates" and "Link Templates".

For further information about the Word document for the test import, see Chapter 6.1 page 48.

1.3.2 Creating your own rulesets

Use of the following procedure is recommended to define your own rules or rulesets:

1. First, it is necessary to analyse the word documents to be imported and to create a list of the section (paragraph) and character formats used.
2. In a second step, it is necessary to check whether the section and character templates (paragraph and character style sheets) are used consistently in these Word documents. Depending on the individual configuration of Word and personal way of working, different style sheets (format templates) can be used, although the layout and function of the templates are the same. To achieve the best possible result for the Word import, the style sheets in Word should be consolidated first, because the fewer the number of different style sheets there are in a document, the easier it is to create the rulesets for the subsequent import.
3. A corresponding FirstSpirit section and character format template should be available for each paragraph and character format contained in the Word document. If they are not yet available, these templates should be created in FirstSpirit. The "Properties" tab is used to define how texts with this formatting are displayed in the DOM Editor; in the "HTML" tab (or tabs for other output

channels) is used to define how texts with this formatting are output on the website (or in other output media). If applicable, it is advisable to create a table as shown in Chapter 6.1 page 48.
In addition, an internal link template must be available for importing pictures; this template contains an input component for selecting a media object (`<CMS_PARAM name="mediaref" hidden="0"/>`). An external link template must be available for importing links.

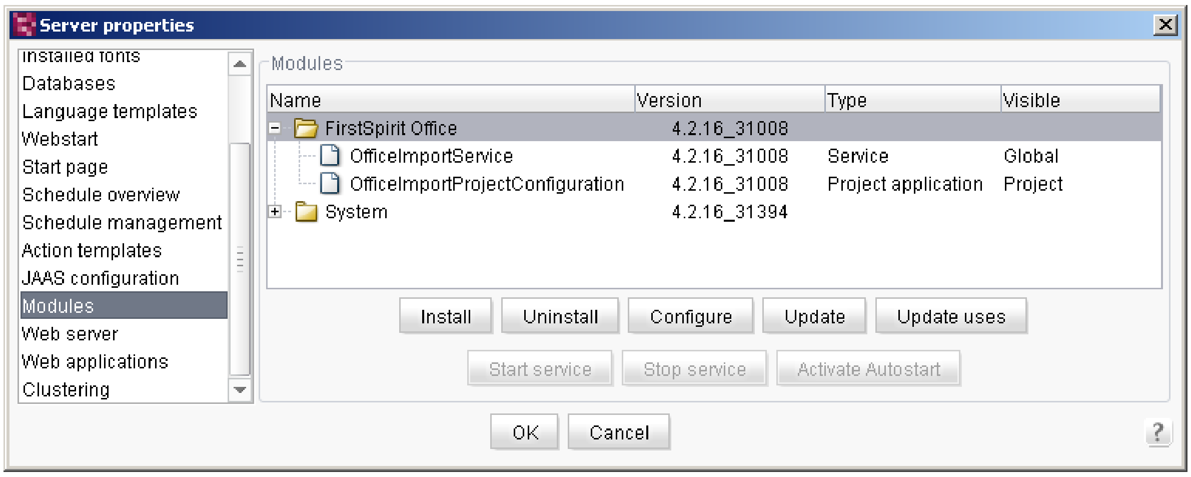
4. The HTML information which has to be translated for importing into FirstSpirit can be determined by exporting the Word document in HTML format. The XML rule definitions must then be created on the basis of this information (see Chapter 4 page 16, in particular Chapter 4.6 page 36).
5. The FirstSpirit input components to be used to import Word documents (DOM Editor, DOM Table) must then be configured for the Office function. To do this, among other things, the format templates created in the 4th step must be integrated in the input components in which the Office function is to be available (see Chapter 5.3 page 44).

The actual data import can then take place (see Chapter 6 page 48).

2 Installation

2.1 Installing the module on the FirstSpirit server

The "FirstSpirit Office" module is a licensable function. First, it must be installed in the FirstSpirit server and project configuration. To do this, the "Modules" menu entry is selected in Server Properties. Click the "Install" button to open a file selection dialog. The fsm file to be installed (file name "fs-office.fsm) can be selected here. The successfully installed module file is then displayed in the list of installed FirstSpirit modules with the name "FirstSpirit Office":



Name	Version	Type	Visible
FirstSpirit Office	4.2.16_31008		
OfficeImportService	4.2.16_31008	Service	Global
OfficeImportProjectConfiguration	4.2.16_31008	Project application	Project
System	4.2.16_31394		

Figure 2-1: Installation of the "FirstSpirit Office" module

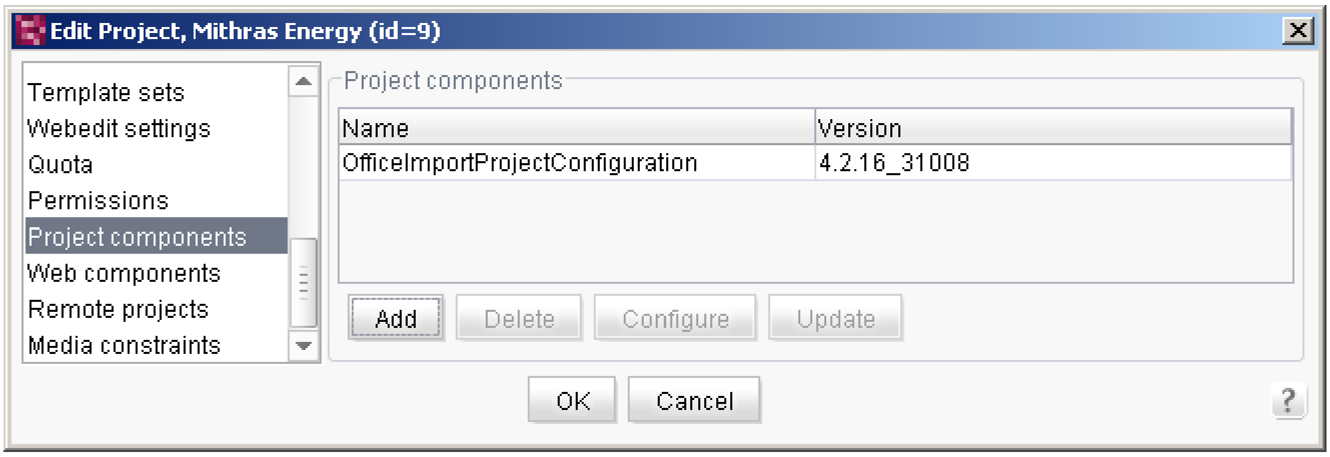
The module's components are the "OfficeImportService" service and the "OfficeImportProjectConfiguration" project application. For further information on this dialog, see *FirstSpirit Manual for Administrators*.

The "OfficeImportService" **service** enables the central definition of rules by which Word documents are imported. It is available globally, i.e. server-wide, in each project located on the FirstSpirit server. After starting the service, the Office function for the whole server can be configured (see Chapter 3.1 page 13).

The "OfficeImportProjectConfiguration" **project application** is a "local project" component. Following installation, it can be added to the required projects via their project properties (see Chapter 2.2 page 11) and the Office function can be configured for each specific project (see Chapter 3.2 page 14).

2.2 Installing the project component

If other or extending rules are to be used in a project, the "OfficeImportProjectConfiguration" project component must be installed in the required project. To do this, the "Project Components" menu entry within the project properties is opened.



Name	Version
OfficeImportProjectConfiguration	4.2.16_31008

Figure 2-2: Installing the project component

Add: Click the button to open the "Add" dialog. The list shows all project components installed on the server (see Chapter 2.1 page 10). Select the "OfficeImportProjectConfiguration" entry. The globally configured functions of the installed component are then available in the project and can be further configured (see Chapter 3.2 page 14).

If the project component is not added, the global rules only are available in the JavaClient.

For further information on this dialog, see *FirstSpirit Manual for Administrators*.

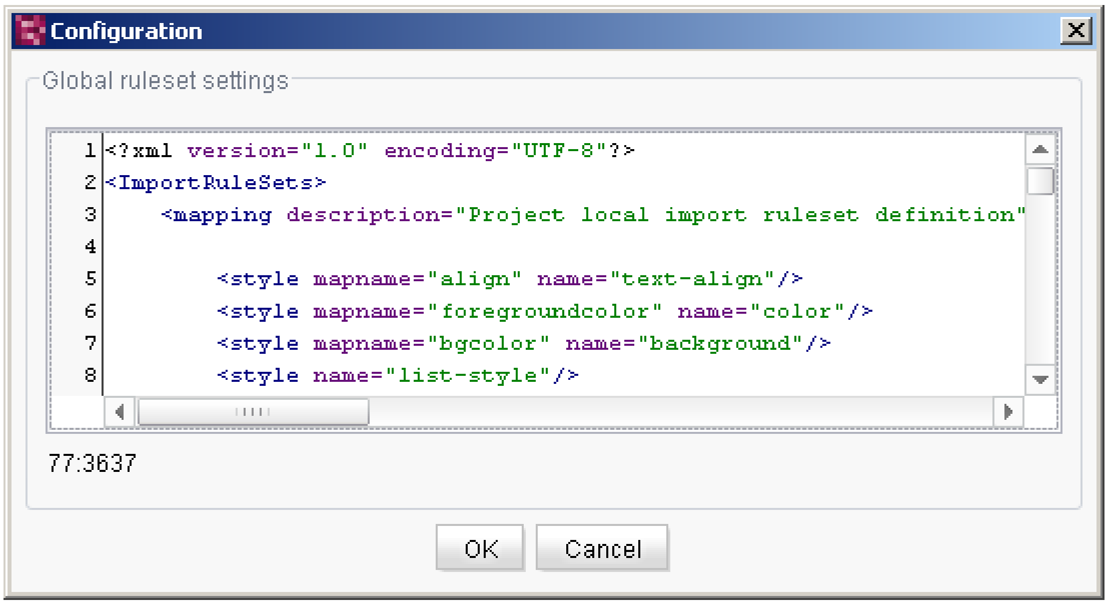
3 Configuring the Module

The following three options are available for configuring the module:

- To globally design one or several rulesets for **all** the FirstSpirit projects of a server, it is only necessary to configure the "OfficeImportService" accordingly (see Chapter 3.1 page 13).
- The "OfficeImportProject Configuration“ project component must be configured accordingly, in order to define one or several individual rulesets for **one** FirstSpirit project – independent of the global rulesets applicable server-wide. The "Extend the global ruleset configuration" option must be disabled (see Chapter 3.2 page 14).
- The "OfficeImportProject Configuration" project component must be configured accordingly, in order to use the global rulesets defined server-wide for **one** FirstSpirit project and to additionally extend it to include individual rulesets. For this, the "Extend the global ruleset configuration" option must be enabled (see Chapter 3.2 page 14).

3.1 Configuring the "OfficeImportService" service

Click the "OfficeImportService" service and the Configure button to open the service's configuration dialog:



```
<?xml version="1.0" encoding="UTF-8"?>
<ImportRuleSets>
    <mapping description="Project local import ruleset definition"

        <style mapname="align" name="text-align"/>
        <style mapname="foregroundcolor" name="color"/>
        <style mapname="bgcolor" name="background"/>
        <style name="list-style"/>
```

77:3637

Figure 3-1: Service configuration

The service must be started first, before it can be configured. If it is not, the "Start service?" message appears as a prompt. Click "Yes" to start the service and the configuration dialog appears (see Figure 3-1).

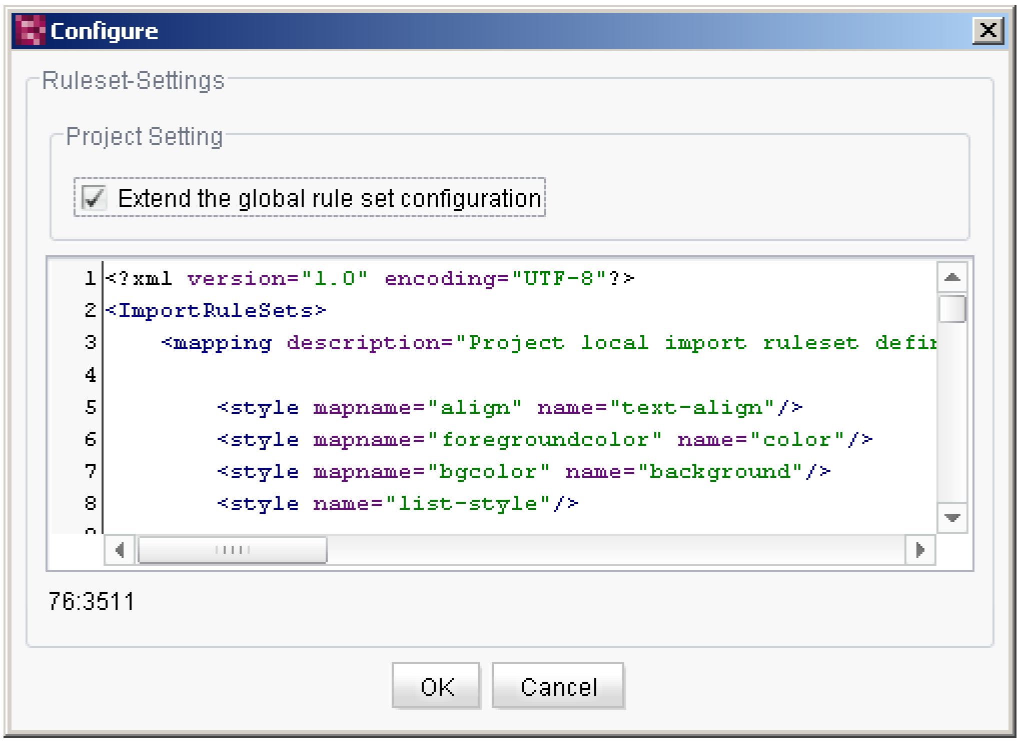
This text field can be used to enter global rules for the import of texts from Word documents, which apply to the whole server. These rules define how the formatting from the Word document subsequently has to be converted into FirstSpirit expressions. For detailed information on XML definition of the rulesets, see Chapter 4 page 16.

Different rulesets can be defined for different purposes or different document types. For example, one ruleset can be defined for technical documents and one for marketing documents. The name ("name" attribute) and description ("description" attribute) of the rulesets are subsequently displayed to the editor while working with the module in JavaClient and are offered for selection.

3.2 Configuring the "OfficeImportProject Configuration" project component

The service must be started first before the "OfficeImportProject Configuration" project component can be configured (see Chapter 2.1 page 10).

Click the "OfficeImportProjectConfiguration" project component and the Configure button to open the project component's configuration dialog:



☑ Extend the global rule set configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<ImportRuleSets>
    <mapping description="Project local import ruleset defi

        <style mapname="align" name="text-align"/>
        <style mapname="foregroundcolor" name="color"/>
        <style mapname="bgcolor" name="background"/>
        <style name="list-style"/>
```

76:3511

Figure 3-2: Configuring the project component

This dialog can be used to configure special import rules for the respective project. Here too, the text field can be used to define various rulesets for different purposes (see Chapter 4 page 16).

Extend the global ruleset configuration: This option can be used to define whether the server-wide configured rules (see Chapter 3.1 page 13) are to be replaced by the rules given here or whether project-specific rulesets are to be added. If the option is enabled, in addition to the rules defined for server-wide application, the project-specific rules defined here are also subsequently offered to the editor for selection while they are working with the module in JavaClient. If the option is disabled, the server-wide defined rules are not available to the editor; they can only use the project-specific rules.

4 XML Definition of the Rulesets

If content from Word documents is copied to the clipboard using <CTRL> + <C>, it is mainly stored in pure HTML format. In addition, the clipboard also contains extensive Word-specific information, which cannot be used for the import into FirstSpirit. This HTML information can, e.g. be displayed by saving the Word document concerned as an html file and then opening it in a browser using the "Show Source Text" option.

The rulesets used in FirstSpirit Office are there for linking the HTML information from the clipboard (e.g. <p> for paragraphs, <b> for bold, <h1> for level one headings, <td> for table cells, <img src=...> for pictures, <a href=...> for links, etc.) with the character and section format templates and the link templates (for pictures and links) in FirstSpirit. As a result, for example, "bold" character formatting from Word is "translated" into the "bold" character formatting of the FirstSpirit DOM Editor. The result of this is, on the one hand, that bold text from a Word document is also formatted and displayed as bold text in the DOM Editor, as defined on the "Properties" tab of the relevant format template. The FirstSpirit format template also defines how the "bold" character formatting is to be output (e.g. in the HTML channel, "HTML" tab of the format template concerned).

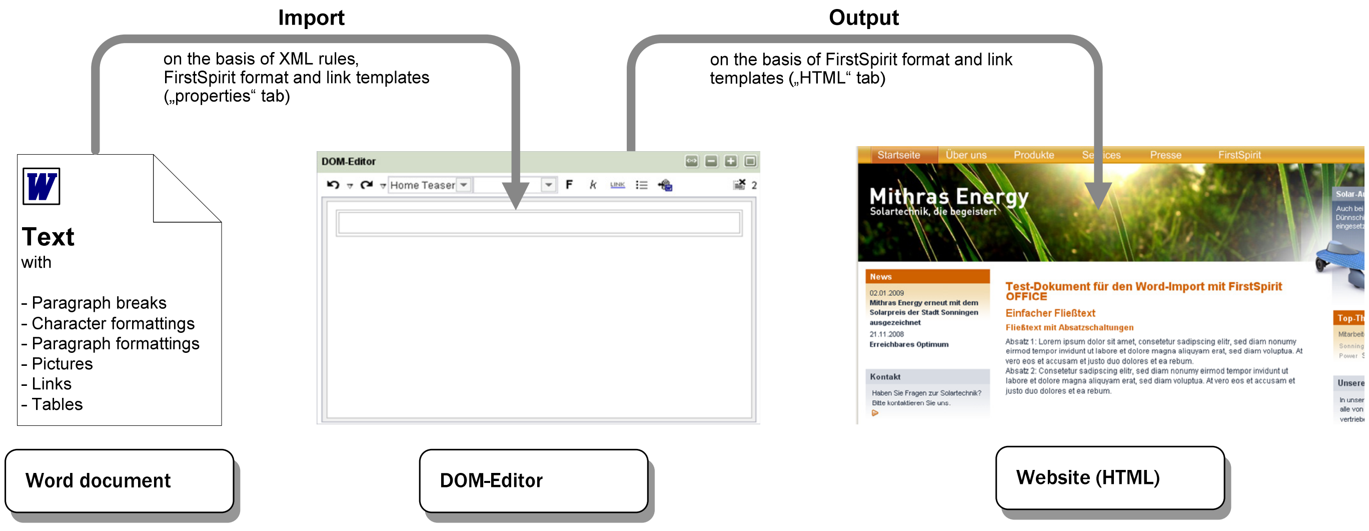

Import
on the basis of XML rules, FirstSpirit format and link templates („properties“ tab)
Output
on the basis of FirstSpirit format and link templates („HTML“ tab)
Text
with
- Paragraph breaks
- Character formattings
- Paragraph formattings
- Pictures
- Links
- Tables
DOM-Editor
Home Teaser
Mithras Energy
Solartechnik, die begeistert
Test-Dokument für den Word-Import mit FirstSpirit OFFICE
Einfacher Fließtext
Word document
DOM-Editor
Website (HTML)

Figure 4-1: Schema import – Output

The rulesets in FirstSpirit Office are configured in XML format.

4.1 XML syntax

XML documents must be well-formed and valid, so that they can be read out and interpreted by a parser. The following rules must be followed:

- **XML declaration**: An XML declaration must be given at the start of the XML definition. Example:

```
<?xml version="1.0" encoding="UTF-8"?>
```

- **Structure**: For valid, well-formed XML, all opened tags must also be closed again.
- **Comments**: Comments can be defined with `<!--... -->`. Everything contained within the angle brackets is ignored in the import.

When non well-formed XML is saved, a corresponding error dialog is displayed and the document cannot be saved.

4.2 Structure, nesting and inheritance

The basic structure of the XML rule definition consists of

- the XML declaration (see Chapter 4.1 page 17) and
- the root element (see Chapter 4.3.1 page 20).

It is followed by a `mapping` element for each ruleset (see Chapter 4.3.2 page 20). Within this element, an XML tag must be given for each piece of information with special formatting originating from the Word document (see Chapter 4.3.3 to 4.3.6 from page 23). This can be:

▪ `style`	(see Chapter 4.3.3 page 23)
▪ `element`	(see Chapter 4.3.4 page 24)
▪ `attribute`	(see Chapter 4.3.5 page 26)
▪ `text`	(see Chapter 4.3.6 page 27)

All elements (apart from the root element) must be assigned attributes, namely the following:

for the `mapping` tag: (see Chapter 4.3.2 page 20).

- `description`
- `linkConfigExternal`
- `linkConfigInternal`
- `mimeType`
- `name`
- `versionTag`

for the `style` tag: (see Chapter 4.3.3 page 23).

- `mapname`
- `name`

for the `element` tag:

▪ `handler`	(see Chapter 4.4.3 page 29)
▪ `maptag`	(see Chapter 4.4.2 page 29)
▪ `tag`	(see Chapter 4.4.1 page 28)
▪ `default`	(see Chapter 4.4.4 page 31)
▪ `id`	(see Chapter 4.4.5 page 32)
▪ `findtag`	(see Chapter 4.4.6 page 33)
▪ `content`	(see Chapter 4.4.7 page 33)
▪ `mediaref`	(see Chapter 4.4.8 page 33)
▪ `inherit`	(see Chapter 4.4.9 page 33)
▪ `break`	(see Chapter 4.4.10 page 34)
▪ `class`	(see Chapter 4.4.11 page 35)
▪ `mapattributes`	(see Chapter 4.4.11 page 35)

for the `attribute` tag:

▪ `mapname`	(see Chapter 4.3.5 page 26)
▪ `name`	(see Chapter 4.3.5 page 26)
▪ `handler`	(see Chapter 4.4.3 page 29)

for the `text` tag: (see Chapter 4.3.6 page 27).

- `action`

The examples given in the following are based on the rulesets supplied in the module's default configuration.

The fundamental assignment of HTML information from the Word document to format and link templates in FirstSpirit is defined by `element` tags (see Chapter 4.3.4 page 24). The `element` tags in turn contain tags (`attribute`, `text` or other `element` tags), which define the conversion rules for sub-elements. They can be nested to any depth (lower level) required. However, for improved clarity, the nesting depth should be kept small and the XML code should be kept as "lean" as possible.

Sub-`element` tags inherit die attributes of the higher-level `element` tags as well as of the `attribute` and `text` tags. This inheritance can be inhibited by the `inherit="NONE"` attribute (see Chapter 4.4.9 page 33). An `element` tag can also inherit the attributes of another `element` tag which is not on a higher-level. To do this, a name must be given to the `element` tag from which the attributes are to be adopted using `id="..."`. The `element` tag which is to inherit the attributes can access the attributes under this name via `inherit="ID_DES_VERERBENDEN_ELEMENTS"`. This is relevant, e.g. for table cells, whose contents are to be converted in the same way as continuous text content.

Example:

```
<element default="true" handler="map" id="HTML.paragraph" tag="p">
...
<element handler="table" tag="table">
   <element tag="tr">
       <attribute name="colspan"/>
       <attribute name="rowspan"/>
       <attribute handler="style" name="style"/>

          <element inherit="HTML.paragraph" tag="td">

             <element handler="strip" inherit="HTML.paragraph"
                      mapattributes="true" tag="p"/>
             </element>

          </element>

   </element>
</element>
```

In this example, the attributes of the first line of the HTML tag `p` of a continuous text (body) paragraph are also used for table cells (HTML tag `td`) through the `id` "HTML.paragraph" and paragraphs within a table cell (HTML tag `p` within the table, which is opened with `tag="table"`).

The rough structure of the XML ruleset is based on the structure of the HTML file from the Word clipboard and therefore includes, among other things, the following elements:

- Head
- Body
- Paragraphs (`<p>`)
    - Pictures
    - Breaks (`<br>`)
    - Character and paragraph formatting
    - Links
    - Lists
    - Tables
        - Table rows
        - Table cells
        - Headers (`<th>`)

This is also the schema of the ruleset example supplied with the module. The indenting below the rules for paragraphs means that all indented elements (pictures, tables, etc.) inherit the rules for paragraphs.

4.3 Tags

4.3.1 ImportRuleSets

The XML definition must be enclosed by the root element

```
<ImportRuleSets>... </ImportRuleSets>
```

The `ImportRuleSet` tag does not require any attributes.

4.3.2 mapping

The root element can contain several rulesets. Each ruleset is opened with a mapping element

```
<mapping...>
```

Various attributes can be given for each mapping element:

Attribute	Explanation / Example	Mandatory attribute
`description`	Gives a description for the ruleset; this is displayed to the editor in the DOM input component when they work with the Office module (see Figure 6-2), e.g. `description="Project local import ruleset definition"`	yes
`linkConfigExternal`	Gives the unique name of the link template, which is to be used for the definition of external links when importing Word documents. This link template must also be integrated in the respective DOM input component via the *LINKEDITOR* tag, e.g. `linkConfigExternal="textlinkexternal.standard"` This attribute does not have to be given if links are not to be converted in the import from Word documents.	no
`linkConfigInternal`	Gives the unique name of the link template, which is to be used for the definition of media when importing Word documents. This link template must also be integrated in the respective DOM input component via the *LINKEDITOR* tag, e.g. `linkConfigInternal="textlinkinternal.standard"` This attribute does not have to be given if media is not to be imported.	no

Attribute	Explanation / Example	Mandatory attribute
`mimeType`	Details of the MIME type, e.g. `mimeType="text/html"` The "Standard (text only import)" ruleset supplied with the module is intended to be used to import texts, which are not to be converted when imported in the installed Office module. It therefore has the MIME type `mimeType="text/plain"`.	yes
`name`	Gives a unique name for the ruleset; this is displayed to the editor in the DOM Editor when they work with the Office module (see Figure 6-2), e.g. `name="Project local Import Ruleset"`	yes
`versionTag`	This attribute gives the version number of the XML ruleset. It is automatically increased by one each time a change is made, e.g. `versionTag="24"` With each import, this version number is used to initially check whether changes have been made to the rulesets since the last import. If not, the rulesets still in the client cache are used directly. If changes have been made in the meantime, the modified rulesets are loaded first and are then applied.	automatic

The mapping element must be closed again at the end of the XML definition.

Example of several rulesets:

```
<?xml version="1.0" encoding="UTF-8"?>
  <ImportRuleSets>
    <mapping description="Project specific ruleset"
linkConfigExternal="textlinkexternal.standard"
linkConfigInternal="textlinkinternal.standard"
```

```
mimeType="text/html" name="Project" versionTag="24">
      ...
    </mapping>

    <mapping description="Project specific ruleset for
plain text"
linkConfigExternal="textlinkexternal.standard"
linkConfigInternal="textlinkinternal.standard"
mimeType="text/html" name="Text" versionTag="24">
      ...
    </mapping>
  </ImportRuleSets>
```

After the root and mapping element, an assignment rule by which the information is to be reproduced in FirstSpirit is defined for each piece of information copied onto the clipboard from a Word document.

4.3.3 style

The `style` tag is used to define the conversion of "style" tags from the HTML of the Word document. The conversion rules defined with the help of the `style` tag apply globally to the whole Word document.

The conversion of "style" tags from the Word document into information which can be evaluated in FirstSpirit is established within a `style` tag by the `mapname` and `name` attributes: `name` gives the values of the "style" tags from the Word document which are to be converted when imported into FirstSpirit; `mapname` is used to give the FirstSpirit value into which the HTML attribute is to be translated. Both attributes must exist in a `style` tag, e.g.

```
<style mapname="align" name="text-align"/>
```

In this example, information from the Word document, which is marked up by the "style" tag `text-align`, is linked with the FirstSpirit `align` attribute on being imported into a DOM input component. With this, on being imported into a DOM input component, text alignments in Word, e.g. `right` (flush-right/right-aligned) or `center` (centred) are reproduced in `align`, which is used to control text alignment in FirstSpirit.

A `style` tag must be defined in the XML ruleset for each "style" tag from the Word document to be evaluated in FirstSpirit.

If `mapname` is not explicitly given, the name given by the `name` attribute is used. Example:

```
<style name="list-style"/>
```

In this example, the "list-style" value defined for `name` is used as the `mapname`. This is therefore an alternative, shortened notation for

```
<style mapname="list-style" name="list-style"/>
```

Attribute	Explanation / Example	Mandatory attribute
`name`	Details of "style" tags from the Word clipboard, e.g. `name="text-align"`	yes
`mapname`	Gives a value in FirstSpirit, into which the `name` attribute is to be translated, e.g. `mapname="align"` If the values of the "style" tag from the Word document and the corresponding value in FirstSpirit are identical, it is not necessary to give `mapname`.	no

No other attributes have to be given for the `style` tag.

4.3.4 element

The `element` tag is used to define the conversion rules for the HTML elements of the Word document, e.g. elements for

- the HTML framework: `html`, `head`, `body`
- Text formatting: `p`, `div`, `span`, `br`, `b`, `strong`, `I`, `h1`, `li`, `ol`, `ul`
- the definition of links: `a`
- the definition of tables: `table`
- the integration of graphic pictures: `img`

An `element` tag **must** be given in the XML definition for each HTML element contained in the Word document. If an `element` tag is not given, the default element is used (see Chapter 4.4.4 page 31).

The following **mandatory attributes** must be given for the `element` tag:

- `tag / maptag`: These attributes are used to establish the link between the HTML information from the Word document and the FirstSpirit format and link templates: `tag` is used to give the values of the "style" tags from the Word document which are to be converted on being imported into FirstSpirit; `maptag` gives the FirstSpirit value (abbreviation of the FirstSpirit format template or other values which are used internally to control FirstSpirit), into which the HTML element is to be translated (see also Chapter 4.4.1 page 28 and Chapter 4.4.2 page 29). `maptag` does not need to be given if the same name is used as that given for the `tag` attribute.
- `handler`: This attribute defines how the HTML information from the Word document is to be processed (handled). It is also required to import pictures, links and tables (see Chapter 4.4.3 page 29).

The following **optional attributes** can be used with `element`:

▪ `default`:	(also called "default element") Definition of conversion rules for unknown information from Word documents (see Chapter 4.4.4 page 31)
▪ `id`	Definition of names for `element` tags, in order to be able to reference to them elsewhere (see Chapter 4.4.5 page 32)
▪ `findtag`	Read out tags from the Word document (in conjunction with `handler="find"`), e.g. the title of the Word document (see Chapter 4.4.6 page 33)
▪ `content` (in	Process / Parse objects from Word documents, e.g. pictures conjunction with `handler="media"`) (see Chapter 4.4.7 page 33)
▪ `mediaref`	Import pictures (in conjunction with `handler="media"`) (see Chapter 4.4.8 page 33)
▪ `inherit`	Control the inheritance of attributes (see Chapter 4.4.9 page 33)
▪ `break`	Convert text breaks, e.g. in table cells (see Chapter 4.4.10 page 34)
▪ `mapattributes`	Use attributes from the Word document, e.g. Attributes in tables (see Chapter 4.4.11 page 35)
▪ `class`	Convert self-defined Word style sheets, which are given the

"class" attribute in the HTML clipboard
(see Chapter 4.4.12 page 35)

Example:

```
<element handler="object" maptag="link" tag="a">
```

This `element` tag contains the assignment for links: these are marked up with the `a` tag in the HTML of the Word document. `maptag="link"` must be used in the ruleset to give links. Internally, `link` is reproduced on `CMS_LINK`.

4.3.5 attribute

The `attribute` tag is used to define the conversion rules for the HTML attributes of the Word document, e.g. for

- HTML style sheet definitions: `style`
- the definition of links: `href`
- the definition of tables: `colspan`, `rowspan`, `style`
- the integration of graphic pictures: `src`

An `attribute` tag **must** be given in the XML definition for each HTML style sheet attribute contained in the Word document. It must be enclosed by `<element>... </element>` and therefore inherits all the attributes of the higher level element (see Chapter 4.2 page 17).

The following **mandatory attributes** must be given for the `attribute` tag:

- `name` / `mapname`: These attributes are used to establish the link between the HTML information from the Word document and the FirstSpirit format and link templates: `name` is used to give the values of the HTML attributes from the Word document which are to be converted on being imported into FirstSpirit; `mapname` gives the FirstSpirit value (abbreviation of the FirstSpirit format template or other values which are used internally to control FirstSpirit), into which the HTML attribute is to be translated. `mapname` does not need to be given if the same name is used as that given for the `name` attribute.
- `handler`: This attribute defines how the HTML information from the Word document is to be processed (handled) (see also Chapter 4.4.3 page 29). In addition, `handler="style"` can also be used to import the information from the `style` tag (see Chapter 4.3.3 page 23).

Example: `<attribute handler="style" name="style"/>`

Example for **Links**:

```
<element handler="object" maptag="link" tag="a">
         <attribute mapname="target" name="href"/>
</element>
```

This `attribute` tag contains the assignment for the `href` attribute, with which a link's URL is given. `mapname="target"` must be given in the ruleset.

For further information on importing links, see Chapter 6.5 page 57.

Example for **pictures**:

```
<element content="IGNORE" handler="media" mediaref="src"
                      tag="img">
         <attribute name="src"/>
</element>
```

This `attribute` tag is used to import the corresponding picture into the Media Store. `mapname="src"` must be given in the ruleset; however, it can also be omitted, as the value of `mapname` and `name` is identical.

For further information on importing pictures, see Chapter 6.5 page 57.

4.3.6 text

The `text` tag can be used to define how texts within elements are to be processed on being imported into the DOM input component.

The `text` tag must be enclosed by `<element>... </element>` and therefore inherits all the attributes of the higher level element (see Chapter 4.2 page 17).

The following **mandatory attribute** must be given for the `text` tag:

- `action`: This attribute is used to define how text is to be processed on being imported into a DOM input component.

The following values can be used for `action`:

Attribute	Explanation / Example	Mandatory attribute
`ignore`	Text is not imported into the input component, i.e. it is ignored, e.g. `<text action="ignore"/>`	no
`default`	If a `default` element is defined (see Chapter 4.4.4 page 31), the text is converted with the attributes and values defined by this element, e.g. `<text action="default"/>` Line breaks (`<br>`) and spaces (`&nbsp;`) are removed and are not imported into the DOM input component.	no
`keep`	The text is imported into the DOM Editor, e.g. `<text action="keep"/>` This is the default setting.	Default

4.4 Attributes

4.4.1 tag

The `tag` attribute is used to give the name of the HTML element in the `element` tag, e.g.

```
<element... tag="ol"/>
```

for numbered lists.

For notes about the correct output of nested lists see also Chapter 5.4 page 45.

4.4.2 maptag

The `maptag` attribute is in `element` tags to give the abbreviation of the FirstSpirit format template with which the respective Word template is to be mapped, e.g.

```
<element... maptag="h1".../>
```

for level one headings in the Word document.

If this attribute is not given, the same name is automatically used as that given for the `tag` attribute (see Chapter 4.4.1 page 28). If the format names in FirstSpirit and in the Word document are identical, the `maptag` attribute does not have to be given too.

4.4.3 handler

The `handler` attribute can be used to give different options, defining how the respective HTML element is to be handled on being imported into the DOM Editor. The **map** value is used as a default. In this case, the HTML Element is not converted from the Word document. It can be mapped with this tag name by giving the additional attribute `mapname`. If the `handler` attribute is not defined for an HTML element, `handler="map"` is automatically set.

The **skip** value can be used to skip the HTML element, i.e. it is ignored; **strip** removes the tag of the respective element on being imported into the DOM Editor (e.g. the `<html>` tag), but the content of the element is retained. If the `mapattributes="true"` attribute is also given, the attributes of the element are applied to the higher level element (see Chapter 4.4.11 page 35).

The **default** value can be used to revert to the default conversion rules (see Chapter 4.4.4 page 31 and Chapter 4.5 page 36).

The values **object**, **media** and **table** are required to import links, pictures and tables. As with the **map** value, the respective HTML element is not converted. The following tags and attributes are also required to import these objects:

- for links: `attribute` tag for `href` (see Chapter 4.3.5 page 26)

- for pictures: `content` (see Chapter 4.4.7 page 33), `mediaref` (see Chapter 4.4.8 page 33)
- for tables: `element` and `attribute` tags for the conversion of
  - **Table rows** (Word HTML tag `tr`): e.g. `<element tag="tr">`
  - **Table cells** (Word HTML tag `td`): e.g. `<element inherit="HTML.paragraph" tag="td">`
  - **header cells** (Word HTML tag `th`): e.g. `<element inherit="HTML.tablecell" maptag="td" tag="th"/>`
  - **Merges** (Word HTML tags `colspan` and `rowspan`): e.g. `<attribute name="colspan"/>` and `<attribute name="rowspan"/>`

Value	Explanation / Example	Mandatory information
`map`	the HTML element is not converted on import, e.g. `<element default="true" handler="map" id="HTML.paragraph" tag="p">` This is the default setting.	Default
`skip`	the HTML element is skipped on import, e.g. `<element handler="skip" tag="*"/>`	no
`strip`	the tag of the HTML element is removed and the content of the HTML element is imported into the DOM input component, e.g. `<element handler="strip" tag="html">` If `mapattributes="true"` is also given, the attributes of this element are transferred to the higher-level element.	no
`default`	Revert to the default element, e.g. `<element handler="default" tag="*">`	no

Value	Explanation / Example	Mandatory information
`object`	required to import links, e.g. `<element handler="object" maptag="link" tag="a">`	no
`media`	required to import media, e.g. `<element content="IGNORE" handler="media" mediaref="src" tag="img">`	no
`table`	required to import tables, e.g. `<element handler="table" tag="table">`	no
`find`	Read out tags from the Word document, e.g. `<element findtag="title" handler="find" tag="head"/>`	no

4.4.4 default

The `default` attribute can be used to define an element as a default conversion rule for unknown formatting ("default element"). If the value "true" is set for `default`, the conversion is used for all formatting or objects from the Word document for which there is no ruleset, e.g.

```
<element tag="p" handler="map" default="true" id="HTML.paragraph">
```

This rule can then be reverted to with `<element tag="*" handler="default">`. With this definition, all "unknown" HTML elements from the Word document are converted using the default conversion rule (see Chapter 4.5 page 36).

In the example given, unknown formatting and mark-ups in the Word document are transferred in a paragraph (`<p>... </p>`).

Only one default element should be defined for each ruleset. If several default elements are defined, the last element of the XML ruleset is always used.

Value	Explanation / Example	Mandatory information
`true`	Configuration of the `element` tag as the default conversion rule for unknown HTML formatting from the Word document, e.g. `<element default="true" handler="map" id="HTML.paragraph" tag="p">`	no

4.4.5 id

The `id` attribute can be used to give `element` tags a name. Other `element` tags reference to this name via the `inherit` attribute (see Chapter 4.4.9 page 33) and therefore import all attributes and values, e.g.

```
<element default="true" handler="map" id="HTML.paragraph" tag="p">
...
<element handler="table" tag="table">
   <element tag="tr">
       <attribute name="colspan"/>
       <attribute name="rowspan"/>
       <attribute handler="style" name="style"/>

          <element inherit="HTML.paragraph" tag="td">

             <element handler="strip" inherit="HTML.paragraph"
                      mapattributes="true" tag="p"/>
             </element>

          </element>

   </element>
</element>
```

In this example, the element with the attribute `tag="p"` (continuous text paragraphs) is given the name "HTML.paragraph". Table cells (HTML tag `td`) and paragraphs within a table cell (HTML tag `p` within the table, `tag="table"`) reference to this element via the `id` "HTML.paragraph" and therefore use values and attributes from continuous text paragraphs (see also Chapter 4.2 page 17).

4.4.6 findtag

The `findtag` attribute together with the `handler"find"` attribute (see Chapter 4.4.3 page 29) can be used to read out tags from the Word document, e.g.

```
<element findtag="title" handler="find" tag="head"/>
```

In this example, the text of the `title` element is read out of the `head` section of the Word document.

4.4.7 content

The `content` attribute is required to import pictures, e.g.

```
<element content="IGNORE" handler="media" mediaref="src"
tag="img">
```

The `IGNORE` value ensures that the picture is not parsed.

4.4.8 mediaref

The `mediaref` attribute is required to import pictures, e.g.

```
<element content="IGNORE" handler="media" mediaref="src"
tag="img">
```

`src` must always be given as the value for `mediaref`.

4.4.9 inherit

The `inherit` attribute can be used by an `element` tag to inherit the values and attributes of another `element` tag, e.g.

```
<element default="true" handler="map" id="HTML.paragraph" tag="p">
...
<element handler="table" tag="table">
   <element tag="tr">
       <attribute name="colspan"/>
       <attribute name="rowspan"/>
       <attribute handler="style" name="style"/>

          <element inherit="HTML.paragraph" tag="td">
```

```
              <element handler="strip" inherit="HTML.paragraph"
                       mapattributes="true" tag="p"/>
              </element>

           </element>

   </element>
</element>
```

In this example, the element with the attribute `tag="p"` (continuous text paragraphs) is given the name "HTML.paragraph". Table cells (HTML tag `td`) and paragraphs within a table cell (HTML tag `p` within the table, `tag="table"`) reference to this element via the `id` "HTML.paragraph" and therefore use values and attributes from continuous text paragraphs (see also Chapter 4.2 page 17).

The value `NONE` can be used to inhibit the respective `element` tag from inheriting the attributes and values of a higher-level `element` tag:

```
<element handler="table" tag="table" inherit="NONE">
   <element tag="*" handler="skip" />
   <element tag="tr">
      <element tag="td" id="HTML.tablecell"
                       inherit="HTML.paragraph" >
          <attribute name="colspan"/>
          <attribute name="rowspan"/>
          <attribute handler="style" name="style"/>
          <element tag="p" handler="strip" mapattributes="true"
                       break="br" inherit="HTML.paragraph"/>
          </element>
          <element tag="th" inherit="HTML.tablecell"
                       maptag="td" />
   </element>
</element>
```

This example contains the rules for converting a table. `inherit="NONE"` is used to specify that all previously defined conversion rules are not applied to tables.

4.4.10 break

The `break` attribute can be used to define the conversion of text breaks, e.g.

```
<element handler="table" tag="table" inherit="NONE">
   <element tag="*" handler="skip" />
   <element tag="tr">
      <element tag="td" id="HTML.tablecell"
                       inherit="HTML.paragraph" >
          <attribute name="colspan"/>
          <attribute name="rowspan"/>
          <attribute handler="style" name="style"/>
```

```
            <element tag="p" handler="strip" mapattributes="true"
                        break="br" inherit="HTML.paragraph"/>
            </element>
            <element tag="th" inherit="HTML.tablecell"
                        maptag="td" />
   </element>
</element>
```

This example contains the rules for converting a table. `break="br"` is used to realise text breaks within table cells in Word in the DOM input component too, while `<p>` tags are removed (`tag="p" handler="strip"`).

4.4.11 mapattributes

The `mapattributes` attribute can be used to import attributes from the Word document into FirstSpirit, e.g.

```
<element handler="table" tag="table" inherit="NONE">
   <element tag="*" handler="skip" />
   <element tag="tr">
      <element tag="td" id="HTML.tablecell"
                        inherit="HTML.paragraph" >
            <attribute name="colspan"/>
            <attribute name="rowspan"/>
            <attribute handler="style" name="style"/>
            <element tag="p" handler="strip" mapattributes="true"
                        break="br" inherit="HTML.paragraph"/>
            </element>
            <element tag="th" inherit="HTML.tablecell"
                        maptag="td" />
   </element>
</element>
```

This example contains the rules for converting a table. `mapattributes="true"` specifies that the attributes of the table from the Word document, e.g. `valign`, `bgcolor`, etc. are used in a DOM input component on import. The attributes are not used if the value `false` is set.

4.4.12 class

The `class` attribute is used to map classes from the Word document with format or link templates in FirstSpirit, e.g.

```
<element class="Wichtig" maptag="important" tag="p"/>
```

In this example, the Word character style sheet "Wichtig" is mapped with the FirstSpirit format template with the abbreviation "important", i.e. paragraphs formatted in the Word document using the paragraph style sheet "Wichtig" are formatted with the character format template "Important" on being imported into the DOM Editor.

4.5 Handling unknown tags

Format or paragraph style sheets from the Word document, which are not defined in the XML rulesets cannot be accordingly converted in FirstSpirit.

A default response can be defined for this case, e.g. that all unknown formatting from Word is converted in a paragraph (`<p>`) in FirstSpirit.

To do this, the `handler` attribute in the `element` tag for continuous text paragraphs is set to the value "default" (see Chapter 4.4.4 page 31), e.g.

```
<element default="true" handler="map" id="HTML.paragraph" tag="p">
```

An `element` tag with `tag="*"` is created to format "unknown" elements from Word with the "paragraph formatting" defined above. `*` acts as a wildcard.

```
<element handler="default" tag="*"/>
```

4.6 Example: Creating a rule to reproduce an individual Word format style sheet

This chapter explains the procedure for creating your own XML rules, using the example of the Word style sheet "Hinweis" (German for “Note”).

4.6.1 Step 1: Preparing the Word document

The Word document contains the "Hinweis" style sheet. It has the following properties: red font colour (RGB colour 255, 0, 0) and bold. A paragraph is formatted with this style sheet. In the Word test document e.g.

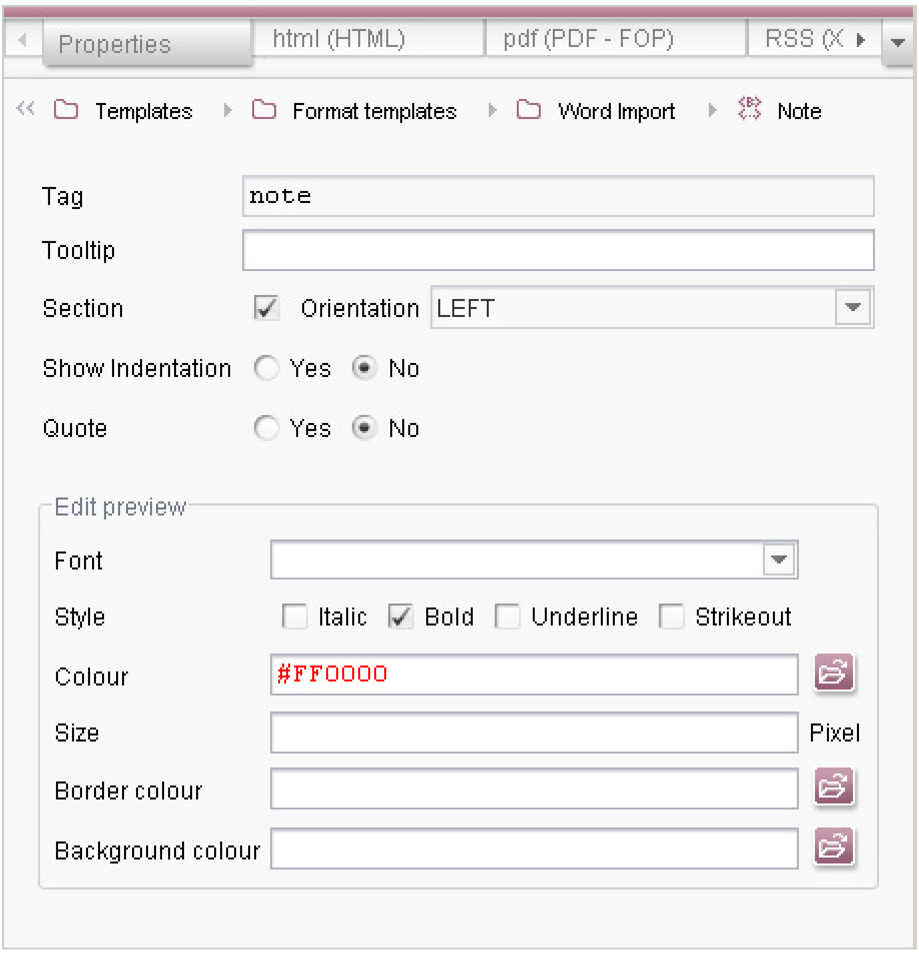
This text is formatted with the paragraph format "Hinweis", with a font colour with RGB value 255, 0, 0."

4.6.2 Step 2: Creating the necessary format template in FirstSpirit

A format template is required in FirstSpirit, which can reproduce the formatting of the

Word "Hinweis" style sheet in the DOM Editor. It should also be red and bold.

A format template with the following properties is created for this purpose:



Tag: note
Tooltip:
Section: [✓] Orientation: LEFT
Show Indentation: Yes [✓ No]
Quote: Yes [✓ No]

Edit preview
Font:
Style: ☐ Italic [✓ Bold] ☐ Underline ☐ Strikeout
Colour: #FF0000
Size: Pixel
Border colour:
Background colour:

Figure 4-2: "note" format template

In addition, the output channels also have to be configured accordingly.

4.6.3 Step 3: Outputting the Word document in HTML

This paragraph is output as follows in the HTML format of the Word document:

```
<p class="Hinweis">This text is formatted with the paragraph
format "Hinweis", with the font colour RGB 255, 0, 0. </p>
```

4.6.4 Step 4: Analysis of the HTML

The `<p>` indicates that it is a paragraph, `class` gives the name of the style sheet.

4.6.5 Step 5: Creating the XML rule definition

An `element` tag is required for the rule definition. As this is an additional rule description for a paragraph, the tag must be located on the level below the rule description for the HTML tag "body". For the `<p>` from the HTML, the XML requires a `tag` attribute:

```
tag="p"
```

However, all properties defined in the XML for paragraphs should also apply to this format template. This is implemented with the `inherit` attribute:

```
inherit="HTML.paragraph"
```

`HTML.paragraph` is the ID of the element (in this case the rule description for paragraphs), whose values and attributes are to be imported.

As this is a self-defined Word style sheet, it is defined with `class`. The `class` attribute with which the name of the Word style sheet is given must therefore be used in the XML ruleset:

```
class="Hinweis"
```

The `maptag` attribute is imported into the corresponding FirstSpirit format template:

```
maptag="note"
```

4.7 Default ruleset of the FirstSpirit Office module

The following XML ruleset is supplied as a standard component with the Office module:

```
<?xml version="1.0" encoding="UTF-8"?>
<ImportRuleSets>
    <mapping description="Project local import ruleset definition"
linkConfigExternal="textlinkexternal.standard"
linkConfigInternal="textlinkinternal.standard" mimeType="text/html"
name="Project local Import Ruleset" versionTag="22">

        <style mapname="align" name="text-align"/>
        <style mapname="foregroundcolor" name="color"/>
        <style mapname="bgcolor" name="background"/>
        <style name="list-style"/>

        <element handler="strip" tag="html">
            <attribute handler="style" name="style"/>
            <text action="ignore"/>
            <element tag="head" handler="find" findtag="title" />
            <element handler="strip" id="HTML.body" tag="body">

                <text action="default" />
                <element tag="*" handler="default" />
                <element tag="div" handler="strip" />
                <element tag="p" class="Wichtig" maptag="important"
                          inherit="HTML.paragraph" />
                <element tag="p" class="Hinweis" maptag="note"
                    inherit="HTML.paragraph" />

                <element default="true" handler="map" id="HTML.paragraph"
                          tag="p">
                    <text action="keep" />
                    <element tag="*" handler="strip" />
                    <element tag="span" handler="strip" />

                    <element content="IGNORE" handler="media" mediaref="src"
                          tag="img">
                        <attribute name="src"/>
                    </element>

                    <element maptag="br" tag="br"/>
                    <element maptag="b" tag="b"/>
                    <element tag="strong" maptag="b" />
                    <element maptag="i" tag="i"/>
                    <element maptag="pre" tag="pre"/>
                                 <element maptag="u" tag="u"/>
                                 <element maptag="s" tag="s"/>

                    <element maptag="h1" tag="h1"/>
                    <element maptag="h2" tag="h2"/>
                    <element maptag="h3" tag="h3"/>
                    <element maptag="h4" tag="h4"/>
                    <element maptag="h5" tag="h5"/>

                    <element tag="p" class="Wichtig" maptag="important"
                          inherit="HTML.paragraph" />
                    <element tag="p" class="Hinweis" maptag="note"
                          inherit="HTML.paragraph" />
                    <element handler="object" maptag="link" tag="a">
                        <attribute mapname="target" name="href"/>
                    </element>

                    <element tag="ul" id="HTML.list">
                        <element inherit="HTML.paragraph" tag="li"
                          mapattributes="false"/>
                    </element>

                    <element tag="ol" maptag="ul" inherit="HTML.list" />

                    <element handler="table" tag="table" inherit="NONE">
                        <element tag="*" handler="skip" />
                        <element tag="tr">
                            <element tag="td" id="HTML.tablecell"
                          inherit="HTML.paragraph" >
                                <attribute name="colspan"/>
                                <attribute name="rowspan"/>
                                <attribute handler="style" name="style"/>
                                <element tag="p" handler="strip"
                                       mapattributes="true" break="br"
                                       inherit="HTML.paragraph"/>
                            </element>
                            <element tag="th" inherit="HTML.tablecell"
                                       maptag="td" />
                        </element>
                    </element>
                </element>

            </element>
        </element>
    </mapping>
    <mapping description="Project local import ruleset definition (generic
link)" linkConfigExternal="textlinkexternal"
linkConfigInternal="textlinkinternal" mimeType="text/html" name="Project
local Import Ruleset (generic link)" versionTag="22">

        <style mapname="align" name="text-align"/>
        <style mapname="foregroundcolor" name="color"/>
        <style mapname="bgcolor" name="background"/>
        <style name="list-style"/>

        <element handler="strip" tag="html">
            <attribute handler="style" name="style"/>
            <text action="ignore"/>
            <element tag="head" handler="find" findtag="title" />
            <element handler="strip" id="HTML.body" tag="body">

                <text action="default" />
                <element tag="*" handler="default" />
                <element tag="div" handler="strip" />
                <element tag="p" class="Wichtig" maptag="important"
                          inherit="HTML.paragraph" />
                <element tag="p" class="Hinweis" maptag="note"
                    inherit="HTML.paragraph" />

                <element default="true" handler="map" id="HTML.paragraph"
                          tag="p">
                    <text action="keep" />
                    <element tag="*" handler="strip" />
                    <element tag="span" handler="strip" />

                    <element content="IGNORE" handler="media" mediaref="src"
                          tag="img">
                        <attribute name="src"/>
                    </element>

                    <element maptag="br" tag="br"/>
                    <element maptag="b" tag="b"/>
                    <element tag="strong" maptag="b" />
                    <element maptag="i" tag="i"/>
                    <element maptag="pre" tag="pre"/>
                                 <element maptag="u" tag="u"/>
                                 <element maptag="s" tag="s"/>

                    <element maptag="h1" tag="h1"/>
                    <element maptag="h2" tag="h2"/>
                    <element maptag="h3" tag="h3"/>
                    <element maptag="h4" tag="h4"/>
                    <element maptag="h5" tag="h5"/>

                    <element tag="p" class="Wichtig" maptag="important"
                          inherit="HTML.paragraph" />
                    <element tag="p" class="Hinweis" maptag="note"
                          inherit="HTML.paragraph" />

                    <element handler="object" maptag="link" tag="a">
                        <attribute mapname="target" name="href"/>
                    </element>

                    <element tag="ul" id="HTML.list">
                        <element inherit="HTML.paragraph" tag="li"/>
                    </element>

                    <element tag="ol" maptag="ul" inherit="HTML.list" />

                    <element handler="table" tag="table" inherit="NONE">
                        <element tag="*" handler="skip" />
                        <element tag="tr">
                            <element tag="td" id="HTML.tablecell"
                          inherit="HTML.paragraph" >
                                <attribute name="colspan"/>
                                <attribute name="rowspan"/>
                                <attribute handler="style" name="style"/>
                                <element tag="p" handler="strip"
                                       mapattributes="true" break="br"
                                       inherit="HTML.paragraph"/>
                            </element>
                            <element tag="th" inherit="HTML.tablecell"
                                 maptag="td" />
                        </element>
                    </element>
                </element>

            </element>
        </element>
    </mapping>
    <mapping description="use a default text only import handler."
mimeType="text/plain" name="Standard (text only import)" versionTag="21"/>

</ImportRuleSets>
```

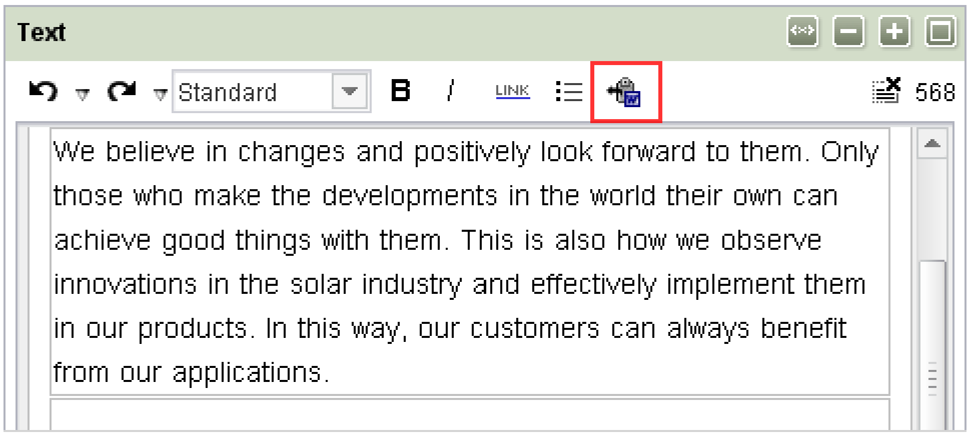
5 Configuring the Input Components

The Office function can be used in the DOM Editor and in the DOM Table.

Only tables from Word documents can be imported into the DOM Table input component, not continuous texts (see also Chapter 6.7 page 61).

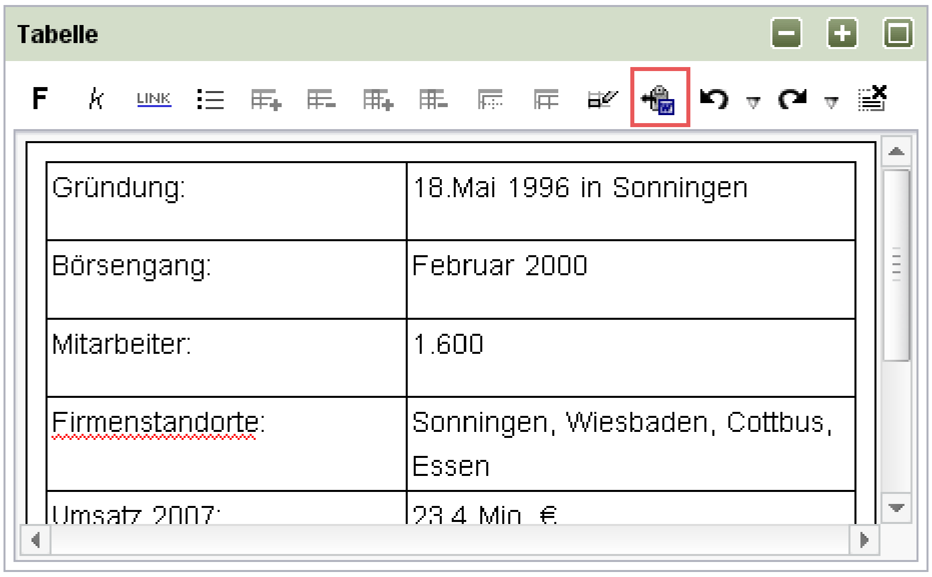
5.1 Activating the import function

In order to make the Office module function available in the DOM Editor or in the DOM table, the parameter `enableImport="yes"` must be added to the required section, page or table templates in the form area of the CMS_INPUT_DOM or CMS_INPUT_DOMTABLE input component. The input components are then assigned an import icon when they are integrated in the Page or Content Store:



We believe in changes and positively look forward to them. Only those who make the developments in the world their own can achieve good things with them. This is also how we observe innovations in the solar industry and effectively implement them in our products. In this way, our customers can always benefit from our applications.

Figure 5-1: DOM Editor with icon for the Office import



Tabelle

Gründung:	18.Mai 1996 in Sonningen
Börsengang:	Februar 2000
Mitarbeiter:	1.600
Firmenstandorte:	Sonningen, Wiesbaden, Cottbus, Essen
Umsatz 2007:	23,4 Mio. €

Figure 5-2: DOM Table with icon for the Office import

5.2 Limiting selectable rulesets

The optional parameter `importRuleset` can be used to define which rulesets are to be made available to the respective DOM Editor or DOM Table and can therefore be selected by the editor. This is done by giving the name of the ruleset defined in the XML definition (see Chapter 4.3.2 page 20) in double inverted commas. These can be both rulesets defined for the whole server and those defined for the respective project only, e.g.:

```
importRuleset="Project"
```

In this case, only the ruleset named "Project" is made available.

More than one ruleset can be given. In that case, the rulesets must be given separated by a comma.

If the parameter is not given, all rulesets valid for the respective project and the rulesets valid server-wide are displayed for selection in the input component.

5.3 Restrictions (format and link templates, lists, tables)

In order for the assignment between the content from the Word document and the FirstSpirit format templates to work properly, these format templates must not be restricted in the required DOM Editor or DOM Table.

The use of format templates can be restricted in the DOM-Editor by using the `<FORMATS>` tag. Each format template which is to be admitted within the respective input component is given in a separate `<TEMPLATE>` tag between the opening and closing `<FORMATS>` tag:

```
<FORMATS>
   <TEMPLATE name="Tag of the format template">
</FORMATS>
```

If no `<FORMATS>` tag is specified all format templates are allowed.

If a formatting, e.g. **bold** is used in the Word document, also the FirstSpirit format template for **bold** (standard format template with the tag "b") should be allowed for the DOM-Editor. Otherwise, content formatted in bold can be imported into the DOM-Editor and the also format willl be displayed, but the format can not be changed in the DOM-Editor. The icon for the format **bold** is deactivated in this case.

If the use of format templates is restricted for the DOM-Editor, to which the Word document is to be imported, pay attention that all formats used in Word are admitted in the DOM-Editor.

The same applies to links, lists and tables:

Links: The use of links can be restricted in the DOM-Editor by using the `<LINKEDITORS>` tag. Each link template which is to be admitted within the respective input component is given in a separate `<LINKEDITOR>` tag between the opening and closing `<LINKEDITORS>` tag, e.g.:

```
<LINKEDITORS>
   <LINKEDITOR name="Unique name of the link template">
</LINKEDITORS>
```

If no `<LINKEDITORS>` tag is specified all link templates are allowed.

If the use of link templates is restricted for the DOM-Editor, to which the Word document is to be imported, pay attention that the required link templates (for the use of the standard rulesets "textlinkinternal.standard" or "textlinkinternal" and “textlinkexternal.standard“ or “textlinkexternal“) are admitted in the DOM-Editor.

Lists: The use of lists can be restricted in the DOM-Editor using the parameter `list="NO"`. If lists are used in the Word document, which are to be imported into the DOM-Editor and edited there, pay attention that the parameter is not set to `NO`. Depending on the project configuration and the Word document the list configuration must be possibly changed, too (attributes `listConfig` and `listDefaultConfig`).

> **!** *For notes about the correct output of nested lists see also Chapter 5.4 page 45.*

Tables (only DOM-Editor): The use of tables in the DOM-Editor (so-called “Inline tables“) can be controlled using the attribute `table`. If tables are used in the Word document the table mode should be activated for the respective DOM-Editor (`table="YES"`).

For further information about the configuration of the DOM-Editor and the DOM-Table see FirstSpirit Online Documentation, section “Template development” / “Forms” / “Input components” / “DOM” or “DOMTABLE”.

5.4 Output of lists

If nested lists are used in Microsoft Office Word, their representation in HTML differs from the recommendations of the World Wide Web Consortiums (W3C).

Example HTML code, generated from Word:

```
<ul>
  <li>abc</li>
  <ol>
     <li>bcd</li>
  </ol>
  <li>cde</li>
</ul>
```

According to the HTML 4.01 specification of the W3C only LI elements are allowed for UL/OL elements (see http://www.w3.org/TR/html401/struct/lists.html).

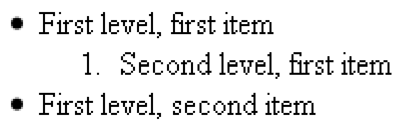
When imported into a DOM editor a valid list will be created from nested list, made in Word, by inserting automatically LI elements:

```
<ul style="1">
```

```
  <li>abc</li>
  <li><ul>
     <li>bcd</li>
  </ul></li>
  <li>cde</li>
</ul>
```

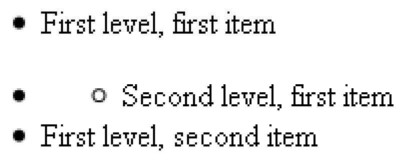
In FirstSpirit ordered lists ("ol") are saved as unordered lists ("ul")!

However, there are differences when output by the browser, as shown in the following figures: Figure 5-3 shows the output of the HTML code generated by Microsoft Word in the browser, Figure 5-4 shows the output of the HTML code generated by FirstSpirit in the browser:



- First level, first item
    1. Second level, first item
- First level, second item

Figure 5-3: Output of nested lists (Microsoft Word)



- First level, first item
-
    - Second level, first item
- First level, second item

Figure 5-4: Output of nested lists (FirstSpirit)

To adjust the render method of FirstSpirit in the browser to that of Word, the FirstSpirit standard format template "List entry " with the tag "li" must be edited.

For this purpose

```
<li>$CMS_VALUE(#content)$</li>
```

in the HTML channel must be replaced as follows:

```
$CMS_IF(!#listitem.element.firstChild.isNull &&
     #listitem.element.firstChild.nodeName == "ul")$
$CMS_VALUE(#content)$
$CMS_ELSE$
     <li>$CMS_VALUE(#content)$</li>
$CMS_END_IF$
```

If the standard format template "li" is changed in this way, this will have an effect on the output of ***all*** *lists in FirstSpirit. To prevent this and to change the output only for DOM editors which are to work with nested lists from Word, accordingly adjusted format templates can be used. At least the format template "Standard" (for sections which are to be formatted with a <p> tag) should be adjusted, as the following example shows:*

For this purpose a format template with activated option "Section" is to be created. A variable (e.g. with the name *_isWordList*) must be defined in the HTML channel:

```
$CMS_SET(_isWordList, true)$
  <p>$CMS_VALUE(#content)$</p>
$CMS_SET(_isWordList, null)$
```

This format template must be available in the DOM editor, which is to use nested lists:

```
<FORMATS>
  <TEMPLATE name="IDENTIFIER_FORMAT_TEMPLATE"/>
</FORMATS>
```

The HTML channel of the standard format template "li" must be modified in this case as follows:

```
$CMS_IF(!_isWordList.isNull &&
          _isWordList &&
          !#listitem.element.firstChild.isNull &&
          #listitem.element.firstChild.nodeName == "ul")$
$CMS_VALUE(#content)$
$CMS_ELSE$
<li>$CMS_VALUE(#content)$</li>
```

6 Editorial Work in JavaClient

The following explanations primarily refer to the Word test document supplied with the module.

6.1 Explanations to the Word document

The following table lists the style sheets used in the Word document and gives their corresponding FirstSpirit templates (some are default format templates, some are project-specific templates from the "Mithras Energy" demo project):

Word style sheet	Example	FirstSpirit template
Überschrift 1	"Test document for the import from Word with FirstSpirit OFFICE"	h1
Überschrift 2	"Simple body text"	h2
Überschrift 3	"Body text with line breaks"	h3
Standard	"Section 1: Lorem ipsum dolor sit amet..."	p
Fett	"This sentence is completely bolded."	b
Kursiv	"This sentence is completely italicised."	k
Unterstrichen	"This sentence is completely underlined."	u
Aufgezählt (Symbol) • Aufgezählt	„• First bullet point"	ul / li

Word style sheet	Example	FirstSpirit template
Aufgezählt (Letter) a. Aufgezählt	„a. Second level, first bullet point“	ul / li
Hinweis	"This text is formatted with the paragraph format “Hinweis“, with the font colour RGB 255, 0, 0."	note
Wichtig	"3.4 (Paragraph formatting = "Wichtig")"	important
Hyperlink	“http://www.firstspirit.de”, “www.firstspirit.de", "latest news"	external link template `textlinkexternal.standard` or `textlinkexternal`
<Pictures>	FirstSpirit™	internal link template `textlinkinternal.standard` or `textlinkinternal`

The **paragraphs** "Section 1", "Section 2" and "Section 3" represent text sections defined in HTML by <p> tags. There is also an empty line between paragraphs 2 and 3. There is a soft line break in "Section 4", in front of the line "Consetetur sadipscing elitr...".

The FirstSpirit logo is integrated in the text body once as an **picture** (paragraph "pictures in lines"), and it is also present once as its own paragraph (paragraph "pictures in paragraphs"). In addition, the e-Spirit logo is integrated in a table cell ("Tables with merging and formatting").

The **character formatting** used are “Fett“, “Kursiv“, “Unterstrichen“, “Durchgestrichen“, in English: "bold", "italic", "underline", "struck through", and combinations of these. There default heading levels contained in Word are used as **paragraph formatting**. In addition, the self-defined Word paragraph style sheet "Hinweis" is used.

Links can be reproduced in the DOM Editor as an external link, irrespective of whether the link is identified as such in the Word document by formatting (blue font, underlined) (e.g. "http://www.firstspirit.de") or a link which is not identified as such (e.g. "latest news", linked to "http://www.e-spirit.de").

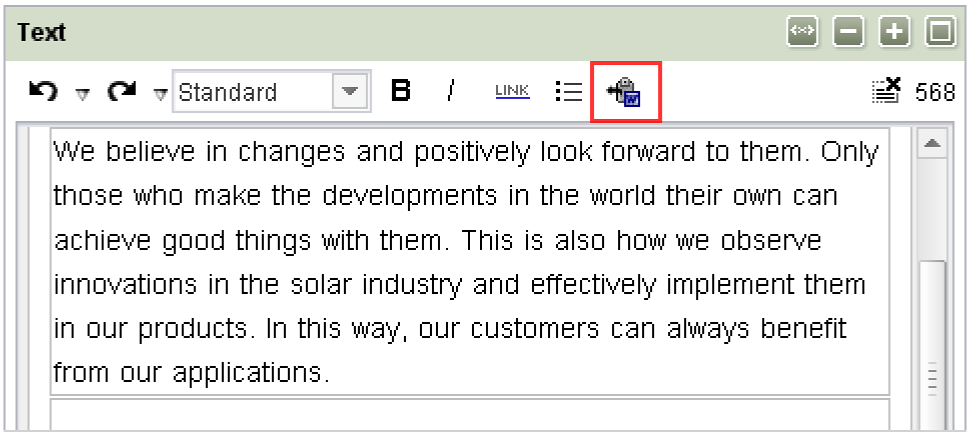
Lists are currently only displayed in the DOM Editor by the supplied XML ruleset using the FirstSpirit "em dash" formatting. Due to limiting in the export format from Word, lists can only be correctly implemented in FirstSpirit up to the second level. All levels above this are formatted as text, e.g. with spaces instead of with tabulator indents.

For notes about the correct output of nested lists see also Chapter 5.4 page 45.

The Word format styles “Standard”, “Fett“, “Kursiv“, “Unterstrichen“ etc., in English "standard", "bold", "italics", "underline", etc. are also used in the **tables**. The cells are consecutively numbered, and the number in front of the dot indicates the column number and the number after the dot indicates the row number. For example, "2.3" stands for a cell in the second column of the third row. Merged cells are represented by a "+": "2.2 + 3.2" means, e.g., that the cell of the second column in the second row has been merged with the cell below it (second column, third row). There is an empty line between each of the tables and the preceding text.

6.2 Importing into FirstSpirit JavaClient

If the FirstSpirit Office module is configured for a project and is activated in a DOM Editor or a DOM table, the "Import" button is visible in JavaClient:



We believe in changes and positively look forward to them. Only those who make the developments in the world their own can achieve good things with them. This is also how we observe innovations in the solar industry and effectively implement them in our products. In this way, our customers can always benefit from our applications.

Figure 6-1: DOM Editor with Import button

In order to import content from Word into the DOM Editor, the cursor must first be placed in the position in which the content is to be inserted.

The content to be imported is then marked in the Word document and is copied onto the clipboard using <CTRL> + <C> or using the context menu (right-click mouse button). After clicking the Import button in the input component, or using the key shortcut <CTRL> + <V>, the content from the Word document is inserted (pasted) directly into the input component at the selected position and automatic formatting takes place in accordance with the ruleset specified by the template developer.

If several rulesets are available for the input component, a window appears from which the ruleset by which the content is to be transformed can be selected:

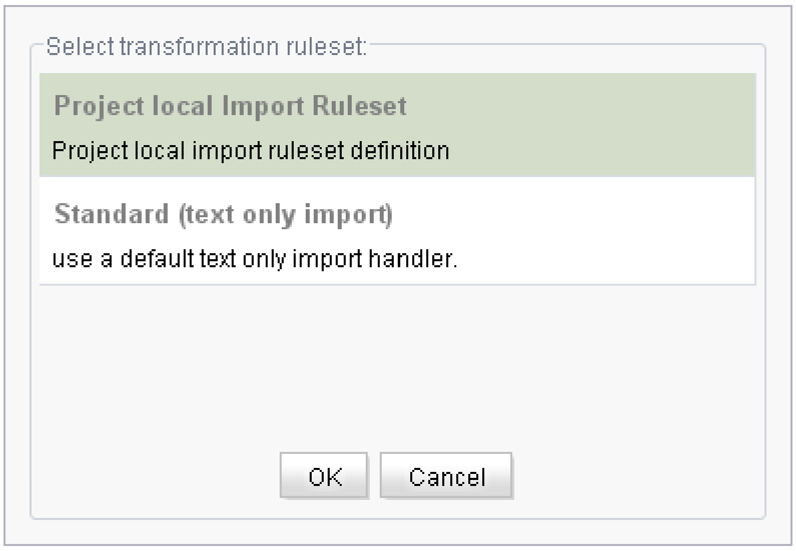

Select transformation ruleset:
Project local Import Ruleset
Project local import ruleset definition
Standard (text only import)
use a default text only import handler.
OK
Cancel

Figure 6-2: Select transformation ruleset

The required ruleset is selected with the mouse and the selection is confirmed with OK. The content of the Word document is then inserted in the input component at the selected position and is formatted according to the required ruleset. Cancel is used to past the text from the Word document as unformatted text.

The content can then be edited as usual using the functions available in the input component.

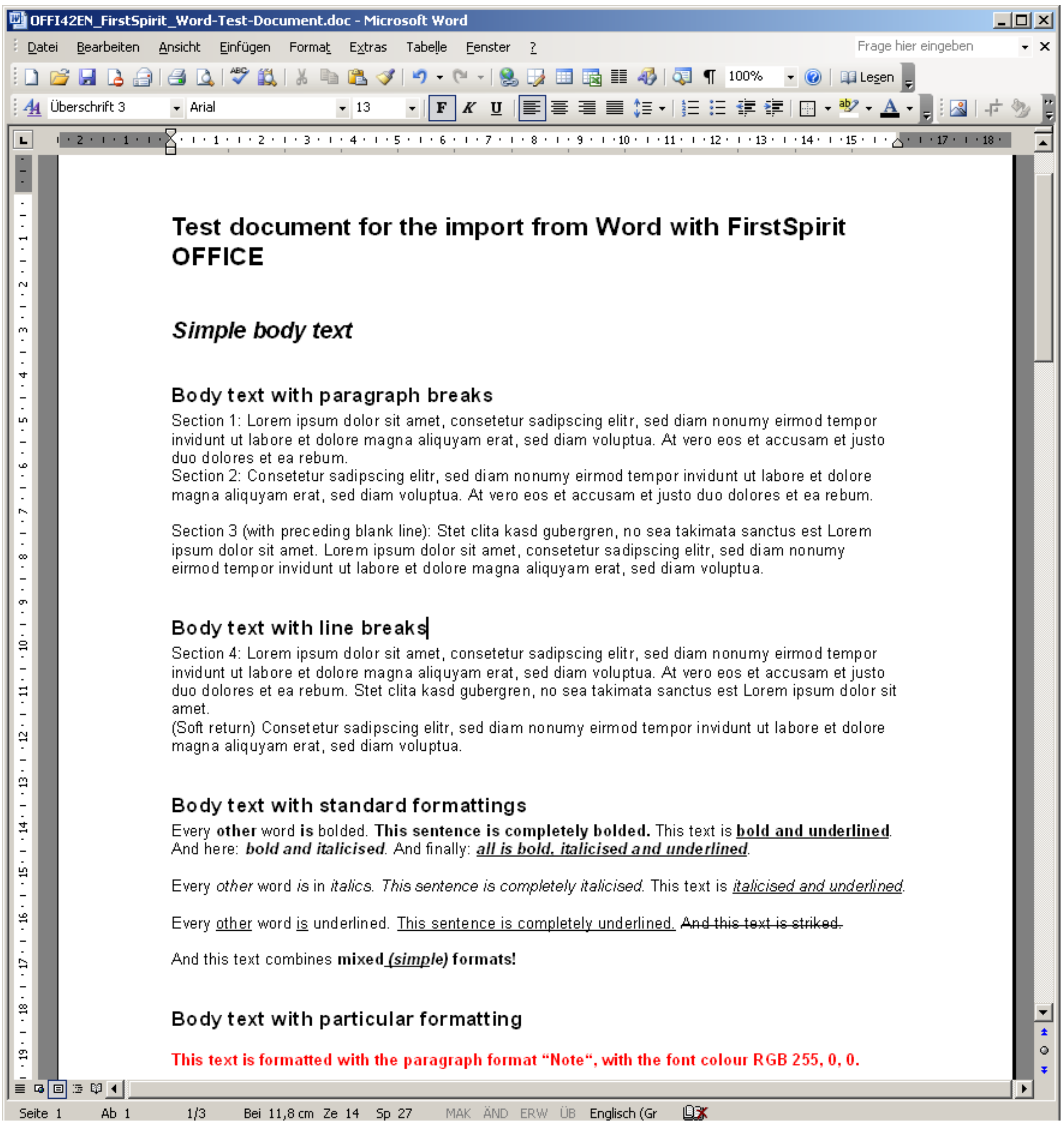
> ! *If no Word format was copied onto the clipboard, after the Import button is clicked (or <CTRL> + <V>), a message appears "No compatible format found on the clipboard!". The import does not take place.*
>
> ! *In order to ensure the Office module functions properly, continuous text from Word documents should only be inserted into the DOM Editor or into so-called inline tables, tables from Word should only be inserted into the DOM Table or into so-called inline tables. Otherwise, the inserted content cannot be further edited.*

6.3 Importing formatted texts

Continuous texts from a Word document can be imported into DOM input components along with their paragraph breaks, and character and paragraph formatting.

6.3.1 Text from Word documents

The following area is marked in the Word test document and is copied onto the clipboard (<CTRL> + <C>):



OFFI42EN_FirstSpirit_Word-Test-Document.doc - Microsoft Word

Test document for the import from Word with FirstSpirit OFFICE

Simple body text

Body text with paragraph breaks

Section 1: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.
Section 2: Consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

Section 3 (with preceding blank line): Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Body text with line breaks

Section 4: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.
(Soft return) Consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Body text with standard formattings

Every **other** word **is** bolded. **This sentence is completely bolded.** This text is **bold and underlined**.
And here: ***bold and italicised***. And finally: ***all is bold, italicised and underlined***.

Every *other* word *is* in *italics*. *This sentence is completely italicised*. This text is *italicised and underlined*.

Every other word is underlined. This sentence is completely underlined. ~~And this text is striked.~~

And this text combines **mixed *(simple)* formats!**

Body text with particular formatting

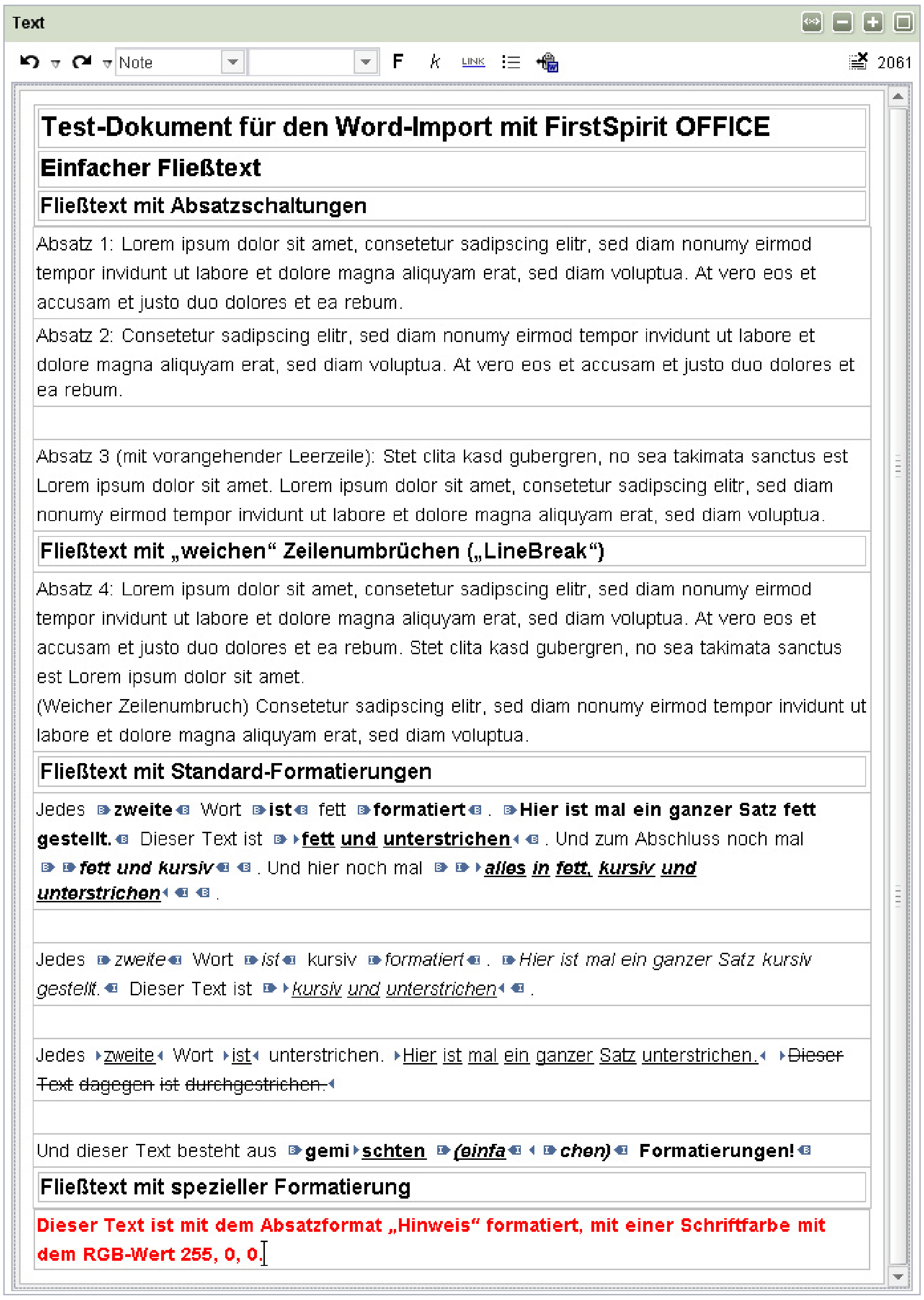
This text is formatted with the paragraph format "Note", with the font colour RGB 255, 0, 0.

Seite 1 Ab 1 1/3 Bei 11,8 cm Ze 14 Sp 27 MAK ÄND ERW ÜB Englisch (Gr

Figure 6-3: Word test document: Continuous text

6.3.2 Text in the DOM Editor

After it has been imported using the Office function, the text from the test document can be displayed in the DOM Editor as follows:



Test-Dokument für den Word-Import mit FirstSpirit OFFICE

Einfacher Fließtext

Fließtext mit Absatzschaltungen

Absatz 1: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

Absatz 2: Consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

Absatz 3 (mit vorangehender Leerzeile): Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Fließtext mit „weichen“ Zeilenumbrüchen („LineBreak“)

Absatz 4: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.
(Weicher Zeilenumbruch) Consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Fließtext mit Standard-Formatierungen

Jedes **zweite** Wort **ist** fett **formatiert**. **Hier ist mal ein ganzer Satz fett gestellt.** Dieser Text ist **fett und unterstrichen**. Und zum Abschluss noch mal ***fett und kursiv***. Und hier noch mal ***alles in fett, kursiv und unterstrichen***.

Jedes *zweite* Wort *ist* kursiv *formatiert*. *Hier ist mal ein ganzer Satz kursiv gestellt.* Dieser Text ist *kursiv und unterstrichen*.

Jedes zweite Wort ist unterstrichen. Hier ist mal ein ganzer Satz unterstrichen. ~~Dieser Text dagegen ist durchgestrichen.~~

Und dieser Text besteht aus **gemi****schten** ***(einfa******chen)*** **Formatierungen!**

Fließtext mit spezieller Formatierung

Dieser Text ist mit dem Absatzformat „Hinweis“ formatiert, mit einer Schriftfarbe mit dem RGB-Wert 255, 0, 0.

Figure 6-4: DOM Editor with text from the Word document (DE)

Explanation: The headings of the first three levels are reproduced in FirstSpirit on the corresponding FirstSpirit format templates with the abbreviations (quicktags) "h1", "h2" and "h3". Paragraph 1 and paragraph 2 represent text sections, which are defined in HTML by `<p>...</p>`. They are reproduced in the DOM Editor in the default format template "Standard". There is an empty line between paragraphs 2 and 3.

The character formatting from the Word document for bold, italics and underline are reproduced in the FirstSpirit format templates in the DOM Editor with the abbreviations (quicktags) "b", "i", "u" and "s". The "Hinweis" paragraph formatting from the Word document is reproduced on the format template in the DOM Editor with the abbreviation "note".

6.3.3 Exemplary HTML output

Depending on the template developer's definition, the HTML output on the website could look like the following:

```
<h1> Test document for the import from Word with FirstSpirit
OFFICE </h1>

<h2>Simple body text</h2>

<h3>Body text with paragraph breaks</h3>
          <p class="section">Section 1: Lorem ipsum dolor sit
          amet, consetetur sadipscing elitr, sed diam nonumy
          eirmod tempor invidunt ut labore et dolore magna
          aliquyam erat, sed diam voluptua. At vero eos et accusam
          et justo duo dolores et ea rebum.</p>

          ...
```

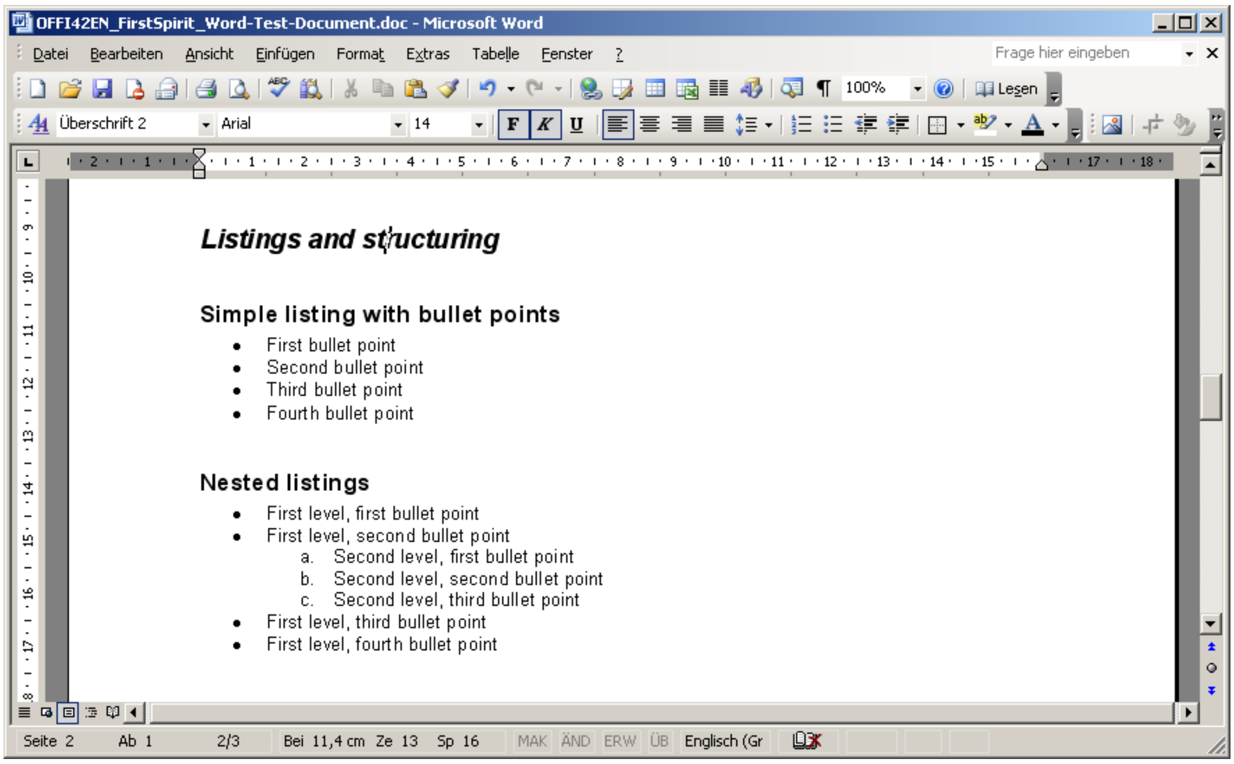
6.4 Importing lists

Due to limiting in the export format, lists from Word are only imported up to their second level when imported into DOM input components.

For notes about the correct output of nested lists see also Chapter 5.4 page 45.

6.4.1 Lists from Word documents

The following area is marked in the Word test document and is copied onto the clipboard (<CTRL> + <C>):


OFFI42EN_FirstSpirit_Word-Test-Document.doc - Microsoft Word
Datei Bearbeiten Ansicht Einfügen Format Extras Tabelle Fenster ?
Frage hier eingeben
100%
Lesen
Überschrift 2
Arial
14
Seite 2 Ab 1 2/3 Bei 11,4 cm Ze 13 Sp 16 MAK ÄND ERW ÜB Englisch (Gr

Listings and structuring

Simple listing with bullet points

- First bullet point
- Second bullet point
- Third bullet point
- Fourth bullet point

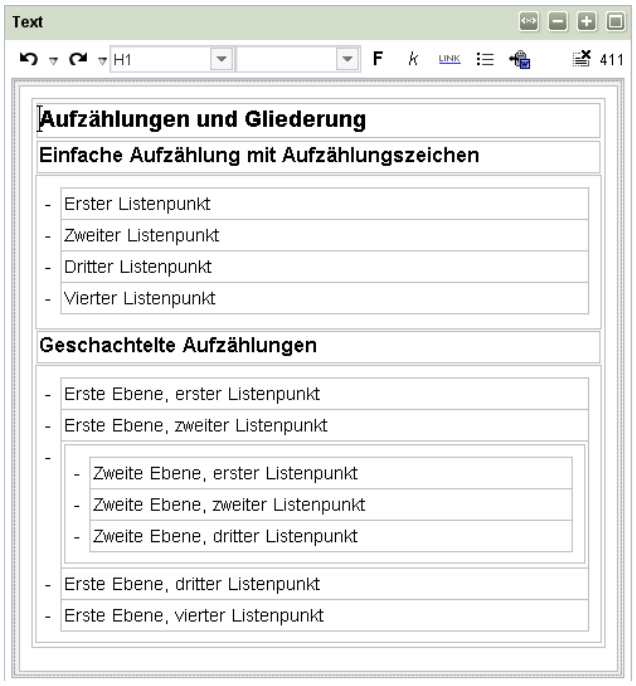
Nested listings

- First level, first bullet point
- First level, second bullet point
    a. Second level, first bullet point
    b. Second level, second bullet point
    c. Second level, third bullet point
- First level, third bullet point
- First level, fourth bullet point

Figure 6-5: Word test document: Lists

6.4.2 Lists in the DOM Editor

After it has been imported using the Office function, the text with lists from the test document can be displayed in the DOM Editor as follows:



Aufzählungen und Gliederung

Einfache Aufzählung mit Aufzählungszeichen

- Erster Listenpunkt
- Zweiter Listenpunkt
- Dritter Listenpunkt
- Vierter Listenpunkt

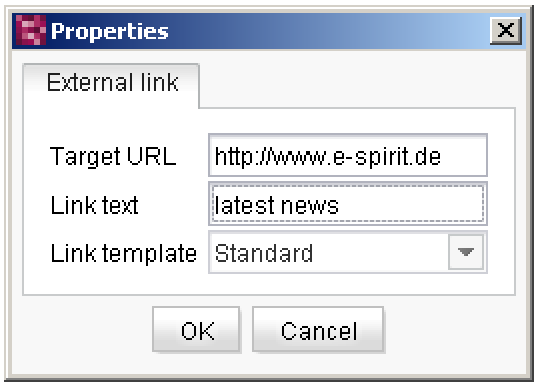
Geschachtelte Aufzählungen

- Erste Ebene, erster Listenpunkt
- Erste Ebene, zweiter Listenpunkt
-
  - Zweite Ebene, erster Listenpunkt
  - Zweite Ebene, zweiter Listenpunkt
  - Zweite Ebene, dritter Listenpunkt
- Erste Ebene, dritter Listenpunkt
- Erste Ebene, vierter Listenpunkt

Figure 6-6: : DOM Editor with lists from the Word document (DE)

6.5 Importing links

When links are imported they are realised as external links when they are pasted into the DOM Editor:



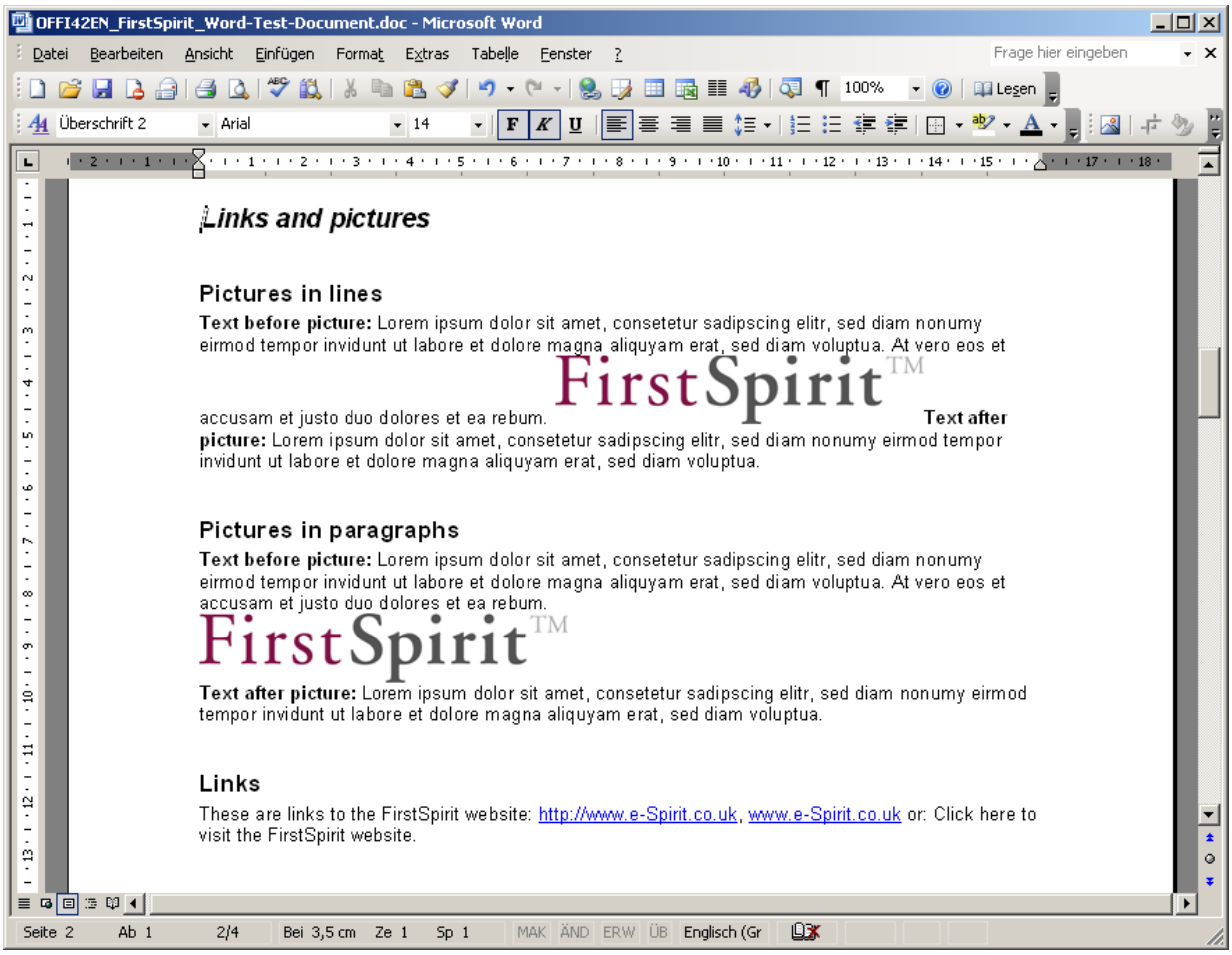
Target URL	http://www.e-spirit.de
Link text	latest news
Link template	Standard

Figure 6-7: External Link

The display text from the Word document (here "latest news") is imported into the "Link text" field, the link is imported into the "Target URL" field. All fields can be further edited as usual.

6.5.1 Links from Word documents

The following area is marked in the Word test document and is copied onto the clipboard (<CTRL> + <C>):



OFFI42EN_FirstSpirit_Word-Test-Document.doc - Microsoft Word
Datei Bearbeiten Ansicht Einfügen Format Extras Tabelle Fenster ?
Frage hier eingeben
100%
Lesen
Überschrift 2
Arial
14
Seite 2 Ab 1 2/4 Bei 3,5 cm Ze 1 Sp 1 MAK ÄND ERW ÜB Englisch (Gr

Links and pictures

Pictures in lines

Text before picture: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. **Text after picture:** Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Pictures in paragraphs

Text before picture: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

Text after picture: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

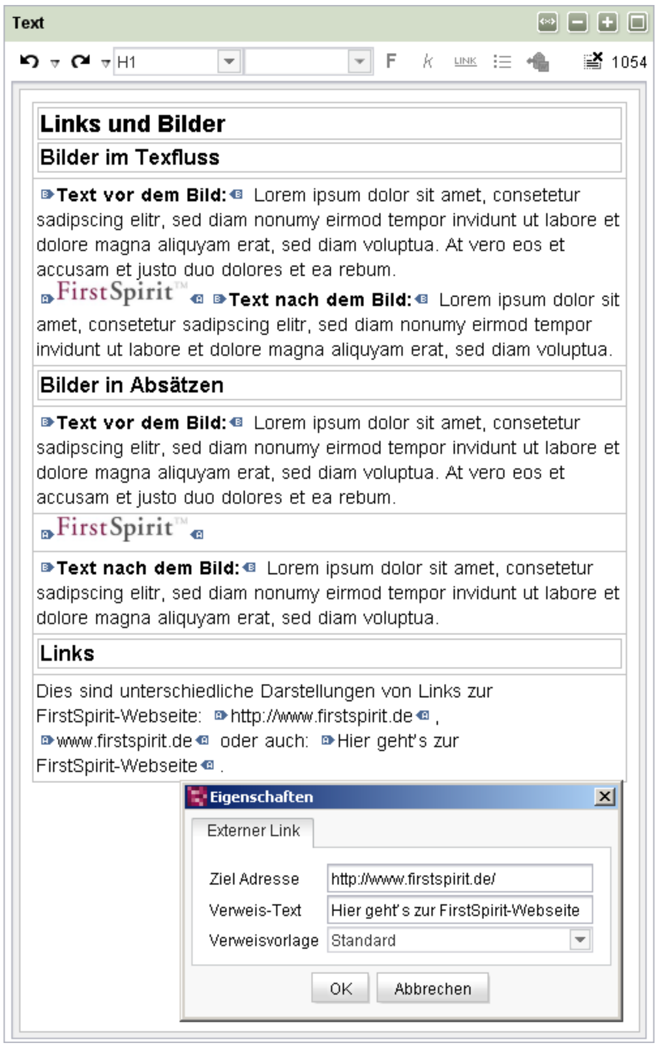
Links

These are links to the FirstSpirit website: http://www.e-Spirit.co.uk, www.e-Spirit.co.uk or: Click here to visit the FirstSpirit website.

Figure 6-8: Word test document: Links and pictures

6.5.2 Links in the DOM Editor

After it has been imported using the Office function, the text with links from the test document can be displayed in the DOM Editor as follows:


Text

H1 1054

Links und Bilder

Bilder im Texfluss

Text vor dem Bild: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.
FirstSpirit™ Text nach dem Bild: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Bilder in Absätzen

Text vor dem Bild: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

FirstSpirit™

Text nach dem Bild: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Links

Dies sind unterschiedliche Darstellungen von Links zur FirstSpirit-Webseite: http://www.firstspirit.de , www.firstspirit.de oder auch: Hier geht's zur FirstSpirit-Webseite .

Eigenschaften

Externer Link

Ziel Adresse	http://www.firstspirit.de/
Verweis-Text	Hier geht's zur FirstSpirit-Webseite
Verweisvorlage	Standard

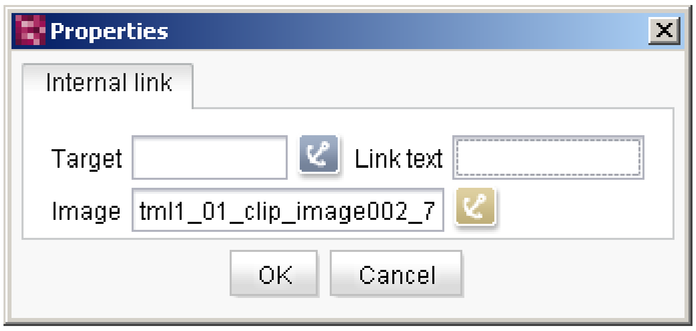
Figure 6-9: DOM Editor with pictures and links from the Word document (DE)

Depending on the configuration of a project external links can also be constructed on the basis of generic link editors. In this case, functionality and layout can differ from description above. For further information about the editorial use of Generic link editors see FirstSpirit Manual for Editors (JavaClient).

6.6 Importing pictures

6.6.1 Inserting into the DOM editor

If the clipboard contains one or several pictures, they are realised as an internal link when they are pasted into the DOM Editor. A differentiation is made between whether they are embedded in the text body ("pictures in lines") or occupy their own paragraph ("pictures in paragraphs") (see also Figure 6-9):



Target:
Link text:
Image: tml1_01_clip_image002_7

Figure 6-10: Internal link for displaying pictures

This is done using the link template, which is defined in the `mapping` element (see Chapter 4.3.2 page 20). The "Image" field contains the path to the picture in the Media Store (see Chapter 6.6.2 page 60).

Depending on the configuration of a project internal links can also be constructed on the basis of generic link editors. In this case, functionality and layout can differ from description above. For further information about the editorial use of Generic link editors see FirstSpirit Manual for Editors (JavaClient).

6.6.2 Creating pictures in the Media Store

On being imported, the pictures are automatically inserted in the Media Store, namely in a folder, which is also created as a new folder when the pictures are inserted into the DOM Editor.

The UID of the **folder** is formed according to the following schema:

```
import_date_time_user
```

The date is given in *YYmmDD* format, the time is given in *HH_MM* format and `User` is the login name of the user who inserted the picture, e.g.

```
import_09040115_16_Admin
```

The UID of the inserted **picture** is formed according to the following schema:

file*_path_filename*

The path always leads to a temporary folder on the local workstation ("Temp"), in which the picture is temporarily stored. The file name is given without file format (extension), e.g.

```
file____C__DOCUME_1_userAccount_1_LOCALE_1_Temp_msohtml1_01_clip_ima
ge002
```

This is the same form used to display folders and pictures are they have been added to the tree structure, a display name and a file name for the picture can be individually assigned later. These elements may have to be released in release projects.

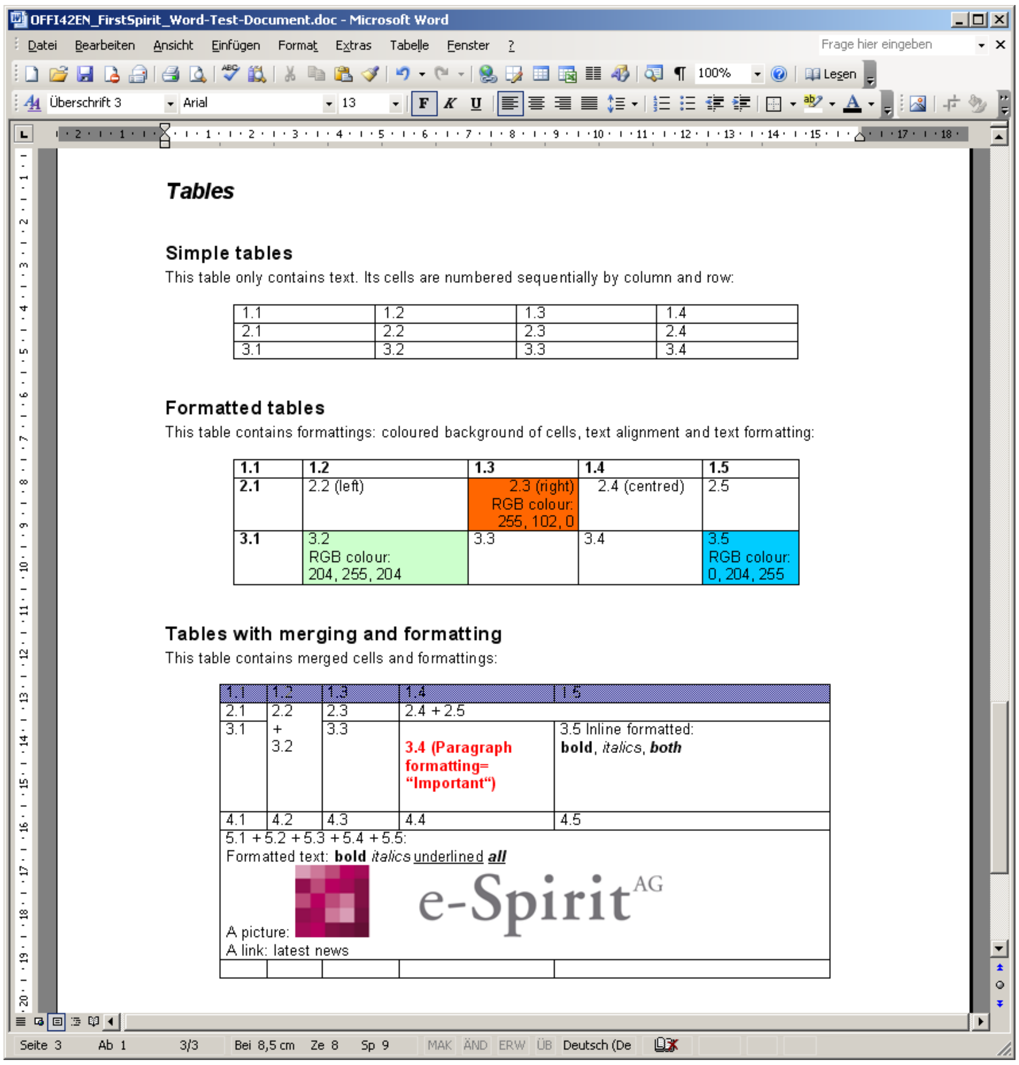
Each time pictures are imported from Word documents, they are inserted into the Media Store together with a folder, regardless of whether the same picture already exists or not. If any duplicate pictures are deleted from the Media Store, the reference in the relevant DOM input components in which these pictures are used must be adjusted manually.

6.7 Importing tables

Tables from Word documents can be imported including their formatting. Merges are imported by a ruleset without special configuration.

6.7.1 Tables from Word documents

The following area is marked in the Word test document and is copied onto the clipboard (<CTRL> + <C>):


OFFI42EN_FirstSpirit_Word-Test-Document.doc - Microsoft Word
Datei Bearbeiten Ansicht Einfügen Format Extras Tabelle Fenster ?
Frage hier eingeben
100%
Lesen
Überschrift 3
Arial
13
Seite 3 Ab 1 3/3 Bei 8,5 cm Ze 8 Sp 9 MAK ÄND ERW ÜB Deutsch (De

Tables

Simple tables

This table only contains text. Its cells are numbered sequentially by column and row:

1.1	1.2	1.3	1.4
2.1	2.2	2.3	2.4
3.1	3.2	3.3	3.4

Formatted tables

This table contains formattings: coloured background of cells, text alignment and text formatting:

1.1	**1.2**	**1.3**	**1.4**	**1.5**
2.1	2.2 (left)	2.3 (right) RGB colour: 255, 102, 0	2.4 (centred)	2.5
3.1	3.2 RGB colour: 204, 255, 204	3.3	3.4	3.5 RGB colour: 0, 204, 255

Tables with merging and formatting

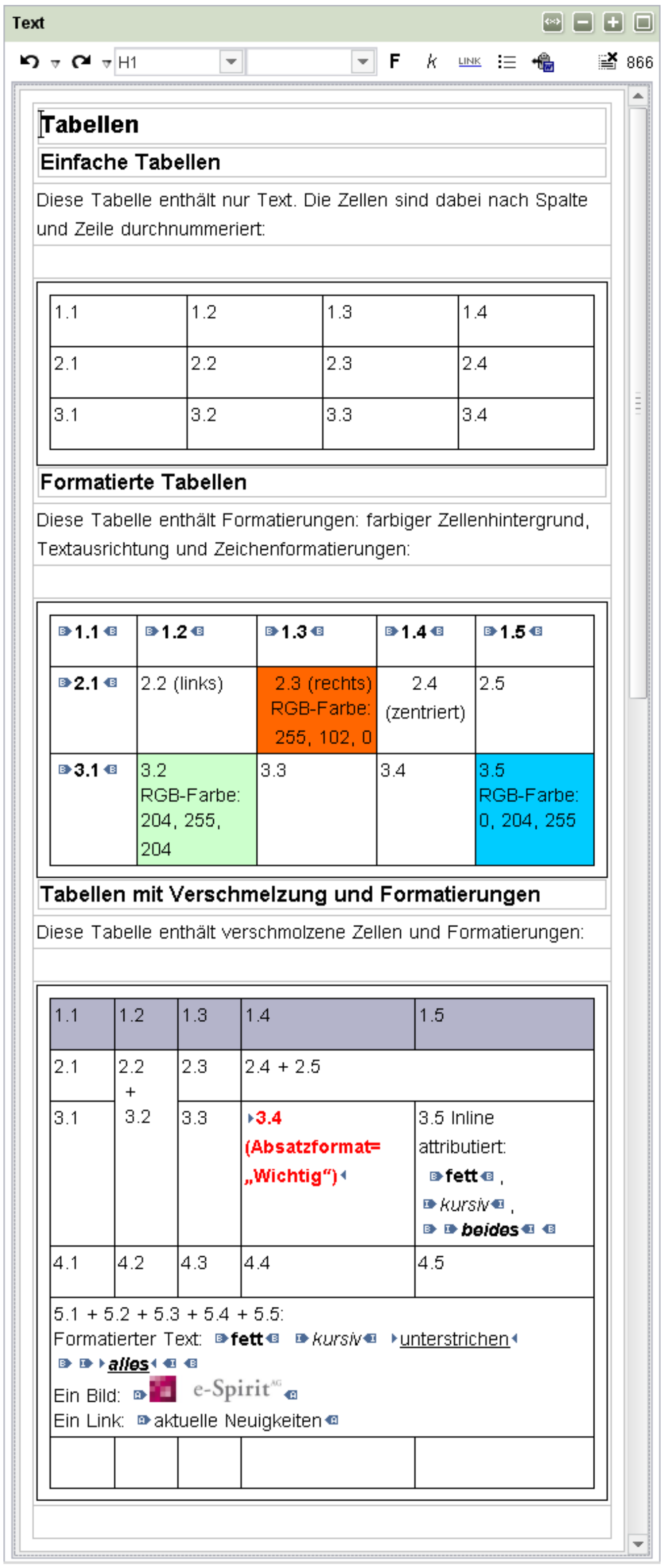
This table contains merged cells and formattings:

1.1	1.2	1.3	1.4	1.5
2.1	2.2 + 3.2	2.3	2.4 + 2.5	
3.1		3.3	**3.4 (Paragraph formatting= "Important")**	3.5 Inline formatted: **bold**, *italics*, ***both***
4.1	4.2	4.3	4.4	4.5
5.1 + 5.2 + 5.3 + 5.4 + 5.5: Formatted text: **bold** *italics* underlined ***all*** A picture: e-Spirit AG A link: latest news				

Figure 6-11: Word test document: Tables

6.7.2 Tables in the DOM Editor

After it has been imported using the Office function, the text with lists from the test document can be displayed in the DOM Editor as follows:



Tabellen

Einfache Tabellen

Diese Tabelle enthält nur Text. Die Zellen sind dabei nach Spalte und Zeile durchnummeriert:

1.1	1.2	1.3	1.4
2.1	2.2	2.3	2.4
3.1	3.2	3.3	3.4

Formatierte Tabellen

Diese Tabelle enthält Formatierungen: farbiger Zellenhintergrund, Textausrichtung und Zeichenformatierungen:

1.1	**1.2**	**1.3**	**1.4**	**1.5**
2.1	2.2 (links)	2.3 (rechts) RGB-Farbe: 255, 102, 0	2.4 (zentriert)	2.5
3.1	3.2 RGB-Farbe: 204, 255, 204	3.3	3.4	3.5 RGB-Farbe: 0, 204, 255

Tabellen mit Verschmelzung und Formatierungen

Diese Tabelle enthält verschmolzene Zellen und Formatierungen:

1.1	1.2	1.3	1.4	1.5
2.1	2.2 + 3.2	2.3	2.4 + 2.5	
3.1		3.3	**3.4 (Absatzformat= „Wichtig“)**	3.5 Inline attributiert: **fett** , *kursiv* , ***beides***
4.1	4.2	4.3	4.4	4.5
5.1 + 5.2 + 5.3 + 5.4 + 5.5: Formatierter Text: **fett** *kursiv* unterstrichen ***alles*** Ein Bild: e-Spirit Ein Link: aktuelle Neuigkeiten				

Figure 6-12: DOM Editor with tables from the Word document (DE)

7 Legal notices

The "FirstSpirit Office" module is a product of e-Spirit AG, Dortmund, Germany.

The user may only use the module as defined under the terms of the licence agreed with e-Spirit AG.

Details of possible external software products used, not produced by e-Spirit AG, their own licences and any update information, is given on the homepage of each FirstSpiritserver in the "Legal Notices" area.