FirstSpirit™
Your Content Integration Platform

FirstSpirit™ Office

FirstSpirit™ Version 4.2

Version	**1.2**
Status	**RELEASED**
Datum	**2009-07-14**
Abteilung	FS-Core
Autor/ Autoren	A. Pfeiler, B. Gutknecht
Copyright	2009 e-Spirit AG
Dateiname	OFFI42DE_FirstSpirit_Modules_Office.doc

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

Inhaltsverzeichnis

1 Einführung

Das Modul FirstSpirit™ Office ermöglicht es, Textpassagen aus Word-Dokumenten oder komplette Word-Dokumente in den FirstSpirit™ JavaClient zu übernehmen und dort weiterzubearbeiten. Dabei können neben reinem Text auch Formatierungen, Absatzschaltungen, Bilder, Verlinkungen und Tabellen übernommen werden.

Das Modul übersetzt die Informationen, die aus dem Word-Dokument stammen, in Informationen, die FirstSpirit™ verarbeiten und darstellen kann. Konkret werden dazu Meta-Infomationen aus dem Word-Dokument (z.B. Formatierungen) mit Format- und Verweisvorlagen in FirstSpirit™ verbunden. Dies ist möglich, da das Standard-Dokumentenaustauschformat für MS Office-Anwendungen HTML ist. Das Modul greift auf individuell definierbare Regelsätze zurück, die beinhalten, welche HTML-Information aus dem Word-Dokument welcher Format- bzw. Verweisvorlage in FirstSpirit™ zugeordnet ist.

1.1 Übersicht der Funktionen

Die Office-Funktionalität kann in den Eingabekomponenten

- DOM-Editor (CMS_INPUT_DOM) und
- DOM-Tabelle (CMS_INPUT_DOMTABLE)

eingesetzt werden.

Die gesamte vorliegende Dokumentation bezieht sich größtenteils auf beide Eingabekomponenten, wenn nichts anderes erwähnt ist. Alternativ werden sie im Folgenden auch als „DOM-Eingabekomponenten“ zusammengefasst.

Folgende Informationen aus Word-Dokumenten werden bei einer Übernahme nach FirstSpirit berücksichtigt:

- Fließtext
- Zeichenformatierungen (**fett**, *kursiv*, unterstrichen sowie Kombinationen)
- Absatzformatierungen (z.B. Überschriften)
- Absatzschaltungen und „weiche“ Zeilenumbrüche (<RETURN> bzw. <SHIFT> + < RETURN >)
- Aufzählungen (mit Nummern oder Symbolen)

- Bilder (inklusive automatischem Upload in die Medien-Verwaltung)
- Links
- Tabellen (inklusive Verschmelzung, Zellenfarbe, Zellenausrichtung sowie Zellenformat)

1.2 Thema dieser Dokumentation

Kapitel 2: Beschreibt die Installation des Moduls „FirstSpirit Office“ auf dem Server und die Installation der Projekt-Komponente in einem Projekt (ab Seite 10).

Kapitel 3: Erläutert die server- und projektweiten Konfigurationsmöglichkeiten, die das Modul bietet (ab Seite 12).

Kapitel 4: Beschreibt anhand von Beispielen die Syntax, die für die XML-Definition der Regelsätze, mit denen Inhalte aus Word-Dokumenten im DOM-Editor (Eingabekomponente CMS_INPUT_DOM) bzw. in der DOM-Table (Eingabekomponente CMS_INPUT_DOMTABLE) abgebildet werden können (ab Seite 16).

Kapitel 5: Damit das Modul im JavaClient verwendet werden kann, muss der DOM-Editor und / oder die DOM-Table vom Vorlagenentwickler entsprechend angepasst werden (ab Seite 43).

Kapitel 6: Dieses Kapitel ist in erster Linie für FirstSpirit-Redakteure gedacht und veranschaulicht, wie das Modul im JavaClient redaktionell verwendet wird. Dazu wird beispielhaft das mitgelieferte Word-Test-Dokument schrittweise importiert und die Auswirkung im DOM-Editor bzw. in der DOM-Tabelle sowie in der HTML-Ausgabe dargestellt. Aber auch für die Entwicklung der XML-Regelsätze ist dieses Kapitel interessant (ab Seite 49).

1.3 Vorgehen bei der Einführung des Moduls

1.3.1 Nutzung der mitgelieferten Regelsätze und Vorlagen

In der Standard-Konfiguration des Office-Moduls sind bereits drei Regelsätze im Lieferumfang enthalten, die direkt verwendet werden können:

- „Project local Import Ruleset“: Regelsatz für die Umformung von Inhalten aus Word-Dokumenten mit „statischen“ Links (für Projekte, die mit statischen Links arbeiten, bis FirstSpirit Version 4.2 einschließlich)
- „Project local Import Ruleset (generic link)“: Regelsatz für die Umformung von

Inhalten aus Word-Dokumenten mit „generischen“ Links (für Projekte, die mit „generischen“ Links arbeiten, für weitere Informationen siehe FirstSpirit Online Dokumentation, Kapitel „Vorlagenentwicklung“ / „Verweisvorlagen“ / „Generische Link-Editoren“)

- „Standard (text only import)“: Regelsatz für Inhalte, die nicht umgeformt werden sollen

Mit den im globalen Regelsatz definierten Regeln ist es möglich, das ebenfalls erhältliche Word-Test-Dokument mit allen Formatierungen, Links und Bildern in den DOM-Editor bzw. die DOM-Table zu importieren.

Zur Abbildung der Formatierungen aus Word wird dabei einerseits auf die im Standard-Lieferumfang von FirstSpirit enthaltenen Formatvorlagen zurückgegriffen, z.B. **fett** (Formatvorlagen-Kürzel „b“), kursiv (Formatvorlagen-Kürzel „i“) oder unterstrichen (Formatvorlagen-Kürzel „u“).

Andererseits werden darüber hinaus erforderliche Formatvorlagen mit folgenden Eigenschaften in einem Demo-Projekt enthalten sein. Dies liegt aktuell (Stand 07/2009) noch nicht vor.

- **H1-5**: Absatzformatvorlagen für Überschriften erster bis fünfter Ebene:
  - Kürzel: „h1“ bis „h5“
  - Stil: „Fett“
  - Größen: "20", „18“, „16“, „15“, „13“
  - Konvertierungsregel: „Unicode to HTML4"
  - HTML-Kanal: `<h1>$CMS_VALUE(#content)$</h1>`
- **Note**: Absatzformatvorlage für Hinweise in roter Schrift:
  - Kürzel: „note“
  - Farbe: „#FF0000"
  - Konvertierungsregel: „Unicode to HTML4"
  - HTML-Kanal: `<font color="red">$CMS_VALUE(#content)$ </font>`
- **Important**:Zeichenformatvorlage für Hinweise in roter Schrift:
  - Kürzel: „important“
  - Farbe: „#FF0000"
  - Konvertierungsregel: „Unicode to HTML4"
  - HTML-Kanal: `<font color="red">$CMS_VALUE(#content)$ </font>`
- **s**: Zeichenformatvorlage für Durchstreichungen:
  - Kürzel: „s“
  - Stil: „Durchgestrichen"
  - Konvertierungsregel: „Unicode to HTML4"

- HTML-Kanal: `<s>$CMS_VALUE(#content)$</s>`

Zudem werden für den Import von Links und Bildern spezielle **Verweisvorlagen** benötigt. Die für den Import des Word-Test-Dokuments erforderlichen befinden sich im Demo-Projekt „Mithras Energy“ unterhalb des Knotens „Verweisvorlagen“ in der Vorlagen-Verwaltung, unterhalb der Verweiskonfigurationen „Fließtext Link (extern)“ und „Fließtext Link (intern)“:

- **textlinkinternal.standard**: Verweisvorlage zum Abbilden von importierten Bildern
- **textlinkexternal.standard**: Verweisvorlage zum Abbilden von externen Links

Hinweis zur Verwendung von generischen Links: Wurden diese Verweisvorlage zu „generischen Links“ konvertiert (Kontextmenü „Extras“ / „Verweisvorlage konvertieren“), lauten die eindeutigen Namen **textlinkinternal** und **textlinkexternal**. In diesem Fall muss beim Import in den DOM-Editor (siehe Kapitel 6.2 Seite 51 und Abbildung 6-2) der Standard-Regelsatz „Project local Import Ruleset (generic link)“ ausgewählt werden.

Das Erscheinungsbild der verwendeten Format- und Verweisvorlagen im JavaClient und bei der Ausgabe auf der Website kann an die Erfordernisse des jeweiligen Projektes angepasst werden. Weiterführende Informationen zu den Konfigurationsmöglichkeiten siehe FirstSpirit Online Dokumentation, Bereich „Vorlagenentwicklung“, Kapitel „Formatvorlagen“ bzw. „Verweisvorlagen“.

Weitere Informationen zum Word-Dokument für den Test-Import siehe Kapitel 6.1 Seite 49.

1.3.2 Erstellen eigener Regelsätze

Für die Definition eigener Regeln bzw. Regelsätze ist folgendes Vorgehen empfehlenswert:

1. Zunächst ist es notwendig, die zu importierenden Word-Dokumente zu analysieren und eine Liste der verwendeten Absatz- und Zeichenformate zu erstellen.
2. In einem zweiten Schritt sollte überprüft werden, ob in diesen Word-Dokumenten die Absatz- und Zeichenvorlagen auch konsequent verwenden werden. Je nach individueller Konfiguration von Word und persönlicher Arbeitsweise können

verschiedene Formatvorlagen verwendet werden, obwohl Layout und Funktion der Vorlagen gleich sind. Für ein möglichst gutes Ergebnis beim Word-Import sollten zunächst die Formatvorlagen in Word konsolidiert werden, denn je weniger unterschiedliche Formatvorlagen in einem Dokument vorhanden sind, desto einfacher können die Regelsätze für den späteren Import erstellt werden.

3. Für jedes Absatz- und Zeichenformat, das im Word-Dokument enthalten ist, sollte eine entsprechende FirstSpirit-Formatvorlage vorhanden sein. Falls noch nicht vorhanden, sollten diese Vorlagen in FirstSpirit angelegt werden. Dabei wird im Register „Eigenschaften“ festgelegt, wie Texte mit dieser Formatierung im DOM-Editor angezeigt werden, im Register „HTML“ (bzw. Register für andere Ausgabekanäle) wie Texte mit dieser Formatierung auf der Webseite (bzw. in anderen Ausgabe-Medien) ausgegeben werden. Ggf. empfiehlt sich die Aufstellung einer Tabelle wie in Kapitel 6.1 Seite 49.
   Darüber hinaus muss für den Import von Bildern eine interne Verweisvorlage vorhanden sein, die eine Eingabekomponente für die Auswahl eines Medienobjekts enthält (`<CMS_PARAM name="mediaref" hidden="0"/>`). Für den Import von Links muss eine externe Verweisvorlage vorhanden sein.
4. Durch einen HTML-Export des Word-Dokuments können die HTML-Informationen, die für einen Import nach FirstSpirit umgeformt werden müssen, ermittelt werden. Aufgrund dieser Informationen muss dann die XML-Regeldefinition erstellt werden (siehe Kapitel 4 Seite 16, insbesondere Kapitel 4.6 Seite 37).
5. Anschließend müssen die FirstSpirit-Eingabekomponenten, die für den Import von Word-Dokumenten verwendet werden sollen (DOM-Editor, DOM-Tabelle), für die Office-Funktionalität konfiguriert werden. Dazu müssen u.a. die im 4. Schritt angelegten Formatvorlagen in die Eingabekomponenten eingebunden werden, in denen die Office-Funktionalität verfügbar sein soll (siehe Kapitel 5.3 Seite 45).

Danach kann die eigentliche Datenübernahme erfolgen (siehe dazu Kapitel 6 Seite 49).

2 Installation

2.1 Installation des Moduls auf dem FirstSpirit-Server

Das Modul „FirstSpirit Office“ ist eine lizenzabhängige Funktionalität. Es muss zunächst in der FirstSpirit Server- und Projektkonfiguration installiert werden. In den Servereigenschaften wird dazu der Menüeintrag „Module“ ausgewählt. Mit einem Klick auf den Button „Installieren“ öffnet sich ein Dateiauswahldialog. Hier kann die zu installierende fsm-Datei (Dateiname „fs-office.fsm“) ausgewählt werden. Die erfolgreich installierte Modul-Datei wird anschließend in der Liste der installierten FirstSpirit-Module mit dem Namen „FirstSpirit Office“ angezeigt:

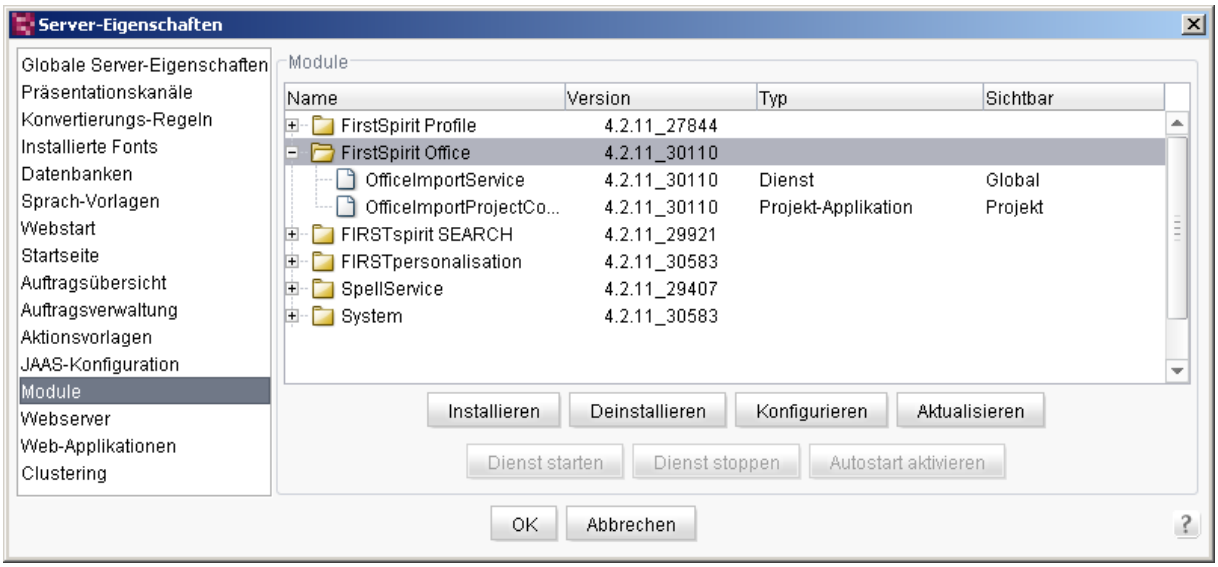

Abbildung 2-1: Installation des Moduls „FirstSpirit Office“

Bestandteile des Moduls sind der Dienst „OfficeImportService“ sowie die Projekt-Applikation „OfficeImportProjectConfiguration“. Weitere Informationen zu diesem Dialog siehe *FirstSpirit Handbuch für Administratoren*.

Der **Dienst** „OfficeImportService“ ermöglicht eine zentrale Definition von Regeln, nach denen Word-Dokumente übernommen werden. Er ist global, d.h. serverweit, in jedem auf dem FirstSpirit-Server befindlichen Projekt verfügbar. Nach dem Starten des Dienstes kann die Office-Funktionalität für den gesamten Server konfiguriert werden (siehe Kapitel 3.1 Seite 12).

Bei der **Projekt-Applikation** „OfficeImportProjectConfiguration“ handelt es sich um eine „projekt-lokale“ Komponente. Diese kann nach der Installation den

gewünschten Projekten über deren Projekteigenschaften hinzugefügt (siehe Kapitel 2.2 Seite 11) und die Office-Funktionalität projektspezifisch konfiguriert (siehe Kapitel 3.3 Seite 15) werden.

2.2 Installation der Projekt-Komponente

Sollen in einem Projekt andere oder erweiternde Regeln verwendet werden, muss die Projekt-Komponente „OfficeImportProjectConfiguration“ im gewünschten Projekt installiert werden. Dazu wird innerhalb der Projekteigenschaften der Menüeintrag „Projekt-Komponenten“ aufgerufen.

Abbildung 2-2: Installation der Projekt-Komponente

Hinzufügen: Mit einem Klick auf den Button öffnet sich der Dialog „Hinzufügen“. In der Liste werden alle Projekt-Komponenten angezeigt, die auf dem Server installiert sind (siehe Kapitel 2.1 Seite 10). Wählen Sie den Eintrag „OfficeImportProjectConfiguration“. Danach stehen die global konfigurierten Funktionalitäten der installierten Komponente im Projekt zur Verfügung und können weiter konfiguriert werden (siehe Kapitel 3.3 Seite 15).

Wird die Projekt-Komponente nicht hinzugefügt, stehen im JavaClient nur die globalen Regeln zur Verfügung.

Weitere Informationen zu diesem Dialog siehe *FirstSpirit Handbuch für Administratoren*.

3 Konfiguration des Moduls

Die folgenden drei Möglichkeiten stehen zur Konfiguration des Moduls zur Verfügung:

- Um einen oder mehrere Regelsätze global für **alle** FirstSpirit-Projekte eines Servers zu definieren, braucht nur der Dienst „OfficeImportService“ entsprechend konfiguriert werden (siehe Kapitel 3.2 Seite 14).
- Um einen oder mehrere individuelle Regelsätze für **ein** FirstSpirit-Projekt – unabhängig von den globalen, serverweit geltenden Regelsätzen – zu definieren, muss die Projekt-Komponente „OfficeImportProject Configuration“ entsprechend konfiguriert werden. Die Option „Erweitern der globalen Regelsatz-Konfiguration“ muss dabei deaktiviert sein (siehe Kapitel 3.3 Seite 15).
- Um für **ein** FirstSpirit-Projekt die globalen, serverweit definierten Regelsätze zu nutzen und diese zusätzlich um individuelle Regelsätze zu erweitern, muss die für Projekt-Komponente „OfficeImportProject Configuration“ entsprechend konfiguriert werden. Die Option „Erweitern der globalen Regelsatz-Konfiguration“ muss dabei aktiviert sein (siehe Kapitel 3.3 Seite 15).

3.1 Lizenzdatei prüfen

Über das Menü „FirstSpirit – Konfiguration – Lizenz“ des FirstSpirit Server-Monitorings können die gültigen FirstSpirit-Funktionen der Lizenzdatei `fs-license.conf` angezeigt werden. Der Parameter `license.OFFICE_IMPORT` muss für die Verwendung der Office-Funktionalität auf den Wert `1` gesetzt sein (siehe Abbildung 3-1).

Eine gültige Lizenz kann beim Hersteller angefordert und wird im blauen Fensterbereich eingefügt. Mit einem Klick auf den Button Speichern kann die neue Lizenzdatei gespeichert werden.

Manipulationen an der `fs_license.conf` führen zu einer ungültigen Lizenz. Sollten Änderungen notwendig werden, wenden Sie sich bitte an den Hersteller.

Beim Einfügen einer neuen Konfigurationsdatei `fs_license.conf` ist kein Neustart des Servers erforderlich. Die Datei wird automatisch auf dem Server aktualisiert.

Lizenz

```
license.ID=365
#FIRSTspirit license
#Mon Jun 15 11:43:18 CEST 2009
license.USER=e-spirit
license.EXPDATE=15.01.2010
license.MAXPROJECTS=0
license.MAXSESSIONS=0
license.MAXUSER=0
license.SOCKET_PORT=0
license.VERSION=4
license.MODULES=integration,personalisation,portal,search
license.WEBEDIT=1
license.WORKFLOW=1
license.REMOTEPROJECT=1
license.PACKAGEPOOL=1
license.DOCUMENTGROUP=1
license.CLUSTERING=1
license.ENTERPRISE_BACKUP=1
license.OFFICE_IMPORT=1
```

Abbildung 3-1: Anzeige der Parameter der Lizenzdatei (Server Monitoring)

3.2 Konfiguration des Dienstes „OfficeImportService“

Mit einem Klick auf den Dienst „OfficeImportService“ und den Button Konfigurieren öffnet sich der Konfigurationsdialog des Dienstes:

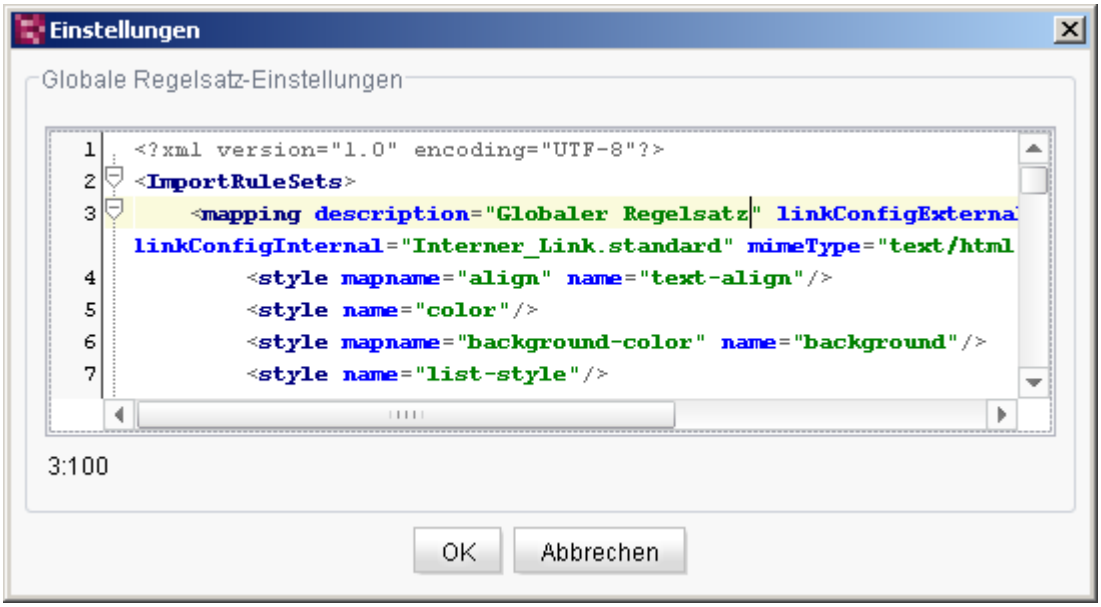

Abbildung 3-2: Konfiguration des Dienstes

Um den Dienst konfigurieren zu können, muss er zuvor gestartet worden sein. Ist dies nicht der Fall, erscheint die Meldung „Service starten?“ Mit einem Klick auf „Ja“ wird der Dienst gestartet und es erscheint der Konfigurations-Dialog (siehe Abbildung 3-2).

Über dieses Textfeld können globale Regeln für den Import von Texten aus Word-Dokumenten angegeben werden, die serverweit gelten. Diese Regeln definieren, wie die Formatierungen aus dem Word-Dokument später in FirstSpirit-Ausdrücke umgewandelt werden müssen. Detaillierte Informationen zur XML-Definition der Regelsätze siehe Kapitel 4 Seite 16.

Es können verschiedene Regelsätze für verschiedene Einsatzzwecke oder Dokumenttypen angegeben werden. So kann es beispielsweise einen Regelsatz für technische Dokumente und einen für Dokumente aus dem Marketing geben. Name (Attribut „name“) und Beschreibung (Attribut „description“) der Regelsätze werden dem Redakteur später beim Arbeiten mit dem Modul im JavaClient angezeigt und zur Auswahl angeboten.

3.3 Konfiguration der Projekt-Komponente „OfficeImportProject Configuration“

Bevor die Projekt-Komponente „OfficeImportProject Configuration“ konfiguriert werden kann, muss der Dienst (siehe Kapitel 2.1 Seite 10) gestartet sein.

Mit einem Klick auf die Projekt-Komponente „OfficeImportProjectConfiguration“ und den Button Konfigurieren öffnet sich der Konfigurationsdialog der Projekt-Komponente:

Abbildung 3-3: Konfiguration der Projekt-Komponente

Über diesen Dialog können spezielle Import-Regeln für das jeweilige Projekt konfiguriert werden. Auch hier können in das Textfeld verschiedene Regelsätze für verschiedene Einsatzzwecke definiert werden (siehe Kapitel 4 Seite 16).

Erweitern der globalen Regelsatz-Konfiguration: Über diese Option kann definiert werden, ob die serverweit konfigurierten Regeln (siehe Kapitel 3.1 Seite 12) durch die hier angegebenen Regeln ersetzt oder ob projektspezifische Regelsätze ergänzt werden sollen. Ist die Option aktiviert, werden dem Redakteur später bei der Arbeit mit dem Modul im JavaClient zusätzlich zu den serverweit definierten Regeln auch die hier definierten, projektspezifischen Regeln zur Auswahl angeboten. Ist die Option deaktiviert, stehen die serverweit definierten Regeln dem Redakteur nicht zur Verfügung, sondern nur die projektspezifischen.

4 XML-Definition der Regelsätze

Werden Inhalte aus Word-Dokumenten mittels <STRG> + <C> in die Zwischenablage übertragen, werden diese überwiegend in reinem HTML-Format abgelegt. Darüber hinaus sind in der Zwischenablage aber noch zahlreiche Word-spezifische Informationen enthalten, die für den Import nach FirstSpirit nicht verwendet werden können. Diese HTML-Informationen können z.B. angezeigt werden, indem das betreffende Word-Dokument als html-Datei abgespeichert und dann in einem Browser mit der Option „Seitenquelltext anzeigen“ geöffnet wird.

Die Regelsätze, die in FirstSpirit Office verwendet werden, sind dazu da, die HTML-Informationen aus der Zwischenablage (z.B. <p> für Absätze, <b> für fett, <h1> für Überschriften erster Ebene, <td> für Tabellenzellen, <img src=...> für Bilder, <a href=...> für Links usw.) mit den Zeichen- und Absatzformatvorlagen sowie Verweisvorlagen (für Bilder und Links) in FirstSpirit zu verknüpfen. Dadurch wird dann beispielsweise eine Zeichenformatierung aus Word für „fett“ in die Zeichenformatierung „fett“ des FirstSpirit-DOM-Editors „übersetzt“. Daraus folgt einerseits, dass gefetteter Text aus einem Word-Dokument im DOM-Editor ebenfalls als fett formatiert und gefettet dargestellt wird, so wie es auf dem Register „Eigenschaften“ der betreffenden Formatvorlage definiert ist. Über die FirstSpirit-Formatvorlage ist wiederum definiert, wie die Zeichenformatierung „fett“ (z.B. im HTML-Kanal) ausgegeben werden soll (Register „HTML“ der betreffenden Formatvorlage).

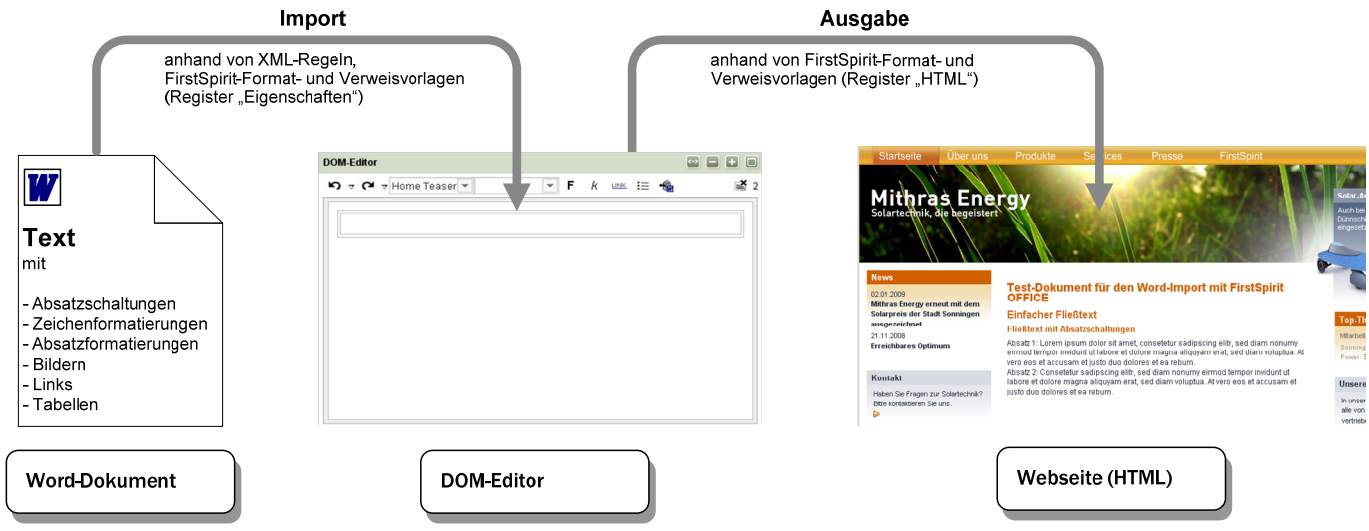

Abbildung 4-1: Schema Import – Ausgabe

Die Konfiguration der Regelsätze in FirstSpirit Office erfolgt in XML-Format.

4.1 XML-Syntax

XML-Dokumente müssen wohlgeformt und gültig sein, damit sie von einem Parser ausgelesen und interpretiert werden können. Folgende Regeln sind dabei zu beachten:

- **XML-Deklaration**: Zu Beginn der XML-Definition muss eine XML-Deklaration angegeben werden. Beispiel:

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

- **Aufbau**: Für gültiges und wohlgeformtes XML müssen alle öffnenden Tags auch wieder geschlossen werden.
- **Kommentare**: Kommentare können mit `<!-- ... -->` definiert werden. Alles, was innerhalb der spitzen Klammern steht, wird beim Import nicht berücksichtigt.

Beim Speichern von nicht wohlgeformtem XML wird ein entsprechender Fehlerdialog angezeig und das Dokument kann nicht gespeichert werdent.

4.2 Aufbau, Schachtelung und Vererbung

Das Grundgerüst der XML-Regeldefinition besteht aus

- der XML-Deklaration (siehe Kapitel 4.1 Seite 17) und
- dem Wurzelelement (siehe Kapitel 4.3.1 Seite 20).

Es folgt ein `mapping`-Element für jeden Regelsatz (siehe Kapitel 4.3.2 Seite 20). Innerhalb dieses Elements muss zu jeder Information, die aus dem Word-Dokument stammt und eine spezielle Formatierung hat, ein XML-Tag angegeben werden (siehe Kapitel 4.3.3 bis 4.3.6 ab Seite 23). Dies können sein:

- `style` (siehe Kapitel 4.3.3 Seite 23)
- `element` (siehe Kapitel 4.3.4 Seite 24)
- `attribute` (siehe Kapitel 4.3.5 Seite 26)
- `text` (siehe Kapitel 4.3.6 Seite 27)

Alle Elemente (bis auf das Wurzelelement) müssen mit Attributen versehen werden, und zwar mit folgenden:

zum Tag `mapping`: (siehe Kapitel 4.3.2 Seite 20)

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

zum Tag `style`: (siehe Kapitel 4.3.3 Seite 23)

- `mapname`
- `name`

zum Tag `element`:

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

zum Tag `attribute`:

- `mapname` (siehe Kapitel 4.3.5 Seite 26)
- `name` (siehe Kapitel 4.3.5 Seite 26)
- `handler` (siehe Kapitel 4.4.3 Seite 29)

zum Tag `text`: (siehe Kapitel 4.3.6 Seite 27)

- `action`

Die im folgendenen angegebenen Beispiele sind an die mitgelieferten Regelsätze der Standard-Konfiguration des Moduls angelehnt.

Die grundlegende Zuordnung von HTML-Informationen aus dem Word-Dokument zu Format- und Verweisvorlagen in FirstSpirit wird durch `element`-Tags definiert (siehe Kapitel 4.3.4 Seite 24). Die `element`-Tags enthalten wiederum Tags (`attribute`, `text` oder auch weitere `element`-Tags), die Umformungsregeln für untergeordnete Elemente festlegen. Sie können prinzipiell beliebig tief verschachtelt werden. Für eine bessere Übersicht sollte die Schachtelungstiefe aber gering und der XML-Code möglichst „schlank“ gehalten werden.

Untergeordnete `element`-Tags erben die Attribute des übergeordneten `element`-Tags sowie `attribute`- und `text`-Tags. Diese Vererbung kann durch das Attiribut `inherit="NONE"` (siehe Kapitel 4.4.9 Seite 34) unterbunden werden. Ein `element`-Tag kann auch die Attribute eines anderen `element`-Tags erben, das nicht übergeordnet ist. Dazu muss für das `element`-Tag, von dem Attribute übernommen werden sollen, mithilfe von `id="..."` ein Name vergeben werden. Auf diesen Namen kann das `element`-Tag, das die Attribute erben soll, über `inherit="ID_DES_VERERBENDEN_ELEMENTS"` zugegriffen werden. Dies ist z.B. für Tabellenzellen relevant, deren Inhalte genauso umgeformt werden sollen, wie Fließtext-Inhalte.

Beispiel:

```
<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 diesem Beispiel werden die Attribute der ersten Zeile für den HTML-Tag `p` eines Fließtext-Absatzes über die `id` „HTML.paragraph“ auch für Tabellenzellen (HTML-Tag `td`) und Absätze innerhalb einer Tabellenzelle (HTML-Tag `p` innerhalb der Tabelle, die mit `tag="table"` eingeleitet wird) verwendet.

Der grobe Aufbau des XML-Regelsatzes lehnt sich an den Aufbau der HTML-Datei aus der Word-Zwischenablage an und umfasst damit unter anderem folgende Elemente:

- Head
- Body
- Absätze (<p>)
  - Bilder
  - Umbrüche (
)
  - Zeichen- und Absatzformatierungen
  - Links
  - Aufzählungen
  - Tabellen
    - Tabellenzeilen
    - Tabellenzellen
    - Kopfzeilen (<th>)

Dies ist auch das Schema des mitgelieferten Beispiel-Regelsatzes. Die Einrückung unterhalb der Regeln für Absätze bedeuten, dass alle eingerückten Elemente (Bilder, Tabellen usw.) die Regeln für Absätze erben.

4.3 Tags

4.3.1 ImportRuleSets

Die XML-Definition muss vom Wurzelelement

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

umschlossen sein.

Das Tag `ImportRuleSet` benötigt keine Attribute.

4.3.2 mapping

Innerhalb des Wurzelelements können mehrere Regelsätze enthalten sein. Jeder

Regelsatz wird mit einem Mapping-Element

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

eingeleitet.

Für jedes Mapping-Element können verschiedene Attribute angegeben werden:

Attribut	Erläuterung / Beispiel	Pflicht-attribut
`description`	Angabe einer Beschreibung für den Regelsatz; diese wird dem Redakteur in der DOM-Eingabekomponente bei der Arbeit mit dem Office-Modul angezeigt (siehe Abbildung 6-2), z.B. `description="Project local import ruleset definition"`	ja
`linkConfigExternal`	Angabe des eindeutigen Namens der Verweisvorlage, die beim Import von Word-Dokumenten zur Auflösung von externen Links verwendet werden soll. Diese Verweisvorlage muss auch in der jeweiligen DOM-Eingabekomponente über den Tag *LINKEDITOR* eingebunden sein, z.B. `linkConfigExternal="textlinkexternal.standard"` Dieses Attribut braucht nicht angegeben werden, wenn keine Links beim Import aus Word-Dokumenten umgeformt werden sollen.	nein
`linkConfigInternal`	Angabe des eindeutigen Namens der Verweisvorlage, die beim Import von Word-Dokumenten zur Auflösung von Medien verwendet werden soll. Diese Verweisvorlage muss auch in der jeweiligen DOM-Eingabekomponente über den Tag *LINKEDITOR* eingebunden sein, z.B. `linkConfigInternal="textlinkinternal.standard"` Dieses Attribut braucht nicht angegeben werden, wenn keine Medien importiert werden sollen.	nein

Attribut	Erläuterung / Beispiel	Pflicht-attribut
`mimeType`	Angabe des MIME-Typs, z.B. `mimeType="text/html"` Der mitgelieferte Regelsatz „Standard (text only import)“ ist für den Import von Texten vorgesehen, die beim Import bei installiertem Office-Modul nicht umgeformt werden sollen. Dieser hat daher den MIME-Typ `mimeType="text/plain"`.	ja
`name`	Angabe eines eindeutigen Namens für den Regelsatz; dieser wird dem Redakteur im DOM-Editor bei der Arbeit mit dem Office-Modul angezeigt (siehe Abbildung 6-2), z.B. `name="Project local Import Ruleset"`	ja
`versionTag`	Dieses Attribut gibt die Versionsnummer des XML-Regelsatzes an. Sie wird mit jeder Änderung automatisch um eins erhöht, z.B. `versionTag="24"` Bei jedem Import wird anhand dieser Versionsnummer zunächst geprüft, ob seit dem letzten Import Änderungen an den Regelsätzen vorgenommen worden sind. Ist dies nicht der Fall, werden die Regelsätze, die sich noch im Client-Cache befinden, direkt angewendet. Haben sich in der Zwischenzeit Änderungen ergeben, werden die geänderten Regelsätze zunächst geladen und dann angewendet.	automa-tisch

Das Mapping-Element muss am Ende der XML-Definition wieder geschlossen werden.

Beispiel für mehrere Regelsätze:

```
<?xml version="1.0" encoding="UTF-8"?>
  <ImportRuleSets>
    <mapping description="Projektspezifischer Regelsatz"
```

```
        linkConfigExternal="textlinkexternal.standard"
        linkConfigInternal="textlinkinternal.standard"
        mimeType="text/html" name="Projekt" versionTag="24">
      ...
    </mapping>

    <mapping description="Projektspezifischer Regelsatz für
        Plain-Text"
        linkConfigExternal="textlinkexternal.standard"
        linkConfigInternal="textlinkinternal.standard"
        mimeType="text/html" name="Text" versionTag="24">
      ...
    </mapping>
  </ImportRuleSets>
```

Nach dem Wurzel- und dem Mapping-Element wird für jede Information, die aus einem Word-Dokument in die Zwischenablage übertragen wird, eine Zuordnungsregel definiert, nach der die Information in FirstSpirit abgebildet werden soll.

4.3.3 style

Über das `style`-Tag wird die Umformung von „style“-Tags aus dem HTML des Word-Dokuments festgelegt. Die Umformungsregeln, die mithilfe des `style`-Tags definiert werden, gelten global für das gesamte Word-Dokument.

Die Umformung der „style“-Tags aus dem Word-Dokument in Informationen, die in FirstSpirit ausgewertet werden können, wird innerhalb eines `style`-Tags über die Attribute `mapname` und `name` hergestellt: mit `name` werden die Werte der „style“-Tags aus dem Word-Dokument angegeben, die beim Import nach FirstSpirit umgeformt werden sollen, mit `mapname` wird der FirstSpirit-Wert angegeben, in den das HTML-Attribut übersetzt werden soll. Beide Attribute müssen in einem `style`-Tag vorhanden sein, z.B.

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

In diesem Beispiel werden Informationen aus dem Word-Dokument, die mit dem „style“-Tag `text-align` ausgezeichnet sind, beim Import in eine DOM-Eingabekomponente mit dem FirstSpirit-Attribut `align` verknüpft. Damit werden Textausrichtungen in Word wie z.B. `right` (rechtsbündig) oder `center` (zentriert) beim Import in eine DOM-Eingabekomponente auf `align` abgebildet, das zur Steuerung der Textausrichtung in FirstSpirit dient.

Für jedes „style“-Tag aus dem Word-Dokument, das in FirstSpirit ausgewertet werden soll, muss im XML-Regelsatz ein `style`-Tag definiert werden.

Wird `mapname` nicht explizit angegeben, wird die Bezeichnung, die über das Attribut `name` angegeben ist, verwendet. Beispiel:

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

In diesem Beispiel wird als `mapname` der Wert „list-style“, der für `name` definiert ist, verwendet. Dies ist also eine alternative, verkürzte Schreibweise für

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

Attribut	Erläuterung / Beispiel	Pflicht-attribut
`name`	Angabe von „style“-Tags aus der Word-Zwischenablage, z.B. `name="text-align"`	ja
`mapname`	Angabe eines Wertes in FirstSpirit, in den das `name`-Attribut übersetzt werden soll, z.B. `mapname="align"` Sind die Werte des „style“-Tags aus dem Word-Dokument und der entsprechende Wert in FirstSpirit identisch, braucht `mapname` nicht angegeben zu werden.	nein

Für das `style`-Tag müssen keine weiteren Attribute angegeben werden.

4.3.4 element

Über das Tag `element` werden die Umformungsregeln für die HTML-Elemente des Word-Dokuments definiert, z.B. Elemente für

- das HTML-Grundgerüst: `html`, `head`, `body`
- Textformatierungen: `p`, `div`, `span`, `br`, `b`, `strong`, `I`, `h1`, `li`, `ol`, `ul`
- die Definition von Links: `a`
- die Definition von Tabellen: `table`
- das Einbinden von Grafiken: `img`

Für jedes HTML-Element, das im Word-Dokument enthalten ist, **muss** in der XML-Definition ein `element`-Tag angegeben werden. Wird kein `element`-Tag angegeben, wird das Default-Element verwendet (siehe Kapitel 4.4.4 Seite 31).

Folgende **Pflicht-Attribute** müssen für das `element`-Tag angegeben werden:

- `tag` / `maptag`: Über diese Attribute wird die Verknüpfung zwischen der HTML-Information aus dem Word-Dokument und den FirstSpirit-Format- und Verweisvorlagen hergestellt: mit `tag` werden die Werte der „style"-Tags aus dem Word-Dokument angegeben, die beim Import nach FirstSpirit umgeformt werden sollen, mit `maptag` wird der FirstSpirit-Wert (Kürzel der FirstSpirit-Formatvorlage oder andere Werte, die intern zur Steuerung von FirstSpirit verwendet werden) angegeben, in den das HTML-Element übersetzt werden soll (siehe dazu auch Kapitel 4.4.1 Seite 28 und Kapitel 4.4.2 Seite 29). `maptag` braucht nicht angegeben zu werden, wenn dieselbe Bezeichnung verwendet wird, die für das Attribut `tag` angegeben ist.
- `handler`: Über dieses Attribut wird definiert, wie die HTML-Informationen aus dem Word-Dokument weiter verarbeitet werden. Es wird auch für den Import von Bildern, Links und Tabellen benötigt (siehe Kapitel 4.4.3 Seite 29).

Folgende **optionalen Attribute** können mit `element` verwendet werden:

- `default`: (auch „Default-Element") Definition von Umformungsregeln für unbekannte Informationen aus Word-Dokumenten (siehe Kapitel 4.4.4 Seite 31)
- `id` Definition von Namen für `element`-Tags, um an anderer Stelle auf diese referenzieren zu können (siehe Kapitel 4.4.5 Seite 32)
- `findtag` Auslesen von Tags aus dem Word-Dokument (in Verbindung mit `handler=„find"`), z.B. des Titels des Word-Dokuments (siehe Kapitel 4.4.6 Seite 33)
- `content` Verarbeitung / Parsen von Objekten aus Word-Dokumenten, z.B. von Bildern (in Verbindung mit `handler=„media"`) (siehe Kapitel 4.4.7 Seite 33)
- `mediaref` Import von Bildern (in Verbindung mit `handler=„media"`) (siehe Kapitel 4.4.8 Seite 33)
- `inherit` Steuerung der Vererbung von Attributen (siehe Kapitel 4.4.9 Seite 34)
- `break` Umformung von Textumbrüchen, z.B. in Tabellenzellen (siehe Kapitel 4.4.10 Seite 35)
- `mapattributes` Verwenden von Attributen aus dem Word-Dokument, z.B.

Attribute in Tabellen
(siehe Kapitel 4.4.11 Seite 35)

- `class` Umformung von selbst-definierten Word-Formatvorlagen, die in der HTML-Zwischenablage mit „class“ attributiert werden (siehe Kapitel 4.4.12 Seite 36)

Beispiel:

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

Dieses `element`-Tag enthält die Zuordnung für Links: diese werden im HTML des Word-Dokuments mit dem Tag `a` ausgezeichnet. Im Regelsatz muss für Links `maptag="link"` angegeben werden. Intern wird `link` auf `CMS_LINK` abgebildet.

4.3.5 attribute

Über das Tag `attribute` werden die Umformungsregeln für die HTML-Attribute des Word-Dokuments definiert, z.B. für

- HTML-Stylesheet-Definitionen: `style`
- die Definition von Links: `href`
- die Definition von Tabellen: `colspan`, `rowspan`, `style`
- das Einbinden von Grafiken: `src`

Für jedes HTML-Stylesheet-Attribut, das im Word-Dokument enthalten ist, **muss** in der XML-Definition ein `attribute`-Tag angegeben werden. Es muss von `<element> ... </element>` umschlossen werden und erbt damit alle Attribute des übergeordneten Elements (siehe Kapitel 4.2 Seite 17).

Folgende **Pflicht-Attribute** müssen für das `attribute`-Tag angegeben werden:

- `name` / `mapname`: Über diese Attribute wird die Verknüpfung zwischen der HTML-Information aus dem Word-Dokument und den FirstSpirit-Format- und Verweisvorlagen hergestellt: mit `name` werden die Werte von HTML-Attributen aus dem Word-Dokument angegeben, die beim Import nach FirstSpirit umgeformt werden sollen, mit `mapname` wird der FirstSpirit-Wert (Kürzel der FirstSpirit-Formatvorlage oder andere Werte, die intern zur Steuerung von FirstSpirit verwendet werden) angegeben, in den das HTML-Attribut übersetzt werden soll. `mapname` braucht nicht angegeben zu werden, wenn dieselbe Bezeichnung verwendet wird, die für das Attribut `name` angegeben ist.

- `handler`: Über dieses Attribut wird definiert, wie die HTML-Informationen aus dem Word-Dokument weiter verarbeitet werden (siehe auch Kapitel 4.4.3 Seite 29). Zusätzlich kann `handler="style"` verwendet werden, um die Informationen aus dem `style`-Tag (siehe Kapitel 4.3.3 Seite 23) zu übernehmen.
  Beispiel: `<attribute handler="style" name="style"/>`

Beispiel **Links**:

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

Dieses `attribute`-Tag enthält die Zuordnung für das Attribut `href`, mit dem der URI eines Links angegeben wird. Im Regelsatz muss `mapname="target"` angegeben werden.

Weitere Informationen zum Importieren von Links siehe Kapitel 6.5 Seite 58.

Beispiel **Bilder**:

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

Über dieses `attribute`-Tag wird das zugehörige Bild in die Medien-Verwaltung übernommen. Im Regelsatz müsste `mapname="src"` angegeben werden, kann aber entfallen, da der Wert von `mapname` und `name` identisch ist.

Weitere Informationen zum Importieren von Bildern siehe Kapitel 6.5 Seite 58.

4.3.6 text

Über das Tag `text` kann definiert werden, wie Texte innerhalb von Elementen beim import in die DOM-Eingabkomponente verarbeitet werden soll.

Das `text`-Tag muss von `<element> ... </element>` umschlossen werden und erbt damit alle Attribute des übergeordneten Elements (siehe Kapitel 4.2 Seite 17).

Folgendes **Pflicht-Attribut** muss für das `text`-Tag angegeben werden:

- `action`: Über dieses Attribut wird definiert, wie Text beim Import in eine DOM-Eingabekomponente verarbeitet werden soll.

Folgende Werte können für `action` verwendet werden:

Attribut	Erläuterung / Beispiel	Pflicht-attribut
`ignore`	Text wird beim Import nicht in die Eingabekomponente übernommen, also ignoriert, z.B. `<text action="ignore"/>`	nein
`default`	Ist ein `default`-Element definiert (siehe Kapitel 4.4.4 Seite 31), wird der Text mit den über dieses Element definierten Attributen und Werten umgeformt, z.B. `<text action="default"/>` Zeilenumbrüche (` `) und Leerzeichen (` `) werden dabei entfernt und nicht mit in die DOM-Eingabekomponente übernommen.	nein
`keep`	Der Text wird in den DOM-Editor übernommen, z.B. `<text action="keep"/>` Dieses ist die Standard-Einstellung.	Default

4.4 Attribute

4.4.1 tag

Über das Attribut `tag` wird in `element`-Tags die Bezeichnung des HTML-Elements angegeben, z.B.

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

für nummerierte Listen.

Hinweise zur korrekten Ausgabe von geschachtelten Listen siehe auch Kapitel 5.4 Seite 46.

4.4.2 maptag

Über das Attribut `maptag` wird in `element`-Tags das Kürzel der FirstSpirit-Formatvorlage angegeben, die mit der jeweiligen Word-Vorlage verbunden werden soll, z.B.

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

für Überschriften erster Ebene im Word-Dokument.

Wird dieses Attribut nicht angegeben, wird automatisch dieselbe Bezeichnung verwendet, die für das Attribut `tag` (siehe Kapitel 4.4.1 Seite 28) angegeben ist. Sind die Format-Bezeichnungen in FirstSpirit und im Word-Dokument identisch, braucht das Attribut `maptag` nicht extra angegeben zu werden.

4.4.3 handler

Über das Attribut `handler` können verschiedene Optionen angegeben werden, wie mit dem jeweiligen HTML-Element beim Import in den DOM-Editor verfahren werden soll. Standardmäßig wird der Wert **map** verwendet. In diesem Fall wird das HTML-Element aus dem Word-Dokument nicht umgewandelt. Es kann durch die zusätzliche Angabe des Attributs `mapname` mit diesem Tag-Namen verbunden werden. Wird für ein HTML-Element das Attribut `handler` nicht definiert, wird automatisch `handler="map"` gesetzt.

Über den Wert **skip** kann das HTML-Element übersprungen, d.h. ignoriert werden, mit der Angabe von **strip** wird beim Import in den DOM-Editor das Tag des jeweiligen Elements entfernt (z.B. das Tag `<html>`), der Inhalt des Elements bleibt jedoch erhalten. Wird zusätzlich das Attribut `mapattributes="true"` angegeben, werden die Attribute des Elements auf das übergeordnete Element übertragen (siehe Kapitel 4.4.11 Seite 35).

Über den Wert **default** kann auf die Standard-Umfomungsregel zurückgegriffen werden (siehe Kapitel 4.4.4 Seite 31 und Kapitel 4.5 Seite 36).

Die Werte **object**, **media** und **table** werden zum Importieren von Links, Bildern bzw. Tabellen benötigt. Wie beim Wert **map** wird keine Umformung des jeweiligen HTML-

Elements vorgenommen. Folgende Tags und Attribute werden darüber hinaus für den Import dieser Objekte benötigt:

- für Links: `attribute`-Tag für `href` (siehe Kapitel 4.3.5 Seite 26)
- für Bilder: `content` (siehe Kapitel 4.4.7 Seite 33), `mediaref` (siehe Kapitel 4.4.8 Seite 33)
- für Tabellen: `element`- und `attribute`-Tags für die Umformung von
  - **Tabellenzeilen** (Word-HTML-Tag `tr`): z.B. `<element tag="tr">`
  - **Tabellenzellen** (Word-HTML-Tag `td`): z.B. `<element inherit="HTML.paragraph" tag="td">`
  - **Kopfzellen** (Word-HTML-Tag `th`): z.B. `<element inherit="HTML.tablecell" maptag="td" tag="th"/>`
  - **Verschmelzungen** (Word-HTML-Tags `colspan` und `rowspan`): z.B. `<attribute name="colspan"/>` und `<attribute name="rowspan"/>`

Wert	Erläuterung / Beispiel	Pflicht-angabe
`map`	beim Import wird keine Umformung des HTML-Elements vorgenommen, z.B. `<element default="true" handler="map" id="HTML.paragraph" tag="p">` Dieses ist die Standardeinstellung.	Default
`skip`	das HTML-Element wird beim Import übersprungen, z.B. `<element handler="skip" tag="*"/>`	nein
`strip`	das Tag des HTML-Elements wird entfernt und der Inhalt des HTML-Elements in die DOM-Eingabekomponente übernommen, z.B. `<element handler="strip" tag="html">` Wird gleichzeitig `mapattributes="true"` angegeben, werden die Attribute dieses Elements auf das übergeordnete Element übertragen.	nein

Wert	Erläuterung / Beispiel	Pflicht-angabe
`default`	Rückgriff auf das Default-Element, z.B. `<element handler="default" tag="*">`	nein
`object`	erforderlich für den Import von Links, z.B. `<element handler="object" maptag="link" tag="a">`	nein
`media`	erforderlich für den Import von Medien, z.B. `<element content="IGNORE" handler="media" mediaref="src" tag="img">`	nein
`table`	erforderlich für den Import von Tabellen, z.B. `<element handler="table" tag="table">`	nein
`find`	Auslesen von Tags aus dem Word-Dokument, z.B. `<element findtag="title" handler="find" tag="head"/>`	nein

4.4.4 default

Über das Attribut `default` kann ein Element als Standard-Umformungsregel für unbekannte Formatierungen definiert werden („Default-Element“). Wird für `default` der Wert „true“ gesetzt, wird die Umformung für alle Formatierungen oder Objekte aus dem Word-Dokument verwendet, für die kein Regelsatz hinterlegt ist, z.B.

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

Auf diese Regel wird dann mit `<element tag="*" handler="default">` zurückgegriffen. Mit dieser Definition werden alle „unbekannten“ HTML-Elemente aus dem Word-Dokument mit der Standard-Umformungsregel umgeformt (siehe Kapitel 4.5 Seite 36).

Im angegebenen Beispiel werden unbekannte Formatierungen und Auszeichnungen im Word-Dokument in einen Absatz (`<p> ... </p>`) überführt.

Es sollte nur jeweils ein Default-Element pro Regelsatz definiert werden. Sind mehrere Default-Elemente definiert, wird immer das letzte Element des XML-Regelsatzes verwendet.

Wert	Erläuterung / Beispiel	Pflicht-angabe
`true`	Konfiguration des `element`-Tags als Standard-Umformungsregel für unbekannte HTML-Formatierungen aus dem Word-Dokument, z.B. `<element default="true" handler="map" id="HTML.paragraph" tag="p">`	nein

4.4.5 id

Über das Attribut `id` können `element`-Tags mit einem Namen versehen werden. Auf diesen Namen können andere `element`-Tags über das Attribut `inherit` (siehe Kapitel 4.4.9 Seite 34) referenzieren und damit alle Attribute und Werte übernehmen, z.B.

```
<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 diesem Beispiel erhält das Element mit dem Attribut `tag="p"` (Fließtext-Absätze) den Namen „HTML.paragraph“. Tabellenzellen (HTML-Tag `td`) und Absätze innerhalb einer Tabellenzelle (HTML-Tag `p` innerhalb der Tabelle, `tag="table"`)

referenzieren über die `id` „HTML.paragraph“ auf dieses Element und verwenden damit Werte und Attribute von Fließtext-Absätzen (siehe dazu auch Kapitel 4.2 Seite 17).

4.4.6 findtag

Über das Attribut `findtag` können zusammen mit dem Attribut `handler=„find“` (siehe Kapitel 4.4.3 Seite 29) Tags aus dem Word-Dokument ausgelesen werden, z.B.

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

In diesem Beispiel wird aus dem `head`-Abschnitt des Word-Dokuments der Text des `title`-Elements ausgelesen.

4.4.7 content

Das Attribut `content` wird für den Import von Bildern benötigt, z.B.

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

Der Wert `IGNORE` sorgt dafür, dass das Bild nicht geparst wird.

4.4.8 mediaref

Das Attribut `mediaref` ist für den Import von Bildern erforderlich, z.B.

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

Für `mediaref` muss als Wert immer `src` angegeben werden.

4.4.9 inherit

Über das Attribut `inherit` kann ein `element`-Tag die Werte und Attribute eines anderen `element`-Tags erben, z.B.

```
<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 diesem Beispiel erhält das Element mit dem Attribut `tag="p"` (Fließtext-Absätze) den Namen „HTML.paragraph“. Tabellenzellen (HTML-Tag `td`) und Absätze innerhalb einer Tabellenzelle (HTML-Tag `p` innerhalb der Tabelle, `tag="table"`) referenzieren über die `id` „HTML.paragraph“ auf dieses Element und verwenden damit Werte und Attribute von Fließtext-Absätzen (siehe dazu auch Kapitel 4.2 Seite 17).

Mit dem Wert `NONE` kann unterbunden werden, dass das jeweilige `element`-Tag die Attribute und Werte eines übergeordneten `element`-Tags erbt:

```
<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>
```

Dieses Beispiel beinhaltet die Regeln zur Umformung einer Tabelle. Über das `inherit="NONE"` wird angegeben, dass alle zuvor definierten Umformungsregeln nicht auf Tabellen angewendet werden.

4.4.10 break

Über das Attribut `break` kann die Umformung von Textumbrüchen definiert werden, z.B.

```
<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>
```

Dieses Beispiel beinhaltet die Regeln zur Umformung einer Tabelle. Durch `break="br"` werden Textumbrüche innerhalb von Tabellenzellen in Word auch in der DOM-Eingabekomponente realisiert, während `<p>`-Tags entfernt werden (`tag="p" handler="strip"`).

4.4.11 mapattributes

Über das Attribut `mapattributes` können Attribute aus dem Word-Dokument nach FirstSpirit übernommen werden, z.B.

```
<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>
```

Dieses Beispiel beinhaltet die Regeln zur Umformung einer Tabelle. Durch die Angabe von `mapattributes="true"` werden die Attribute der Tabelle aus dem Word-Dokument wie z.B. `valign`, `bgcolor` usw. beim Import in eine DOM-Eingabekomponente verwendet. Wird der Wert `false` gesetzt, werden die Attribute nicht verwendet.

4.4.12 class

Über das Attribut `class` werden Klassen aus dem Word-Dokument mit Format- bzw. Verweisvorlagen in FirstSpirit verbunden, z.B.

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

In diesem Beispiel wird die Word-Zeichenformatvorlage „Wichtig“ mit der FirstSpirit-Formatvorlage mit dem Kürzel „important“ verbunden, d.h. Absätze, die im Word-Dokument mit der Absatzformatvorlage „Wichtig“ formatiert sind, werden bei der Übernahme in den DOM-Editor mit der Zeichenformatvorlage „Important“ formatiert.

4.5 Umgang mit unbekannten Tags

Format- oder Absatzvorlagen aus dem Word-Dokument, die nicht in den hinterlegten XML-Regelsätzen definiert sind, können auch nicht entsprechend in FirstSpirit umgeformt werden.

Für diesen Fall kann ein Standard-Verhalten festgelegt werden, z.B. dass jede unbekannte Formatierung aus Word in FirstSpirit in einen Absatz (`<p>`) umgewandelt wird.

Dazu wird im `element`-Tag für Fließtext-Absätze das Attribut `handler` auf den Wert „default“ gesetzt (siehe Kapitel 4.4.4 Seite 31), z.B.

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

Um „unbekannte“ Elemente aus Word mit der oben definierten „Absatz-Formatierung“ zu formatieren, wird für sie ein `element`-Tag mit `tag="*"` angelegt. Dabei fungiert `*` als Platzhalter.

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

4.6 Beispiel: Erstellen einer Regel zum Abbilden einer individuellen Word-Formatvorlage

In diesem Kapitel wird die Vorgehensweise beim Erstellen eigener XML-Regeln am Beispiel der Word-Formatvorlage „Hinweis“ erläutert.

4.6.1 Schritt 1: Vorbereitung des Word-Dokuments

Im Word-Dokument ist die Vorlage „Hinweis“ enthalten. Sie hat die Eigenschaften: rote Schrift (RGB-Farbe 255, 0, 0) und fett. Ein Absatz ist mit dieser Vorlage formatiert. Im Word-Test-Dokument z.B.

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

4.6.2 Schritt 2: Anlegen der erforderliche Formatvorlage in FirstSpirit

In FirstSpirit wird eine Formatvorlage benötigt, die die Formatierung der Word-Vorlage „Hinweis“ im DOM-Editor abbilden kann. Sie soll ebenfalls rot und gefettet sein.

Dazu wird eine Formatvorlage mit folgenden Eigenschaften angelegt:

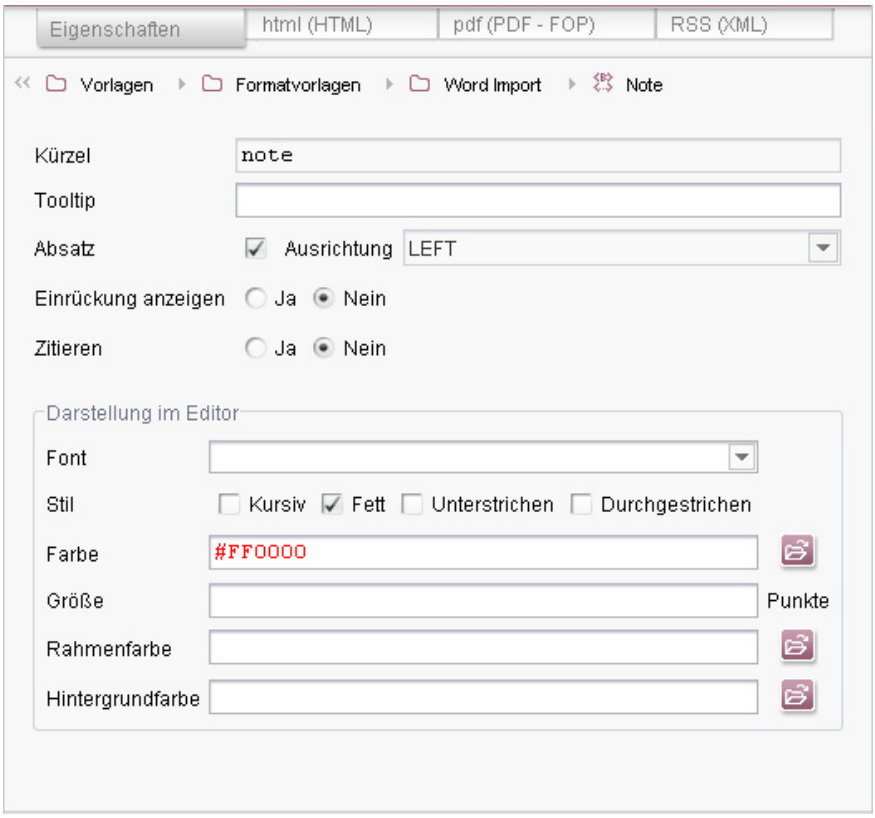

Abbildung 4-2: Formatvorlage „note“

Darüber hinaus müssen selbstverständlich noch die Ausgabekanäle ensprechend konfiguriert werden.

4.6.3 Schritt 3: Ausgabe des Word-Dokuments in HTML

Im HTML-Format des Word-Dokuments wird dieser Absatz folgendermaßen ausgegeben:

```
<p class="Hinweis">Dieser Text ist mit dem Absatzformat „Hinweis“
formatiert, mit einer Schriftfarbe mit dem <span
class="SpellE">RGB-Wert</span> 255, 0, 0. </p>
```

4.6.4 Schritt 4: Analyse des HTMLs

Das `<p>` zeigt an, dass es sich um einen Absatz handelt, mit `class` wird der Name der Vorlage angegeben.

4.6.5 Schritt 5: Erstellen der XML-Regeldefinition

Für die Regeldefinition ist ein `element`-Tag erforderlich. Da es sich um eine

zusätzliche Regelbeschreibung für einen Absatz handelt, muss das Tag auf der Ebene unterhalb der Regelbeschreibung für das HTML-Tag „body“ stehen. Für das `<p>` aus dem HTML erfordert das XML ein `tag`-Attribut:

```
tag="p"
```

Grundsätzlich sollen aber alle Eigenschaften, die im XML für Absätze definiert sind, auch für diese Formatvorlage gelten. Dies wird über das Attribut `inherit` umgesetzt:

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

Dabei ist `HTML.paragraph` die ID des Elementes (in diesem Fall die Regelbeschreibung für Absätze), dessen Werte und Attribute übernommen werden sollen.

Da es sich um eine selbst definierte Word-Formatvorlage handelt, wird diese mit `class` definiert. Im XML-Regelsatz muss daher das `class`-Attribut verwendet werden, mit dem der Name der Word-Formatvorlage angegeben wird:

```
class="Hinweis"
```

Die entsprechende FirstSpirit-Formatvorlage wird dem `maptag`-Attribut übergeben:

```
maptag="note"
```

4.7 Standard-Regelsatz des FirstSpirit-Office-Moduls

Der folgende XML-Regelsatz wird standardmäßig mit dem Office-Modul ausgeliefert:

```
<?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 Konfiguration der Eingabekomponenten

Die Office-Funktionalität kann im DOM-Editor und in der DOM-Tabelle verwendet werden.

In die Eingabekomponente DOM-Tabelle können aus Word-Dokumenten nur Tabellen, aber keine Fließtexte importiert werden (siehe auch Kapitel 6.7 Seite 62).

5.1 Aktivieren der Import-Funktionalität

Um die Office-Modul-Funktion im DOM-Editor bzw. in der DOM-Tabelle zur Verfügung zu stellen, muss im Formular-Bereich der Eingabekomponente CMS_INPUT_DOM bzw. CMS_INPUT_DOMTABLE in den gewünschte Absatz-, Seiten- oder Tabellen-Vorlagen der Parameter `enableImport="yes"` hinzugefügt werden. Die Eingabekomponenten erhalten dann beim Einbinden in die Inhalte- oder Datenquellen-Verwaltung ein Import-Icon:

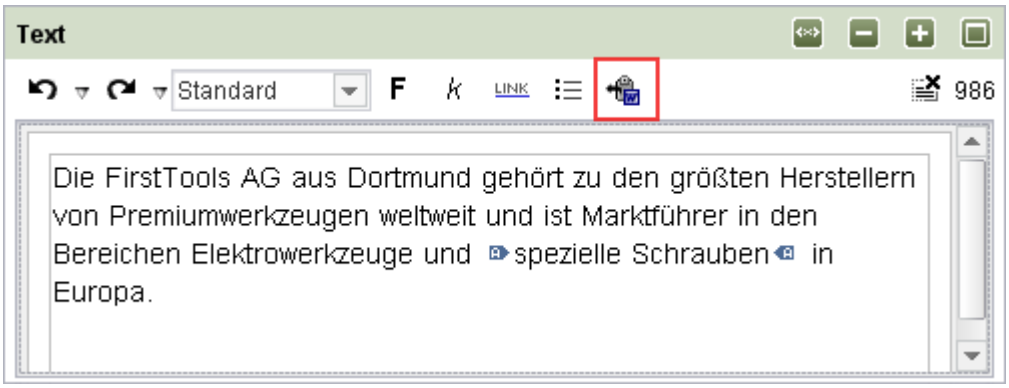

Abbildung 5-1: DOM-Editor mit Icon für den Office-Import

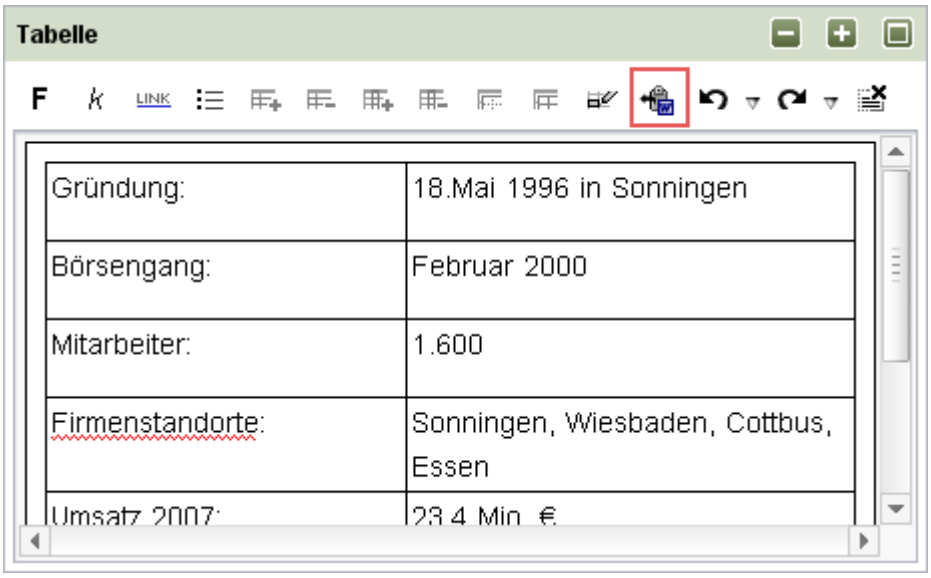

Abbildung 5-2: DOM-Tabelle mit Icon für den Office-Import

5.2 Auswählbare Regelsätze einschränken

Über den optionalen Parameter `importRuleset` kann definiert werden, welche Regelsätze für den jeweiligen DOM-Editor bzw. die jeweilige DOM-Tabelle zur Verfügung stehen sollen und vom Redakteur ausgewählt werden können. Dazu wird der Name des Regelsatzes, der in der XML-Defintion festgelegt wurde (siehe Kapitel 4.3.2 Seite 20), in doppelten Hochkommata angegeben werden. Dabei kann es sich sowohl um Regelsätze handeln, die für den gesamten Server oder nur für das jeweilige Projekt definiert wurden, z.B.:

```
importRuleset="Projekt"
```

In diesem Fall wird nur der Regelsatz mit dem Namen „Projekt“ zur Verfügung gestellt.

Es kann mehr als ein Regelsatz angegeben werden. In dem Fall sind die Regelsätze durch Komma getrennt anzugeben.

Wird der Parameter nicht angegeben, werden in der Eingabekomponente alle für das jeweilige Projekt sowie die serverweit gültigen Regelsätze zur Auswahl angezeigt.

5.3 Einschränkungen (Format-, Verweisvorlagen, Listen, Tabellen)

Damit die Zuordnung zwischen den Inhalten aus dem Word-Dokument und den FirstSpirit-Formatvorlagen korrekt funktioniert, dürfen diese Formatvorlagen im gewünschten DOM-Editor bzw. in der gewünschten DOM-Tabelle nicht eingeschränkt sein.

Die Verwendung von Formatvorlagen wird im DOM-Editor über das Tag `<FORMATS>` eingeschränkt: Jede Formatvorlage, die zugelassen sein soll, wird mit einem eigenen `<TEMPLATE>`-Tag innerhalb des öffnenden und schließenden `<FORMATS>`-Tags angegeben, z.B.:

```
<FORMATS>
   <TEMPLATE name="Kürzel der Formatvorlage">
</FORMATS>
```

Wird kein `<FORMATS>`-Tag angegeben, sind alle Formatvorlagen zugelassen.

Wird im Word-Dokument eine Formatierung verwendet, z.B. **fett**, sollte auch die FirstSpirit-Formatvorlage für **fett** (Standard-Formatvorlage mit dem Kürzel „b“) für den DOM-Editor zugelassen sein. Ansonsten können zwar fett formatierte Inhalte in den DOM-Editor importiert und auch mit der Formatierung angezeigt werden, allerdings kann die Formatierung im DOM-Editor nicht bearbeitet werden. Das Icon zur Formatierung in **fett** ist dann deaktiviert.

Bei der Einschränkung von Formatvorlagen für den DOM-Editor, in den das Word-Dokument importiert werden soll, sollte also darauf geachtet werden, dass alle in Word verwendeten Formate zugelassen werden.

Dasselbe gilt für Verweise, Listen und Tabellen:

Verweise: Die Verwendung von Verweisen kann im DOM-Editor über das Tag `<LINKEDITORS>` eingeschränkt werden. Jede Verweisvorlage, die zugelassen werden soll, wird mit einem eigenen `<LINKEDITOR>`-Tag innerhalb des öffnenden und schließenden `<LINKEDITORS>`-Tags angegeben, z.B.:

```
<LINKEDITORS>
   <LINKEDITOR name="Eindeutiger Name der Verweisvorlage">
</LINKEDITORS>
```

Wird kein `<LINKEDITORS>`-Tag angegeben, sind alle Verweisvorlagen zugelassen.

Bei der Einschränkung von Verweisvorlagen für den DOM-Editor, in den das Word-Dokument importiert werden soll, sollte also darauf geachtet werden, dass die

erforderlichen Verweisvorlagen (für die Nutzung der Standard-Regelsätze „textlinkinternal.standard“ bzw. „textlinkinternal“ und „textlinkexternal.standard“ bzw. „textlinkexternal“) zugelassen werden.

Listen: Die Verwendung von Listen kann im DOM-Editor über den Parameter `list="NO"` eingeschränkt werden. Werden im Word-Dokument Listen verwendet, die in den DOM-Editor importiert und dort bearbeitet werden sollen, sollte darauf geachtet werden, dass der Parameter nicht auf `NO` gesetzt ist. Je nach Projektkonfiguration und Word-Dokument muss eventuell die Listenkonfiguration (Attribute `listConfig` und `listDefaultConfig`) angepasst werden.

Hinweise zur korrekten Ausgabe von geschachtelten Listen siehe Kapitel 5.4 Seite 46.

Tabellen (nur DOM-Editor): Die Verwendung von Tabellen im DOM-Editor (so genannte „Inline-Tabellen“) kann über das Attribut `table` gesteuert werden. Werden im Word-Dokument Tabellen verwendet, so sollte der Tabellen-Modus für den jeweiligen DOM-Editor aktiviert werden (`table="YES"`).

Weitere Informationen zu Konfigurationsmöglichkeiten des DOM-Editors bzw. der DOM-Tabelle siehe FirstSpirit Online Dokumentation, Kapitel „Vorlagenentwicklung“ / „Formulare“ / „Eingabekomponenten“ / „DOM“ bzw. „DOMTABLE“.

5.4 Ausgabe von Aufzählungen

Werden in Microsoft Office Word verschachtelte Aufzählungen verwendet, weicht die Darstellung in HTML von den Empfehlungen des World Wide Web Konsortiums (W3C) ab.

Beispiel HTML-Code, generiert aus Word:

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

Gemäß der HTML 4.01-Spezifikation des W3C sind für UL/OL-Elemente jeweils nur LI-Elemente zulässig (siehe http://www.w3.org/TR/html401/struct/lists.html).

Aus einer in Word erstellten, geschachtelten Liste wird beim Import in den DOM-Editor durch das automatische Hinzufügen von LI-Elementen eine valide Liste erzeugt:

```
<ul style="1">
  <li>abc</li>
  <li><ul>
     <li>bcd</li>
  </ul></li>
  <li>cde</li>
</ul>
```

In FirstSpirit werden geordnete Listen (“ol”) als ungeordnete Listen (“ul”) gespeichert!

Dies führt bei der Ausgabe im Browser allerdings zu einer Abweichung, wie die folgenden Abbildungen zeigen: Abbildung 5-3 stellt die Ausgabe im Browser des von Microsoft Word erzeugten HTML-Codes dar, Abbildung 5-4 stellt die Ausgabe im Browser des von FirstSpirit erzeugten HTML-Codes dar:

- Erste Ebene, erster Punkt
    1. Zweite Ebene, erster Punkt
- Erste Ebene, zweiter Punkt

Abbildung 5-3: Ausgabe von geschachtelten Listen (Microsoft Word)

- Erste Ebene, erster Punkt
-
    - Zweite Ebene, erster Punkt
- Erste Ebene, zweiter Punkt

Abbildung 5-4: Ausgabe von geschachtelten Listen (FirstSpirit)

Um das Render-Verhalten von FirstSpirit im Browser an das von Word anzupassen, muss daher die FirstSpirit-Standard-Formatvorlage „Listeneintrag“ mit dem Kürzel „li“ geändert werden.

Dazu muss im HTML-Kanal anstelle von

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

der Inhalt folgendmaßen ausgegeben werden:

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

*Wird die Standard-Formatvorlage „li“ auf diese Weise geändert, wirkt sich dies auf die Ausgabe **aller** Listen in FirstSpirit aus. Um dies zu unterbinden und die Ausgabe nur für DOM-Editoren zu ändern, die geschachtelte Listen aus Word verwenden, können entsprechend angepasste Formatvorlagen verwendet werden. Mindestens die Formatvorlage „Standard“ (für Absätze, die mit dem <p>-Tag formatiert werden sollen,) sollte angepasst verwendet werden, wie das folgende Beispiel zeigt:*

Dazu wird eine Formatvorlage mit aktivierter Option „Absatz“ angelegt. Im HTML-Ausgabekanal wird eine Variable (z.B. mit dem Namen *_isWordList*) definiert:

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

Diese Formatvorlage muss im DOM-Editor, in dem die geschachtelten Listen verwendet werden sollen, verfügbar sein:

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

Die Angabe im HTML-Kanal der Standard-Formatvorlage „li“ muss in diesem Fall folgendermaßen erweitert werden:

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

6 Redaktionelles Arbeiten im JavaClient

Die nachfolgenden Erklärungen beziehen sich in erster Linie auf das mitgelieferte Word-Test-Dokument.

6.1 Erläuterungen zum Word-Dokument

Die folgende Tabelle listet die Formatvorlagen auf, die im Word-Dokument verwendet werden, und stellt sie den entsprechenden FirstSpirit-Vorlagen (teilweise Standard-Formatvorlagen, teilweise projektspezifische Vorlagen aus dem Demo-Projekt „Mithras Energy“) gegenüber:

Word-Formatvorlage	**Beispiel**	**FirstSpirit-Vorlage**
Überschrift 1	„Test-Dokument für den Word-Import mit FirstSpirit Office“	h1
Überschrift 2	„Einfacher Fließtext“	h2
Überschrift 3	„Fließtext mit Absatzschaltungen“	h3
Standard	„Absatz 1: Lorem ipsum dolor sit amet...“	p
Fett	„Hier ist mal ein ganzer Satz fett gestellt.“	b
Kursiv	„Hier ist mal ein ganzer Satz kursiv gestellt.“	k
Unterstrichen	„Hier ist mal ein ganzer Satz unterstrichen.“	u
Aufgezählt (Symbol) • Aufgezählt	„• Erster Listenpunkt“	ul / li

Word-Formatvorlage	Beispiel	FirstSpirit-Vorlage
Aufgezählt (Buchstabe) a. Aufgezählt	„a. Zweite Ebene, erster Listenpunkt“	ul / li
Hinweis	„Dieser Text ist mit dem Absatzformat „Hinweis“ formatiert, mit einer Schriftfarbe mit dem RGB-Wert 255, 0, 0.“	note
Wichtig	„3.4 (Absatzformat= „Wichtig“)“	important
Hyperlink	„http://www.firstspirit.de“, „www.firstspirit.de“, „aktuelle Neuigkeiten"	externe Verweisvorlage `textlinkexternal.standard` bzw. `textlinkexternal`
<Bilder>	FirstSpirit™	interne Verweisvorlage `textlinkinternal.standard` bzw. `textlinkinternal`

Die **Absätze** „Absatz 1“, „Absatz 2“ und „Absatz 3“ stellen Textabsätze dar, die in HTML durch <p>-Tags definiert werden. Zwischen den Absätzen 2 und 3 befindet sich zusätzlich eine Leerzeile. In „Absatz 4“ befindet sich vor der Zeile „Consetetur sadipscing elitr ...“ ein weicher Zeilenumbruch.

Das FirstSpirit-Logo ist als **Bild** einmal in den Textfluss integriert (Absatz „Bilder in Zeilen“), einmal nimmt es einen eigenen Absatz ein (Absatz „Bilder in Absätzen“). Darüber hinaus ist das e-Spirit-Logo in eine Tabellenzelle integriert („Tabellen mit Verschmelzung und Formatierungen“).

An **Zeichenformatierungen** werden „Fett“, „Kursiv“, „Unterstrichen“, „Durchgestrichen“ und Kombinationen aus diesen verwendet. Als **Absatzformatierungen** werden drei Überschriftsebenen verwendet, die standardmäßig in Word enthalten sind. Darüber hinaus wird die selbst definierte

Word-Absatzvorlage „Hinweis“ verwendet.

Links können im DOM-Editor als externer Verweis abgebildet werden, unabhängig davon, ob es sich um einen im Word-Dokument durch Formatierung (blaue Schrift, unterstrichen) kenntlich gemachten Link (z.B. „http://www.firstspirit.de“) oder einen nicht kenntlich gemachten Link handelt (z.B. „aktuelle Neuigkeiten“, verlinkt auf „http://www.e-spirit.de“).

Aufzählungen werden vom mitgelieferten XML-Regelsatz im DOM-Editor aktuell nur mit der FirstSpirit-Formatierung „Spiegelstrich“ dargestellt. Bedingt durch eine Limitierung im Export-Format aus Word heraus können Aufzählungen nur bis zur zweiten Ebene korrekt in FirstSpirit umgesetzt werden. Alle Ebenen, die darüber hinaus gehen, werden als Text formatiert, z.B. mit Leerzeichen statt mit Tabulator-Einzügen.

Hinweise zur korrekten Ausgabe von geschachtelten Listen siehe auch Kapitel 5.4 Seite 46.

In den **Tabellen** werden ebenfalls die Word-Formatvorlagen „Standard“, „Fett“, „Kursiv“, „Unterstrichen“ usw. verwendet. Die Zellen sind durchnummeriert, und zwar zeigt die Nummer vor dem Punkt die Spaltennummer an, die Nummer nach dem Punkt die Zeilennummer. Beispielsweise steht „2.3“ für eine Zelle in der zweiten Spalte der dritten Zeile. Verschmelzungen werden durch ein „+“ symbolisiert: „2.2 + 3.2“ bedeutet z.B., dass die Zelle der zweiten Spalte in der zweiten Zeile mit der darunter liegenden Zelle (zweite Spalte, dritte Zeile) verschmolzen wurde. Zwischen den Tabellen und dem vorangehenden Text befindet sich jeweils eine Leerzeile.

6.2 Import in den FirstSpirit JavaClient

Ist das Modul FirstSpirit Office für ein Projekt konfiguriert und in einem DOM-Editor oder einer DOM-Tabelle aktiviert, wird im JavaClient der „Import“-Button sichtbar:

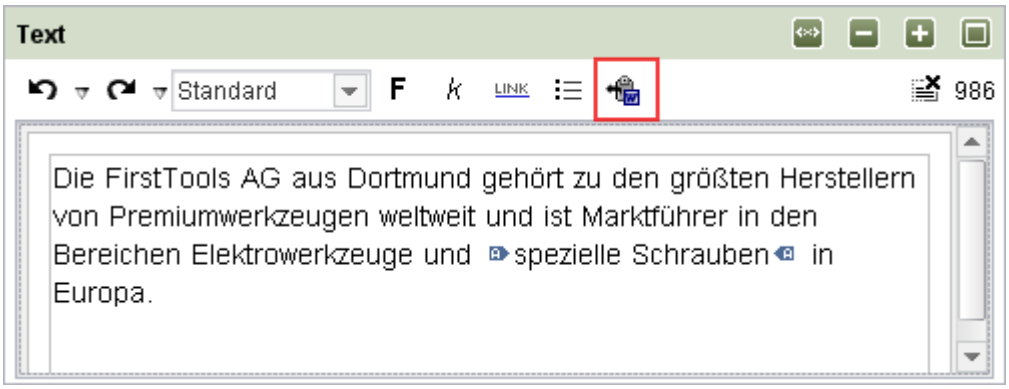

Abbildung 6-1: DOM-Editor mit Import-Button

Um Inhalte aus Word in den DOM-Editor zu importieren, muss zunächst der Cursor an die Position gesetzt werden, an der die Inhalte eingefügt werden sollen.

Im Word-Dokument wird dann der zu importierende Inhalt markiert und mit <STRG> + <C> oder über das Kontextmenü in die Zwischenablage kopiert. Nach einem Klick auf den Import-Button in der Eingabekomponente oder über die Tastaturkombination <STRG> + <V> wird der Inhalt aus dem Word-Dokument direkt in die Eingabekomponente an der gewählten Position eingefügt und eine automatische Formatierung gemäß des vom Vorlagenentwickler vorgegebenen Regelsatzes vorgenommen.

Stehen für die Eingabekomponente mehrere Regelsätze zu Verfügung, erscheint ein Fenster, aus denen der Regelsatz ausgewählt werden kann, nach dem der Inhalt transformiert werden soll:

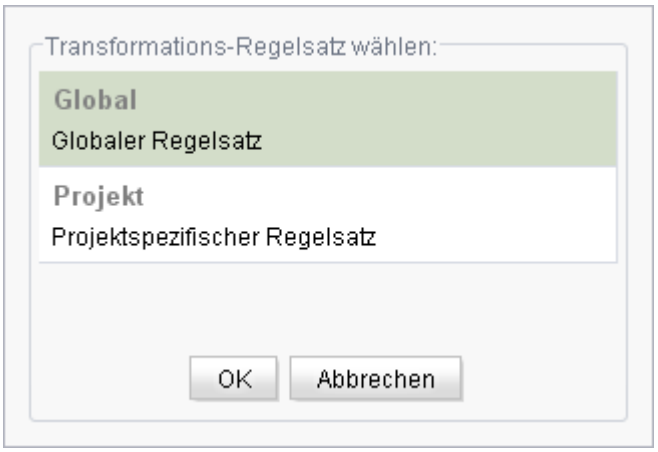

Abbildung 6-2: Transformations-Regelsatz wählen

Der gewünschte Regelsatz wird mit der Maus markiert und die Auswahl mit OK bestätigt. Der Inhalt aus dem Word-Dokument wird anschließend in die Eingabekomponente an der gewählten Position eingefügt und die Formatierung

gemäß des gewünschten Regelsatzes vorgenommen. Mit 

wird der Text aus dem Word-Dokument unformatiert in die Eingabekomponente eingefügt.

Der Inhalt kann anschließend mit den in der Eingabekomponente zur Verfügung stehenden Funktionen wie gewohnt weiter beabeitet werden.

Wurde in die Zwischenablage kein Word-Format übertragen, wird nach dem Klicken des Import-Buttons (bzw. <STRG> + <V>) die Meldung „Kein kompatibles Format in der Zwischenablage gefunden!“ angezeigt. Es wird kein Import vorgenommen.

Um die korrekte Funktionalität des Office-Moduls zu gewährleisten, sollten Fließtext aus Word-Dokumenten nur in den DOM-Editor oder in so genannte Inline-Tabellen eingefügt werden, Tabellen aus Word nur in die DOM-Table oder in so genannte Inline-Tabellen. Ansonsten können eingefügte Inhalte nicht mehr weiter bearbeitet werden.

6.3 Übernahme von formatierten Texten

Fließtexte aus einem Word-Dokument können beim Importieren in DOM-Eingabekomponenten inklusive Absatzschaltungen, Zeichen- und Absatz-Formatierungen übernommen werden.

6.3.1 Text aus Word-Dokumenten

Aus dem Word-Test-Dokument wird folgender Bereich markiert und in die Zwischenablage übertragen (<STRG> + <C>):

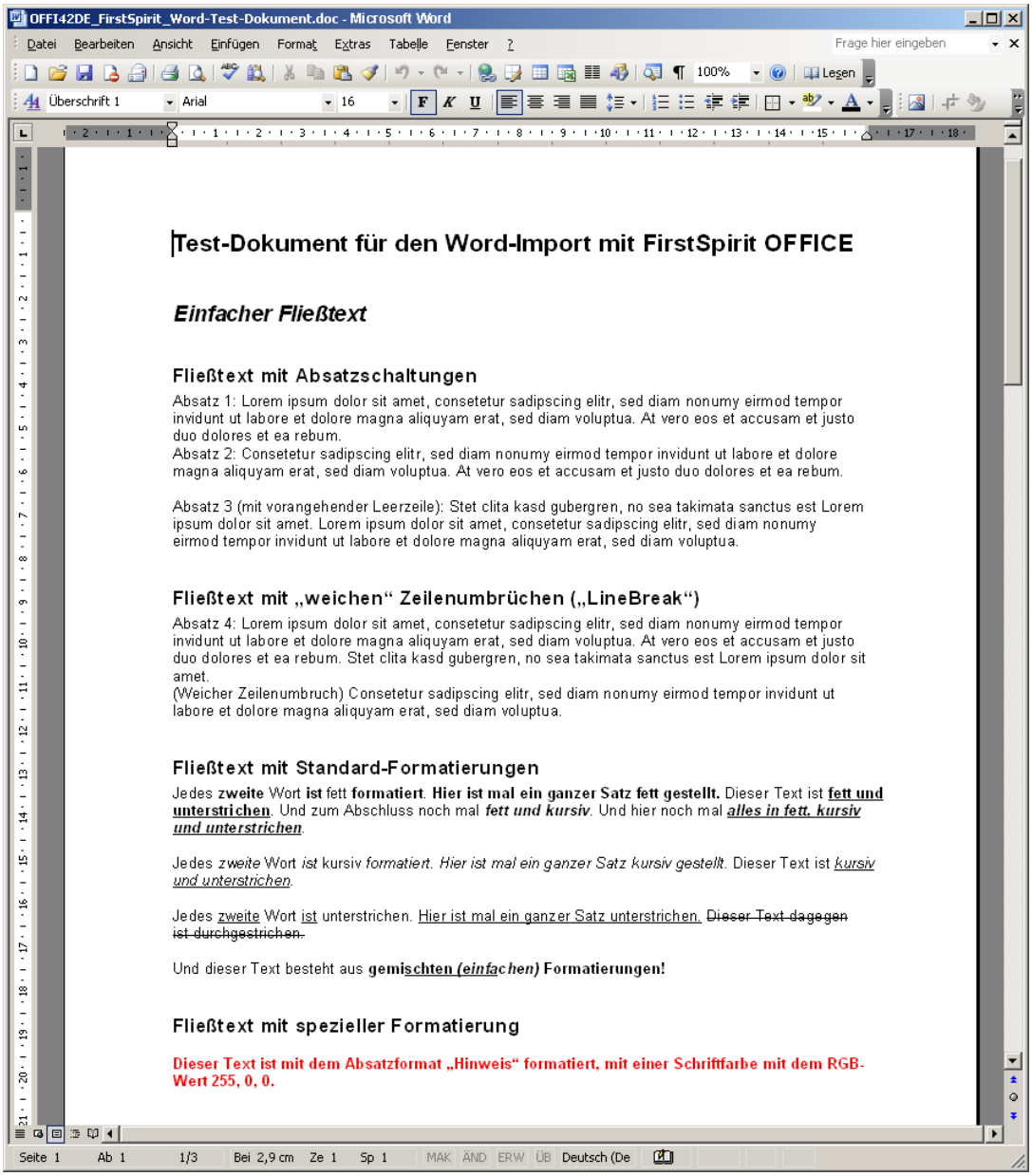

Abbildung 6-3: Word-Test-Dokument: Fließtext

6.3.2 Text im DOM-Editor

Der Text aus dem Test-Dokument kann nach dem Import über die Office-Funktionalität folgendermaßen im DOM-Editor angezeigt werden:

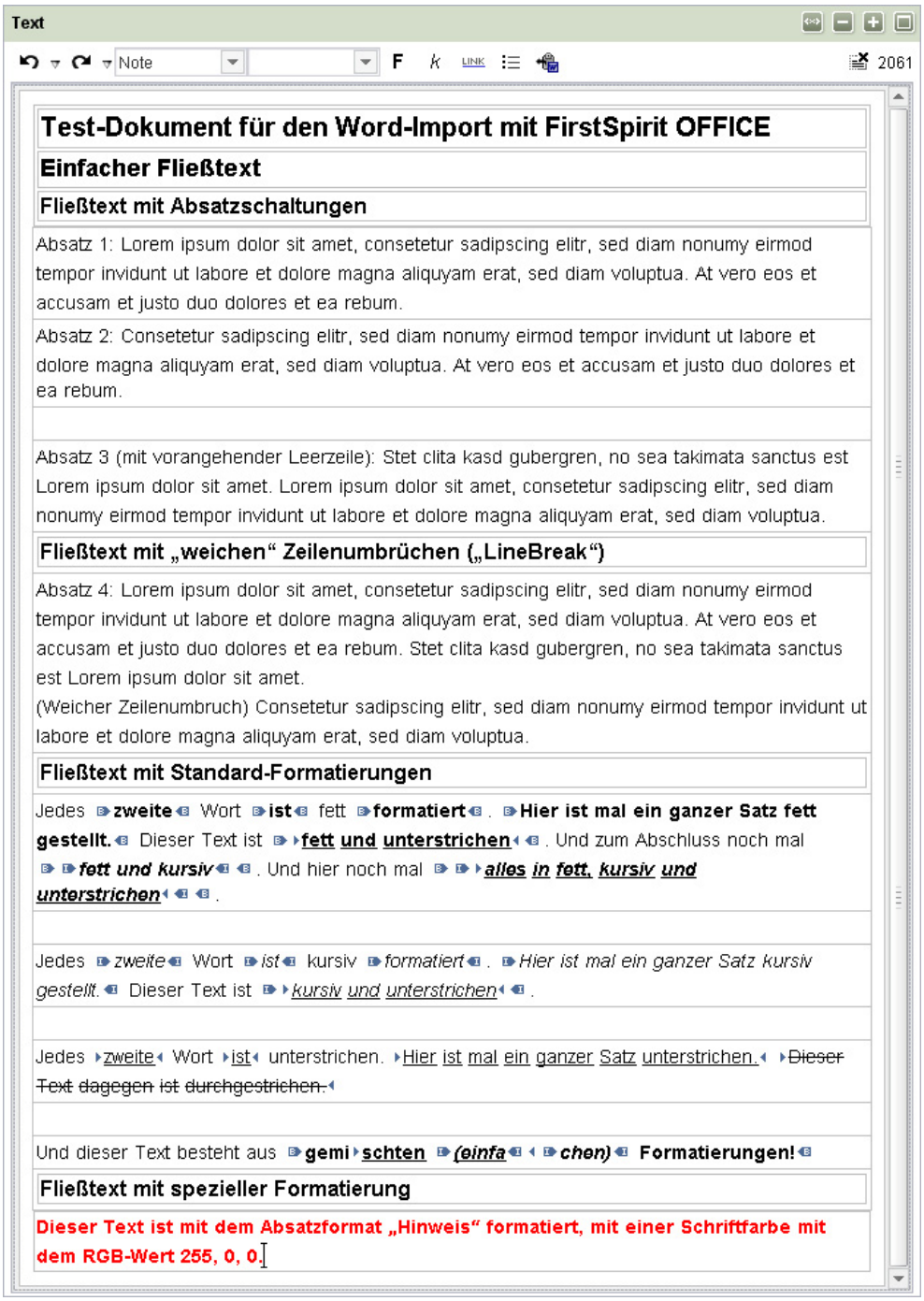

Abbildung 6-4: DOM-Editor mit Text aus dem Word-Dokument

Erläuterung: Die Überschriften der ersten drei Ebenen werden in FirstSpirit auf die zugehörigen FirstSpirit-Formatvorlagen mit den Kürzeln „h1“, „h2“ und „h3“ abgebildet. Absatz 1 und Absatz 2 stellen Textabsätze dar, die in HTML durch `<p>...</p>` definiert werden. Sie werden im DOM-Editor auf die Standard-Formatvorlage „Standard“ abgebildet. Zwischen Absatz 2 und 3 befindet sich eine Leerzeile.

Die Zeichenformatierungen aus dem Word-Dokument für fett, kursiv und unterstrichen werden im DOM-Editor auf die FirstSpirit-Formatvorlagen mit den Kürzeln „b“, „i“, „u“ und „s“ abgebildet. Die Absatzformatierung „Hinweis“ aus dem Word-Dokument wird im DOM-Editor auf die Formatvorlage mit dem Kürzel „note“ abgebildet.

6.3.3 Beispielhafte HTML-Ausgabe

In der HTML-Ausgabe auf der Webseite kann der Text je nach Definition des Vorlagenentwicklers folgendermaßen aussehen:

```
<h1>Test-Dokument für den Word-Import mit FirstSpirit OFFICE</h1>

<h2>Einfacher Fließtext</h2>

<h3>Fließtext mit Absatzschaltungen</h3>
          <p class="section">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.</p>

          ...
```

6.4 Übernahme von Aufzählungen

Aufzählungen werden beim Importieren in DOM-Eingabekomponenten aufgrund einer Limitierung im Export-Format aus Word heraus bis zur zweiten Ebene übernommen.

Hinweise zur korrekten Ausgabe von geschachtelten Listen siehe auch Kapitel 5.4 Seite 46.

6.4.1 Aufzählungen aus Word-Dokumenten

Aus dem Word-Test-Dokument wird folgender Bereich markiert und in die Zwischenablage übertragen (<STRG> + <C>):

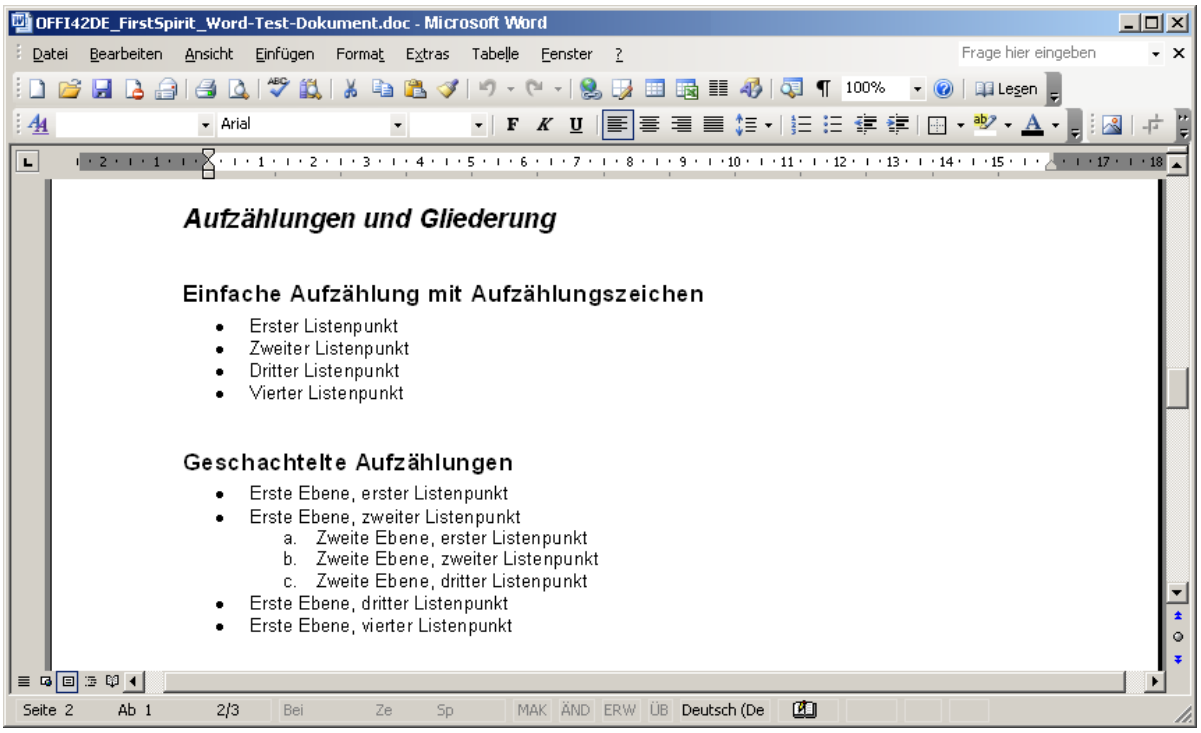

Abbildung 6-5: Word-Test-Dokument: Aufzählungen

6.4.2 Aufzählungen im DOM-Editor

Der Text mit Aufzählungen aus dem Test-Dokument kann nach dem Import über die Office-Funktionalität folgendermaßen im DOM-Editor angezeigt werden:

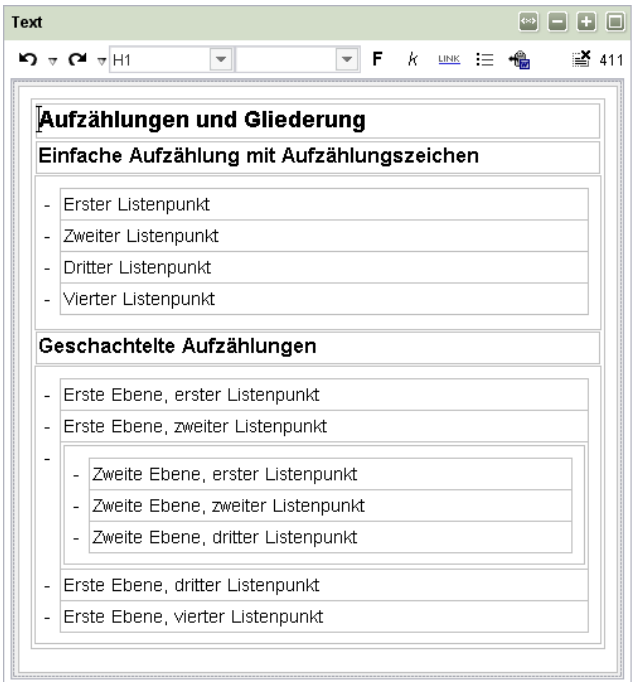

Abbildung 6-6: : DOM-Editor mit Aufzählungen aus dem Word-Dokument

6.5 Übernahme von Links

Links werden beim Importieren werden beim Einfügen in den DOM-Editor als externe Verweise realisiert:

Abbildung 6-7: Externer Link

Dabei wird der Anzeigetext aus dem Word-Dokument (hier „aktuelle Neuigkeiten“) ins Feld „Verweis-Text“ übernommen, der Link in das Feld „Ziel Adresse“. Alle Felder können wie gewohnt weiter bearbeitet werden.

6.5.1 Links aus Word-Dokumenten

Aus dem Word-Test-Dokument wird folgender Bereich markiert und in die Zwischenablage übertragen (<STRG> + <C>):

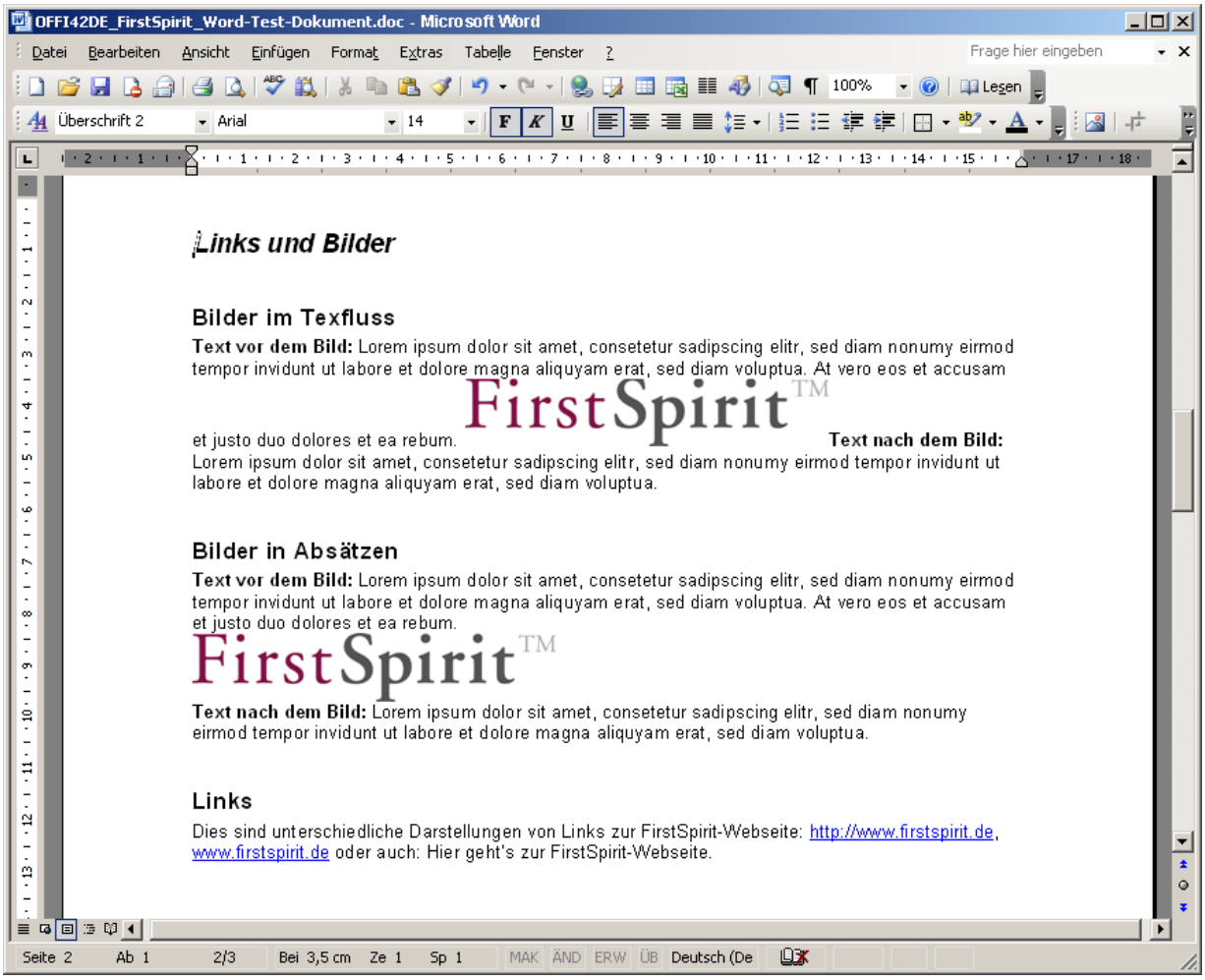

Abbildung 6-8: Word-Test-Dokument: Links und Bilder

6.5.2 Links im DOM-Editor

Der Text mit Links aus dem Test-Dokument kann nach dem Import über die Office-Funktionalität folgendermaßen im DOM-Editor angezeigt werden:

Abbildung 6-9: DOM-Editor mit Bildern und Links aus dem Word-Dokument

Je nach Konfiguration des Projektes können externe Links auch auf der Basis generischer Link-Editoren abgebildet werden. Funktion und Layout können dann abweichen. Zur redaktionellen Arbeit mit Generischen Link-Editoren siehe FirstSpirit Handbuch für Redakteure (JavaClient).

6.6 Übernahme von Bildern

6.6.1 Einfügen in den DOM-Editor

Ist ein oder mehrere Bilder in der Zwischenablage enthalten, werden diese beim Einfügen in den DOM-Editor als interner Link realisiert. Dabei wird unterschieden, ob sie in den Textfluss eingebettet sind („Bilder in Zeilen“) oder einen eigenen Absatz

einnehmen („Bilder in Absätzen“) (siehe auch Abbildung 6-9):

Abbildung 6-10: Interner Link zur Abbildung von Bildern

Dafür wird die Verweisvorlage verwendet, die im `mapping`-Element definiert ist (siehe Kapitel 4.3.2 Seite 20). Im Feld „Bild“ ist der Pfad zum Bild in der Medien-Verwaltung hinterlegt (siehe Kapitel 6.6.2 Seite 61).

Je nach Konfiguration des Projektes können interne Links auch auf der Basis generischer Link-Editoren abgebildet werden. Funktion und Layout können dann abweichen. Zur redaktionellen Arbeit mit Generischen Link-Editoren siehe FirstSpirit Handbuch für Redakteure (JavaClient).

6.6.2 Anlegen von Bildern in der Medien-Verwaltung

Die Bilder werden beim Import gleichzeitig automatisch in die Medien-Verwaltung eingefügt, und zwar in einem Ordner, der ebenfalls beim Einfügen der Bilder in der DOM-Editor neu angelegt wird.

Die UID des **Ordners** wird nach folgendem Schema gebildet:

```
import_Datum_Uhrzeit_User
```

Dabei wird das Datum im Format *YYmmDD* angegeben, die Uhrzeit im Format *HH_MM* und für `User` der Loginname des Benutzers, der das Bild eingefügt hat, z.B.

```
import_09040115_16_Admin
```

Die UID des eingefügten **Bildes** wird nach folgendem Schema gebildet:

file_*Pfad_Dateiname*

Der Pfad zeigt dabei immer auf einen temporären Ordner auf dem lokalen Arbeitsplatzrechner („Temp“), in dem das Bild zwischengespeichert wird. Der

Dateiname wird ohne Dateiformat angegeben, z.B.

```
file____C__DOKUME_1_Benutzerkonto_1_LOKALE_1_Temp_msohtml1_01_clip_image002
```

In dieser Form werden Ordner und Bilder auch nach dem Einfügen in der Baumstruktur angezeigt, ein Anzeigename und ein Dateiname für das Bild können später individuell vergeben werden. In Freigabeprojekten müssen diese Elemente bei Bedarf freigegeben werden.

Bei jedem Import von Bildern aus Word-Dokumenten werden diese in der Medien-Verwaltung inklusive Ordner eingefügt, unabhängig davon, ob das gleiche Bild bereits vorhanden ist. Werden evtl. doppelt vorhandene Bilder aus der Medien-Verwaltung gelöscht, muss in den betreffenden DOM-Eingabekomponenten, in denen diese Bilder verwendet werden, die Referenz manuell angepasst werden.

6.7 Übernahme von Tabellen

Tabellen aus Word-Dokumenten können inklusive Formatierungen übernommen werden. Verschmelzungen werden ohne spezielle Konfiguration durch einen Regelsatz übernommen.

6.7.1 Tabellen aus Word-Dokumenten

Aus dem Word-Test-Dokument wird folgender Bereich markiert und in die Zwischenablage übertragen (<STRG> + <C>):

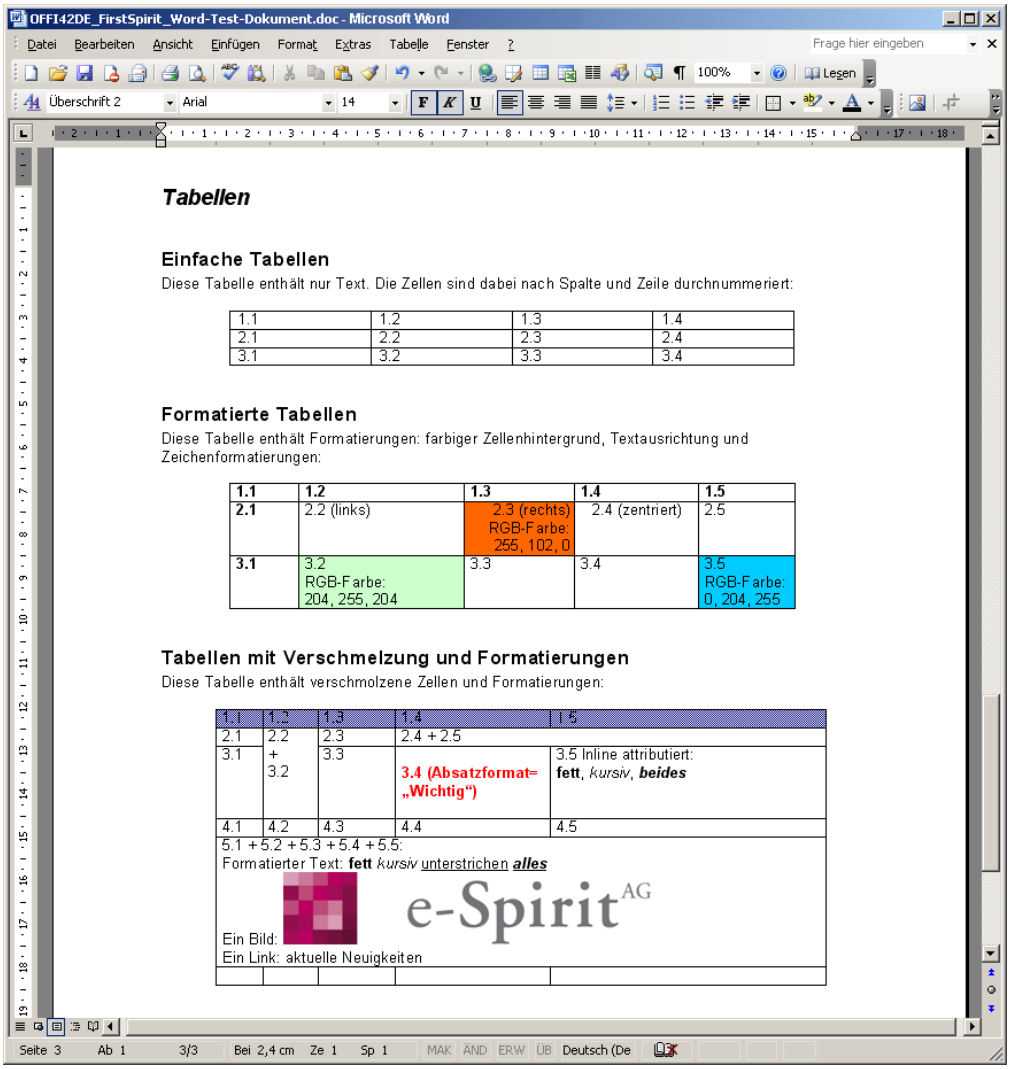

Abbildung 6-11: Word-Test-Dokument: Tabellen

6.7.2 Tabellen im DOM-Editor

Der Text mit Aufzählungen aus dem Test-Dokument kann nach dem Import über die Office-Funktionalität folgendermaßen im DOM-Editor angezeigt werden:

Abbildung 6-12: DOM-Editor mit Tabellen aus dem Word-Dokument

7 Rechtliche Hinweise

Das Modul „FirstSpirit Office“ ist ein Produkt der e-Spirit AG, Dortmund, Germany.

Für die Verwendung des Moduls gilt gegenüber dem Anwender nur die mit der e-Spirit AG vereinbarte Lizenz.

Details zu möglicherweise fremden, nicht von der e-Spirit AG hergestellten, eingesetzten Software-Produkten, deren eigenen Lizenzen und gegebenenfalls Aktualisierungs-Informationen, finden Sie auf der Startseite jedes FirstSpirit-Servers im Bereich „Rechtliche Hinweise".