FirstSpirit™

Unlock Your Content

FirstSpirit™ Handbuch für Entwickler (Grundlagen)

FirstSpirit™ Version 5.0

Version	**1.16**
Status	**RELEASED**
Datum	**2013-04-30**
Abteilung	FS-Core
Copyright	2013 e-Spirit AG

Datei DEVB50DE_FIRSTspirit_DeveloperDocumentationBasics

e-Spirit AG

Barcelonaweg 14
44269 Dortmund | Germany

T +49 231 . 477 77-0
F +49 231 . 477 77-499

info@e-Spirit.com
www.e-Spirit.com

e-Spirit

Inhaltsverzeichnis

1 Einführung

Ziel dieses Handbuches ist es, die Implementierung von FirstSpirit™-Projekten aus der Entwicklerperspektive zu beschreiben. Die Struktur der Dokumentation ist dabei so gewählt, dass ein möglichst umfassender Überblick über die für den Entwickler relevanten FirstSpirit™-Mechanismen und den jeweiligen Einsatzzwecke gegeben wird (siehe Kapitel 1.1 Seite 8).

Einige, insbesondere für die Vorlagenentwicklung erforderliche Bereiche, sind bereits ausführlich in der FirstSpirit™-Online-Dokumentation dokumentiert. Zur Einführung in die FirstSpirit™-Konzeption aus der Sicht der Vorlagenentwicklung bietet Kapitel 1.3 eine Erläuterung allgemeiner Begriffe (ab Seite 10).

1.1 Themen dieser Dokumentation

Diese Dokumentation beschreibt die relevanten Funktionalitäten und Aspekte für die Vorlagen-Entwicklung in FirstSpirit. Die Gliederung orientiert sich grob an der Bedienoberfläche des FirstSpirit JavaClients.

In **Kapitel 2** wird die Vorlagen-Verwaltung des FirstSpirit JavaClients mit allen zur Verfügung stehenden Kontextmenüs und Bearbeitungsmöglichkeiten beschrieben (siehe Kapitel 2 ab Seite 19).

FirstSpirit verfügt über leistungsfähige Mechanismen für die Anbindung von Datenbanken. **Kapitel 3** behandelt die in FirstSpirit verfügbaren Layer-Typen für die Datenbankanbindung und nennt einige generelle Empfehlungen für den Umgang mit Datenquellen in FirstSpirit (siehe Kapitel 3 Seite 117).

Arbeitsabläufe sind eine Abfolge von Aufgaben, die nach einer fest vorgegebenen Struktur abgearbeitet werden. Mit ihrer Hilfe können beispielsweise Freigabeprozesse modelliert werden. **Kapitel 4** erläutert den in FirstSpirit verwendeten Arbeitsablauf-Editor inklusive aller Konfigurationsmöglichkeiten (siehe Kapitel 4 ab Seite 123).

FirstSpirit bietet eine Möglichkeit zur Änderungsverfolgung über die FirstSpirit Access-API an. **Kapitel 5** beschreibt den Zugriff auf die Metadaten einer Revision über bestimmte API-Funktionen. Die Revisions-Metadaten enthalten Informationen über die Art (Welche Änderungen haben stattgefunden?) und den Umfang (Welche Elemente wurden geändert?) einer Änderung im Projekt (siehe Kapitel 5 Seite 210).

Neben der Freigabe über einen Arbeitsablauf können alle Objekte in FirstSpirit serverseitig über die Access-API freigegeben werden. **Kapitel 6** zeigt die Methoden auf, um die unterschiedlichen Freigabeeinstellungen für ein Objekt zu definieren (siehe Kapitel 6 Seite 222).

Zur Unterstützung der Vorlagenentwickler wird mir FirstSpirit Version 5.0 innerhalb der Formular-Register eine Code-Vervollständigung eingeführt. **Kapitel 7** erklärt, wie über diese Code-Vervollständigung alle FirstSpirit-Eingabekomponenten sowie die zugehörigen Parameter per Tastendruck angezeigt und eingefügt werden können.

1.2 Einordnung in die Gesamt-Dokumentation

Einige, insbesondere für die Vorlagenentwicklung erforderlichen Bereiche sind bereits ausführlich in der FirstSpirit-Online-Dokumentation dokumentiert. Die Einordnung der Entwicklerdokumentation in die Gesamtdokumentation veranschaulicht Abbildung 1-1.

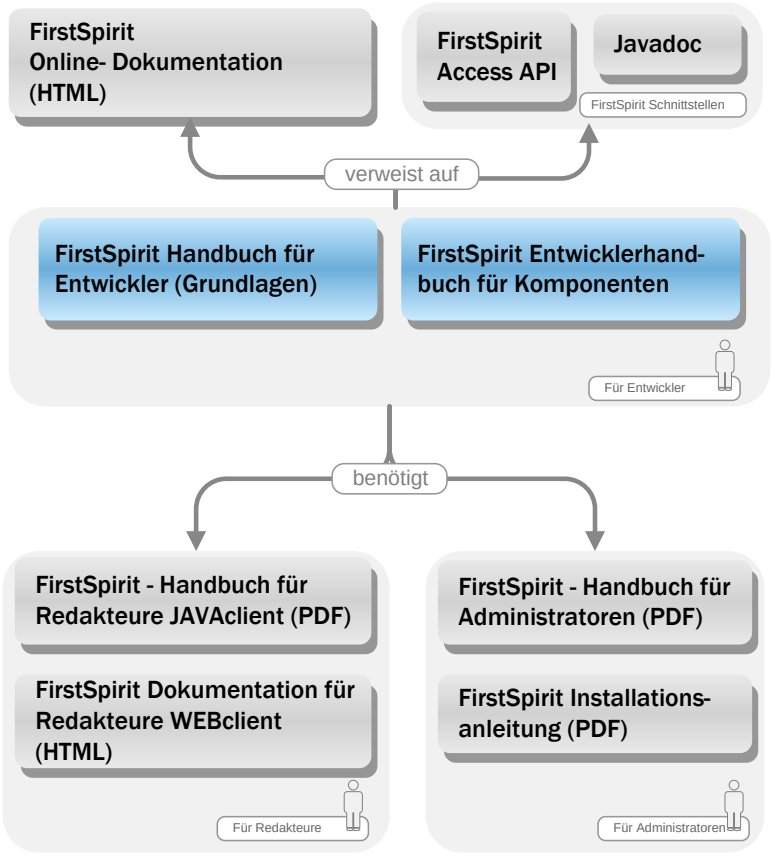


Abbildung 1-1: Einordnung der Entwicklerdokumentation in Gesamtdokumentation

Zum Verständnis der folgenden Kapitel werden zumindest grundlegende Kenntnisse des Handbuchs für Redakteure und des Handbuchs für Administratoren

vorausgesetzt. Eine detaillierte Beschreibung einzelner Vorlagen-Bestandteile und der Schnittstellen erfolgt in der FirstSpirit-Online-Dokumentation.

Aufgrund des Umfangs ist die Dokumentation für Entwickler unterteilt in dieses Handbuch, das die grundlegenden Aspekte der Vorlagen-Entwicklung erläutert, und ein Entwicklerhandbuch für Komponenten, das spezielle Aspekte rund um die Entwicklung von Modulen und Komponenten für FirstSpirit beschreibt.

Das Handbuch für Entwickler (Grundlagen) behandelt die in Kapitel 1.1 (Seite 8) genannten Aspekte.

Das Entwicklerhandbuch für Komponenten behandelt u. a. folgende Aspekte:

- Installation und Konfiguration von Modulen
- Aufbau von Modulen und Komponenten
- Das FirstSpirit GUI Object Model GOM
- Die FirstSpirit Security Architektur
- Beispiel-Listings

Für das Verständnis der Kapitel, die Erweiterungen und Anpassungen von FirstSpirit beschreiben, sind zudem Kenntnisse in den folgenden Bereichen hilfreich:

- Programmieren in Java / BeanShell
- Technik relationaler Datenbanken

1.3 Allgemeine Begriffe

1.3.1 Vorlagen (Templates)

Vorlagen bilden die Grundlage für jeden Webauftritt. In ihnen ist das komplette Layout der Webseite berücksichtigt (u.a. Corporate Design und Corporate Identity). Vorlagen werden benötigt, um beim Generieren der Webseite, die in der Inhalte-Verwaltung eingegebenen Inhalte und die in der Medien-Verwaltung eingebundenen Medien, mit der in der Struktur-Verwaltung hinterlegten Struktur, zu einer vollständigen Präsentation zu verbinden.

Die Grundlagen der Vorlagenentwicklung werden in einer detaillierten Schritt-für-Schritt-Anleitung in der **FirstSpirit Online Dokumentation** vermittelt. Dabei wird die Erstellung der ersten Vorlagen anhand eines einfachen Beispiels erläutert. Die Ausgabesprache ist HTML (siehe FirstSpirit Online Dokumentation / Kapitel Grundlagen / Schritt für Schritt).

In FirstSpirit stehen dem Entwickler verschiedene Arten von Vorlagen zur Verfügung:

- **Seitenvorlagen** erstellen das Grundgerüst einer Seite. In den Seitenvorlagen werden z. B. die Platzierungen von Logos und Navigationen sowie ähnliche allgemeine Einstellungen festgelegt. Außerdem definieren die Seitenvorlagen, an welchen Stellen ein Redakteur Inhalte einfügen kann.
- **Absatzvorlagen** werden verwendet, um Inhalte in dieses Grundgerüst einzufügen. Absatzvorlagen sind in individuell vorgegebene Eingabefelder unterteilt, über die der Redakteur den redaktionellen Inhalt des Absatzes (in der Inhalte-Verwaltung) pflegt.
- Über **Formatvorlagen** werden Formatierungen definiert, die anschließend im Eingabeelement DOM-Editor in der Inhalte-Verwaltung verwendet werden können.
- Über **Verweisvorlagen** wird das Aussehen von Verweisen (Links) innerhalb eines FirstSpirit-Projekts detailliert vorgegeben. Die Vorlagenentwickler definieren dabei sämtliche Eingabefelder, über die die Redakteure alle erforderlichen Inhalte eingeben können und die Darstellung des Verweises auf der HTML-Seite.

Alle Vorlagenarten werden in der Vorlagen-Verwaltung von FirstSpirit gepflegt und verwaltet.

Mit FirstSpiritVersion 5.0 werden als Lesehilfe an vielen Stellen der Vorlagen-Verwaltung Zeilennummern angezeigt. Mithilfe des Tastaturkürzels STRG + L können sie im JavaClient ein- und ausgeblendet werden.

1.3.2 Neue Eingabekomponenten

Im Rahmen der mit der FirstSpirit Version 4.2 begonnenen grundlegenden Überarbeitung und Konsolidierung des Eingabekomponenten-Modells (vgl. "FirstSpirit Roadmap 2009-2012") wurden eine Reihe von bisher getrennt realisierten Eingabekomponenten zusammen geführt.

Diese Zusammenfassung betrifft die folgenden Eingabekomponenten-Gruppen:

- einwertige Eingabekomponenten: Verweise auf andere FirstSpirit-Objekte, z. B. CMS_INPUT_FILE, CMS_INPUT_PICTURE, CMS_INPUT_PAGEREF, usw.
- mengenwertige Eingabekomponenten: CMS_INPUT_CONTENTLIST, CMS_INPUT_TABLIST, CMS_INPUT_CONTENTAREALIST, CMS_INPUT_LINKLIST

Die neue und mit der FirstSpirit Version 5.0 offiziell freigegebene Generation von Eingabekomponenten ist mit dem Namenspräfix „FS_“ anstelle dem bisherigen

„CMS_INPUT_“ gekennzeichnet.

Weiterführende Informationen zu den neuen Eingabekomponenten siehe auch FirstSpirit Online-Dokumentation[1].

1.3.3 Datenquellen-Verwaltung

Die Datenquellen-Verwaltung dient der Erfassung und Verwaltung stark strukturierter Datenbestände, die in FirstSpirit genutzt bzw. gepflegt werden sollen. Solche Datenbestände sind z. B. Produktkataloge und Adresslisten. Diese Datenbestände sind häufigen Änderungen unterworfen. Üblicherweise werden solche Daten in Datenbanken erfasst. Die Speicherung der Daten der Datenquellen-Verwaltung erfolgt in einem von FirstSpirit unterstützten relationalen Datenbanksystem.

Für die Datenerfassung in der Datenquellen-Verwaltung soll auch weiterhin die Trennung von Layout, Inhalt und Struktur gelten. Um dies zu gewährleisten, wird im Bereich Schemata (in der Vorlagen-Verwaltung) sowohl die Struktur der Daten als auch das Layout für die zugehörige Datenerfassungsmaske festgelegt. Mithilfe dieses Layouts werden anschließend in der Datenquellen-Verwaltung die Inhalte in Datenbank-Tabellen verwaltet. In der Struktur-Verwaltung können diese Datenbestände dann abschließend in die Struktur der Webseite eingefügt werden.

In einem ersten Schritt wird in der Vorlagen-Verwaltung über einen grafischen Editor ein Datenbankschema erstellt. Dieses Schema kann entweder auf Basis einer bestehenden Datenbankstruktur angelegt und – falls nötig – im Schema-Editor angepasst werden oder als leeres Schema erzeugt werden, um es dann mithilfe des Schema-Editors zu gestalten. In diesem Schema sind die Tabellen und Beziehungen eines Datenmodells abzubilden. In den Tabellenvorlagen werden anschließend Eingabeelemente für die Tabellenspalten definiert und Abfragen für die Datenbestände formuliert.

In der Datenquellen-Verwaltung erfolgt die Pflege der Datenbestände durch die zuständigen Redakteure. Hierzu werden Datenbank-Tabellen basierend auf den Einstellungen in der Vorlagen-Verwaltung erstellt und über die konfigurierten Eingabeelemente mit Inhalten gefüllt.

Zur Abbildung der strukturierten Inhalte auf einer Webseite wird die Datenbank-Tabelle als Datenquelle auf einer Seite der Inhalte-Verwaltung eingefügt. Diese Seite

[1] ../Vorlagenentwicklung/Formulare/Eingabekomponenten (neu)

wird anschließend in der Struktur-Verwaltung referenziert. Auf dieser Seitenreferenz können Einstellungen für die Anzeige der Datensätze vorgenommen werden. Soll beispielsweise nur ein bestimmter Ausschnitt der Datenbank-Tabelle dargestellt werden, können an dieser Stelle die in der Vorlagen-Verwaltung definierten Abfragen aufgerufen werden.

Alle Menüs der Datenquellen-Verwaltung werden in der "Dokumentation für Redakteure (JavaClient)" beschrieben.

Das Konzept zum Arbeiten mit "Schemata, Tabellenvorlagen, Sichten auf eine Datenbank" *siehe Kapitel 3 Seite 117.*

1.3.4 Arbeitsabläufe (Workflows)

Ein Arbeitsablauf ist eine Abfolge von Aufgaben, die nach einer fest vorgegebenen Struktur abgearbeitet werden. Die Aufgaben dienen dazu ein Objekt, beispielsweise eine Seite aus der Inhalte-Verwaltung, von einem Startzustand (z. B. "Seite geändert") in einen Endzustand (z. B. "geänderte Seite geprüft und freigegeben") zu überführen. Für die zwischen diesen Zuständen auszuführenden Aufgaben können sowohl Fälligkeitszeitpunkte als auch berechtigte Personengruppen festgelegt werden.

Die Arbeitsabläufe lassen sich mithilfe eines grafischen Editors in der Vorlagen-Verwaltung abbilden. Die Aufgabe des Arbeitsablauf-Editors ist es, den Arbeitsablauf so abstrakt und vollständig wie möglich zu beschreiben. Das grafisch erstellte Modell kann anschließend als Grundlage für die Unterstützung des Anwenders bei der Durchführung des Arbeitsprozesses verwendet werden.

Die Struktur (Abfolge der Aufgaben) und die Eigenschaften (z. B. kontextfrei) eines Arbeitsablaufs und die Definition der berechtigten Personen bzw. Gruppen, die einen Arbeitsablauf von einer Aufgabe in die nachfolgende Aufgabe weiterschalten dürfen, werden innerhalb der Vorlagen-Verwaltung definiert (siehe Kapitel 4 Seite 123).

Ein Beispiel für einen FirstSpirit Arbeitsablauf ist der häufig verwendete Freigabeprozess. Die Aufgabe des Freigabeprozesses ist es, sicherzustellen, dass ein vom Redakteur neu erstellter Beitrag oder eine Änderung der Inhalte vor der "Live-Schaltung" noch einer Prüfung unterzogen wird. Je nach dem, welche Arbeitsabläufe im Unternehmen bereits etabliert sind oder etabliert werden sollen, kann der Freigabeprozess variieren.

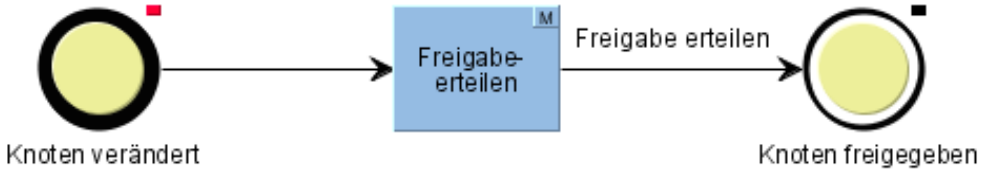


Abbildung 1-2: Beispiel "einfache" Freigabe

In diesem Beispiel ist der Chef-Redakteur für die Kontrolle der Beiträge der Redakteur verantwortlich. Erst, wenn er die Änderungen geprüft hat, ist eine Veröffentlichung möglich (siehe Abbildung 1-2).

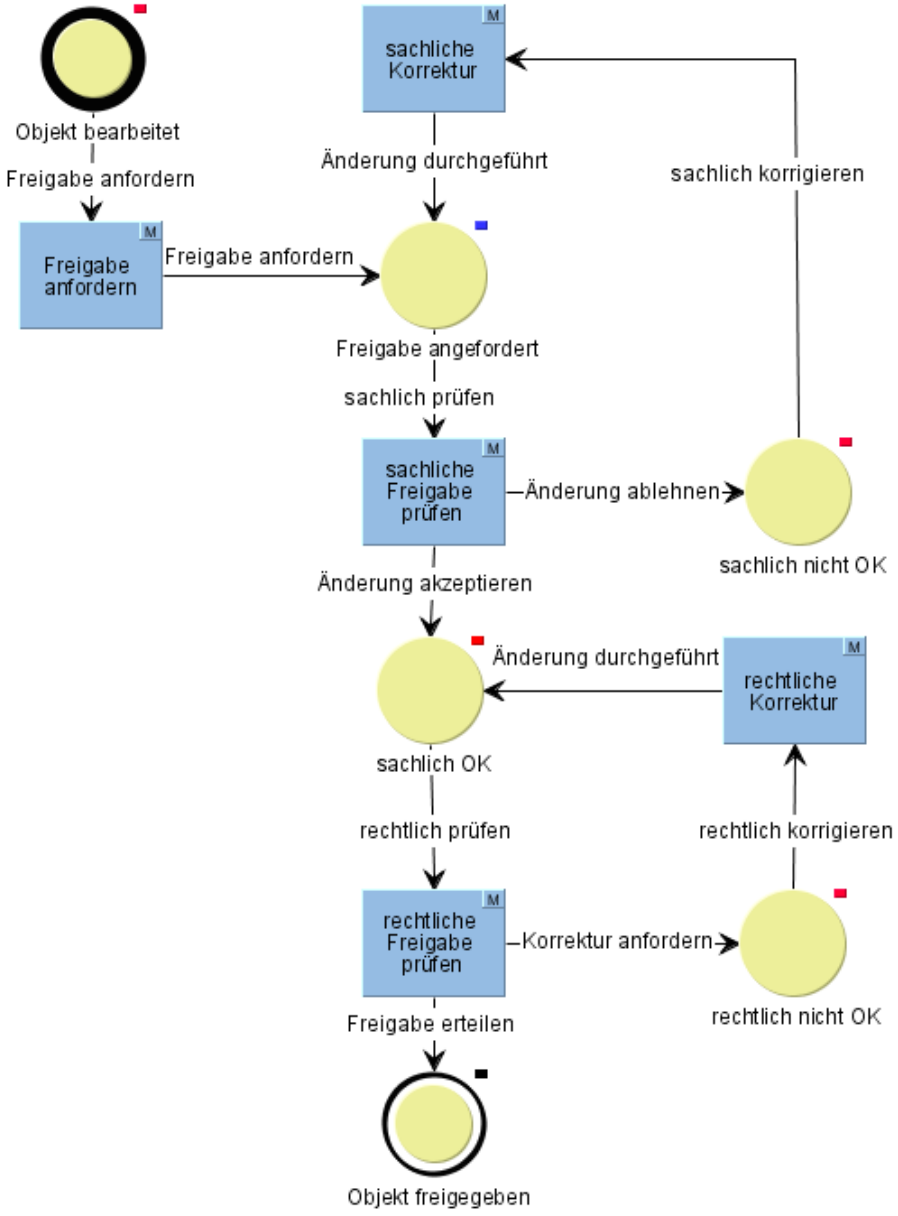


Abbildung 1-3: Beispiel: Freigabe mit "sachlicher und rechtlicher" Prüfung

In diesem Beispiel ist die Prüfung der zur Veröffentlichung bestimmten Beiträge in einen "sachlichen" (also inhaltlichen) und einen "rechtlichen" (also juristischen) Teilschritt gegliedert (siehe Abbildung 1-3). Diese Teilschritte werden in der Regel von unterschiedlichen Personen durchgeführt. In diesem Fall wird über den Arbeitsablauf auch sichergestellt, dass die rechtliche Prüfung erst nach der inhaltlichen Prüfung erfolgt, so dass gegebenenfalls notwendige inhaltliche Korrekturen auch diesen Prüfschritt durchlaufen. Sollte eine Korrektur unter rechtlichen Aspekten notwendig sein, geht das Modell in diesem Fall davon aus, dass eine erneute inhaltliche Prüfung nicht notwendig ist (dies könnte in anders gelagerten Anwendungsfällen aber durchaus notwendig sein).

Wesentliche Aspekte beim Einsatz von Arbeitsabläufen sind unter anderem:

- Zuordnung von Arbeitsabläufen zu "logischen" Teilbereichen:
  Beispiel: In der Medien-Verwaltung ist in Bereich "A" nur der Arbeitsablauf "B" und "C" möglich, während in der Struktur-Verwaltung der Arbeitsablauf "E" zu verwenden ist.

- Zuordnen von Benutzern/Gruppen und Arbeitsabläufen:
  Beispiel: Der Arbeitsablauf "B" darf nur von der Benutzergruppe "G" ausgeführt werden.

- Definieren von Rechten in Arbeitsabläufen:
  Beispiel: Der Übergang in den Status "rechtlich geprüft" kann nur von der Gruppe "Juristen" durchgeführt werden.

- Definieren von Datenfeldern, die beim Durchlaufen des Arbeitsablaufs vom Benutzer ausgefüllt werden können (oder müssen):
  Beispiel: Einfügen eines "Rechtlichen Prüfvermerks" in das entsprechende Formularfeld durch die prüfende Person.

Weitere Informationen zum Erstellen von Arbeitsabläufen siehe Kapitel 4, Seite 123 ff.

1.3.5 Integrierte Vorschau

Der JavaClient bietet mit der Funktion "Integrierte Vorschau" (Menü "Ansicht" / "Integrierte Vorschau anzeigen") eine direkte WYSIWYG-Vorschau. Für die Vorlagenentwicklung kann sie genutzt werden, um Änderungen im Ausgabekanal der Vorlagen (z. B. HTML oder XSL-FO) direkt im Vorschaufenster zu prüfen, da bei jedem Speichern der Vorlage automatisch die (konfigurierbare) Vorschau-Seite

aktualisiert wird.

Außerdem steht innerhalb der Vorlagen-Verwaltung eine integrierte Formular-Vorschau zur Verfügung. Wird der Formularbereich eine Vorlage selektiert, so erscheint im Vorschaubereich eine Live-Ansicht des bearbeiteten Formulars mit den definierten Rückgriffswerten der Eingabekomponenten (siehe Abbildung 1-4).

Die integrierte Vorschau kann wahlweise rechts neben dem Arbeitsbereich oder in einem externen Fenster angezeigt werden.

Innerhalb der integrierten Formularvorschau können alle Eingabefelder direkt bearbeitet werden. Dazu muss die Vorlage nicht zum Bearbeiten gesperrt werden.

Die Bearbeitungsmöglichkeit stellt lediglich eine Hilfe für den Vorlagenentwickler dar. Darüber kann beispielsweise direkt geprüft werden, ob eine definierte Remote-Konfiguration für eine Eingabekomponente die gewünschten Ergebnisse liefert. Die eingetragenen Inhalte werden in der Formularansicht jedoch nicht gespeichert. Vorgabewerte können in der Formular-Vorschau nicht definiert werden

Die **Vorgabewerte für die Eingabekomponenten** können über den Button "Vorgabewerte" im Register "Eigenschaften" einer Vorlage definiert werden (siehe Kapitel 2.5.2 Seite 55). Die sprachabhängigen Vorgabewerte werden direkt nach dem Speichern der Eigenschaften im Vorschaubereich angezeigt.

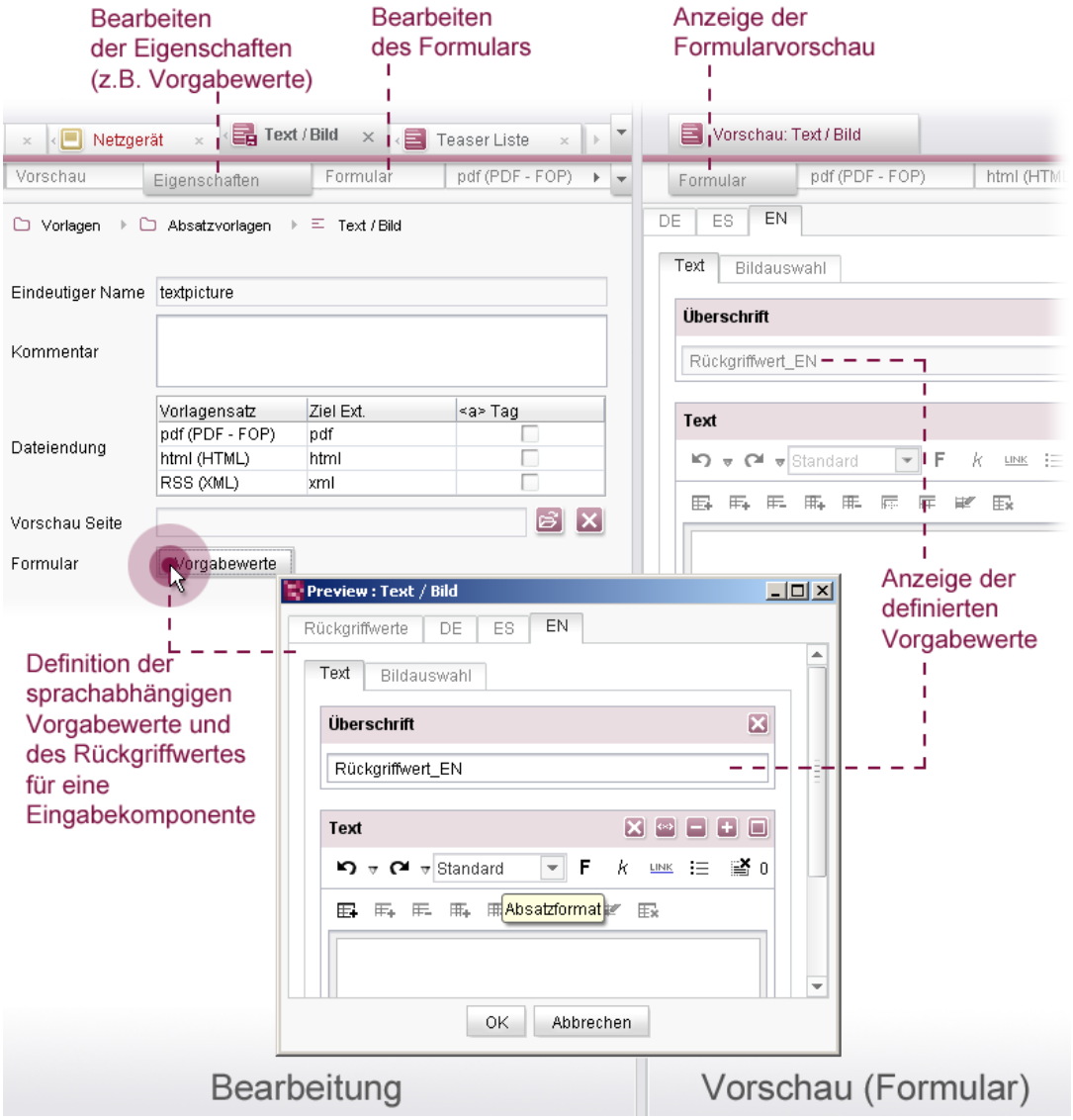


Abbildung 1-4: Vorschau des Formularbereichs in der Vorlagen-Verwaltung

1.3.6 Content Highlighting & EasyEdit

Um das Bearbeiten von redaktionellen Inhalten für den Redakteur möglichst einfach und transparent zu gestalten, bietet FirstSpirit die Funktion „Content Highlighting“ an. Diese Funktionalität verknüpft die redaktionelle Pflege im FirstSpirit JavaClient mit der Ausgabe in der Vorschau, indem Inhalte automatisch aufgerufen und zum Bearbeiten hervorgehoben werden. Content Highlighting funktioniert bidirektional.

Um diese Funktion in einem Projekt zu verwenden, müssen zunächst die Vorlagen angepasst werden. Die HTML-Elemente, die hervorgehoben werden sollen, müssen innerhalb einer HTML-Seite eindeutig referenzierbar sein. Dafür werden sogenannte „Editor-Identifier“ eingesetzt, die vom Vorlagenentwickler innerhalb der HTML-

Vorlagen hinterlegt werden können.

Da das Content-Highlighting mit FirstSpirit Version 5.0 vereinfacht wurde, ist eine Anpassung von bereits bestehenden Projekten, die noch die alte Funktionalität verwenden, zwingend notwendig. Ohne diese Änderung kann eine fehlerfreie Verwendung des Content-Highlightings nicht gewährleistet werden.

Die Vorlagenanpassungen, die zum Verwenden der Funktionalität Content Highlighting notwendig sind, können sich auch im WebClient 5 auswirken und werden dort für das redaktionelle Bearbeiten von Inhalten eingesetzt. Analog zum JavaClient werden die Inhalte im WebClient ebenfalls farblich hervorgehoben und mit einem Rahmen versehen. Innerhalb dieses Rahmens werden Bedienelemente eingeblendet, die es dem Redakteur ermöglichen, die umrahmten Inhalte direkt zu editieren (EasyEdit-Funktionalität).

Eine genaue Beschreibung der Formatvorlagen und des Stylesheets befindet sich in der FirstSpirit Online-Dokumentation[2].

Die Technologien, die für die Funktionalität "Content-Highlighting" eingesetzt werden, greifen stärker als bisherige Funktionalitäten in den HTML-Quellcode eines FirstSpirit-Projektes ein. Es kann nicht garantiert werden, dass "Content Highlighting" ohne Änderungen am Projekt-HTML eingesetzt werden kann. Besonders "pixelgenaue" Layouts in Verbindung mit CHTML können hier zu Problemen führen, da durch die Umrahmung der hervorgehobenen Inhalte einige zusätzliche Pixel im HTML-Umfeld benötigt werden.

1.3.7 Zentrale Fehlersammlung und Systemreport

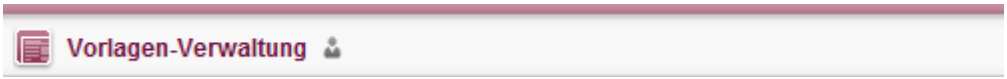
Es wird eine Infrastruktur zur Sammlung von Fehlern und Exceptions bereitgestellt. Dazu wird im linken unteren Bereich des JavaClients ein Loader-Icon eingeblendet, das kontinuierlich die Datenübertragung beim redaktionellen Arbeiten anzeigt. Beim Auftreten einer Exception wird ein kleines Ausrufezeichen im Icon angezeigt. Weitere Informationen zum aufgetretenen Fehler können dann mit einem Klick auf das Icon angefordert werden.

Weitere Informationen zu dieser Funktion siehe FirstSpirit Handbuch für Redakteure (JavaClient), Unterkapitel "Zentrale Fehlersammlung und Systemreport" im Kapitel 3 "Menüs und Icons des FirstSpirit JavaClients".

[2] FirstSpirit Online-Dokumentation - Kapitel: ../Weiterführende Themen/Content Highlighting

2 Vorlagen-Verwaltung des FirstSpirit JavaClients

2.1 Allgemeines



Vorlagen bilden die Grundlage für jeden Webauftritt. In ihnen ist das vollständige Layout der Webseite berücksichtigt (u.a. Corporate Design und Corporate Identity). Vorlagen werden benötigt, um beim Generieren der Webseite, die in der Inhalte-Verwaltung eingegebenen Inhalte und die in der Medien-Verwaltung eingebundenen Medien mit der in der Struktur-Verwaltung hinterlegten Struktur zu einer vollständigen Präsentation zu verbinden.

In der Vorlagen-Verwaltung können unterschiedliche Arten von Vorlagen definiert und bearbeitet werden.

- Seitenvorlagen (siehe Kapitel 2.5 Seite 53)
- Absatzvorlagen (siehe Kapitel 2.6 Seite 61)
- Formatvorlagen (siehe Kapitel 2.7 Seite 63)
- Stilvorlagen (siehe Kapitel 2.8 Seite 67)
- Tabellenformatvorlagen (siehe Kapitel 2.9 Seite 77)
- Verweisvorlagen (siehe Kapitel 2.10 Seite 86)

Außerdem bieten sich weitere Bearbeitungsmöglichkeiten für:

- Skripte (siehe Kapitel 2.11 Seite 89)
- Datenbank-Schemata (siehe Kapitel 2.12 Seite 94)
- Arbeitsabläufe (siehe Kapitel 2.13 Seite 116)

Verschieben per ‚Drag & Drop': Ordner und Vorlagen können innerhalb der Vorlagen-Verwaltung mithilfe der Maus per "Drag & Drop" verschoben werden (erkennbar durch ein kleines Rechteck am Mauszeiger).

Kopieren per ‚Drag & Drop': Weiterhin ist es in der Vorlagen-Verwaltung möglich, Ordner und Vorlagen mithilfe der Maus und gleichzeitig gedrückter Strg-Taste zu kopieren (gekennzeichnet durch ein kleines Plus am Mauszeiger).

Bei "Drag & Drop" (verschieben, kopieren) von Knoten in der Vorlagen-Verwaltung muss der Benutzer die erforderlichen Rechte besitzen. Andernfalls erscheint eine Fehlermeldung "Die Aktion kann nicht ausgeführt werden (fehlende Rechte)!"

Mit FirstSpiritVersion 5.0 werden als Lesehilfe an vielen Stellen der Vorlagen-Verwaltung Zeilennummern angezeigt. Mithilfe des Tastaturkürzels STRG + L können sie im JavaClient ein- und ausgeblendet werden.

2.2 Allgemeine Kontextmenüs der Vorlagen-Verwaltung

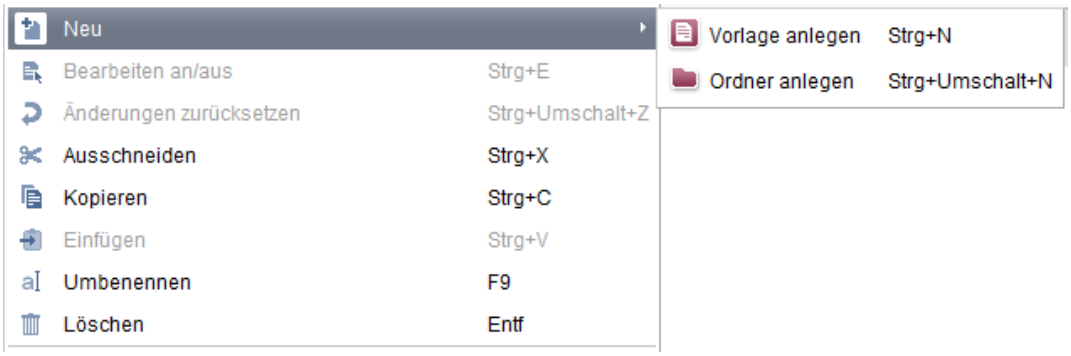


Abbildung 2-1: Allgemeines Kontextmenü der Vorlagen-Verwaltung

In den folgenden Kapiteln werden die Kontextmenüs der Vorlagen-Verwaltung beschrieben:

Strukturierung der Kontextmenüs (allgemein, spezifisch, administrativ):
Kontextmenüs sind alle nach dem gleichen Schema strukturiert:

- im obersten Bereich befinden sich allgemeine Funktionalitäten (siehe dieses Kapitel)
- im mittleren Bereich befinden sich spezifische Funktionen für den ausgewählten Knoten (siehe Kapitel 2.3 Seite 32).
- im unteren Bereich befinden sich Funktionen, die in der Regel nur von Projektadministratoren benötigt werden. Diese können durch den normalen Benutzer normalerweise nicht ausgeführt werden und sind daher meist deaktiviert (siehe Punkt 3) (siehe Kapitel 2.4 Seite 45).

Aufrufen eines Kontextmenüs: Um ein Kontextmenü aufzurufen, wird ein Objekt, beispielsweise ein Ordner oder eine Vorlage, in der Baumansicht der linken Bildschirmhälfte markiert und dann mit einem Klick auf die rechte Maustaste das

Kontextmenü zu diesem Knoten geöffnet. Mit einem Klick der linken Maustaste kann der gewünschte Menüpunkt ausgewählt werden.

Deaktivierte Menüpunkte werden "grau" dargestellt. In diesem Fall steht die Funktionalität dem Benutzer nicht zur Verfügung. Mögliche Gründe dafür sind:

- das Objekt wird momentan durch einen anderen Vorlagenentwickler bearbeitet
- der Status des aktuellen Objekts
- dem Benutzer fehlen die Rechte, um eine bestimmte Aktion ausführen zu können.

2.2.1 Neu

Abbildung 2-2: Funktion Neu

Über den Kontextmenüeintrag "Neu" oder über das Icon "Neu" in der Symbolleiste können neue Objekte in das Projekt eingefügt werden. Die zur Verfügung stehende Auswahl ist abhängig vom Objekttyp, auf dem das Kontextmenü bzw. die Funktion aufgerufen wurde.

Insgesamt lassen sich diese Objekte in der Vorlagen-Verwaltung erstellen:

- Seiten- und Absatz-Vorlagen
- Ordner
- Formatvorlagen
- Tabellenformatvorlagen
- Stilvorlagen
- Verweisvorlagen
- Skripte
- Schemata
- Schemata aus Datenbanken
- Tabellenvorlagen
- Abfragen
- Arbeitsabläufe

Die Funktion zum Anlegen eines neuen Objekts öffnet einen Dialog, welcher die folgende Struktur aufweist (siehe Abbildung 2-3):

- Anzeigenamen
- Referenznamen

- Zusatzinformationen (wenn diese benötigt werden)

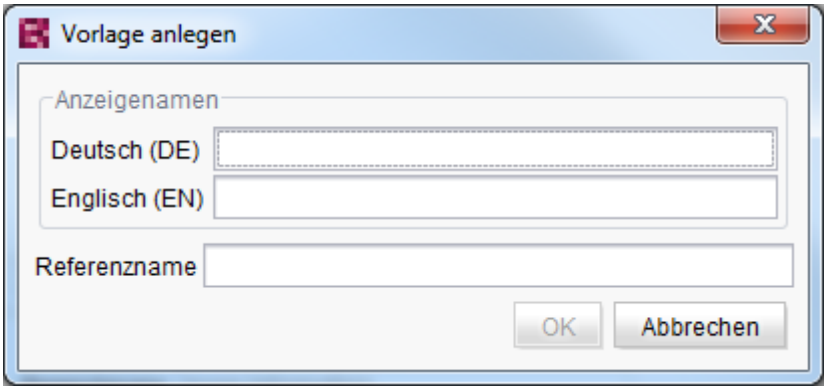


Abbildung 2-3: Vorlage anlegen

Anzeigename: Der sprachabhängige Anzeigename kann einzeln für alle Projektsprachen definiert werden. Ist die sprachabhängige Anzeige der Baumansicht im JavaClient aktiviert (Menü Ansicht / Bevorzugte Anzeigesprache), werden die hier eingetragenen Anzeigenamen abhängig von der ausgewählten Projektsprache angezeigt. Im Gegensatz zum eindeutigen Referenznamen können die sprachabhängigen Bezeichnungen jederzeit geändert werden.

Referenzname: Im Feld "Referenzname" wird ein eindeutiger Name für die neue Vorlage angegeben. Beim Anlegen wird das Feld für die Eingabe des Referenznamens parallel zur Eingabe eines Anzeigenames automatisch gefüllt. Dazu wird der (bei mehreren Sprachen der zuerst eingetragene) Anzeigename in das Feld übernommen. Leer- und Sonderzeichen werden dabei durch Unterstriche ersetzt. Über die Anzeige des Referenznamen (Menü Ansicht / Bevorzugte Anzeigesprache / Referenznamen im Baum anzeigen) lässt sich der Referenzname auch in der Baumansicht des JavaClients darstellen. Der Referenzname kann beim Anlegen vom Benutzer geändert werden. Nach dem Anlegen ist das Ändern des Referenznamens über das Kontextmenü „Extras“ möglich. Eine nachträgliche Änderung des Referenznames wird jedoch nicht empfohlen, da ansonsten alle Bezüge innerhalb des Projekts verloren gingen.

Über den Referenznamen kann innerhalb eines Projekts ein eindeutiger Bezug zur Vorlage hergestellt werden. Innerhalb der Eingabekomponenten (im Formularbereich) wird beispielsweise der eindeutige Referenzname verwendet, um Bezüge zu Vorlagen herzustellen (vgl. FirstSpirit-Online-Dokumentation, z. B. zur Eingabekomponente CMS_INPUT_DOM).

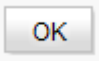
Wird bei der Erstellung eines neuen Objekts ein Referenzname eingetragen, der innerhalb eines Namensraums bereits vergeben wurde, wird der Name von FirstSpirit automatisch durch einen eindeutigen Namen ersetzt, zumeist durch Anhängen einer Nummerierung. (Ungültige Sonderzeichen werden ebenfalls

automatisch ersetzt.)

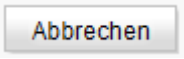
Bsp.: Es existiert bereits eine Seitenvorlage *template*. Eine neu angelegte Absatzvorlage *template* würde dann automatisch unter dem Referenznamen *template_1* gespeichert.

Der Referenzname einer Vorlage kann im Register "Eigenschaften" ermittelt werden (siehe Kapitel 2.5.2 Seite 55).

Ein Referenzname einer Vorlage sollte nach dem Anlegen nicht mehr geändert werden, da ansonsten alle Bezüge innerhalb des Projekts verloren gingen!



Mit einem Klick auf den Button wird die neue Vorlage in den Verzeichnisbaum eingehängt und kann weiter bearbeitet werden.



Mit einem Klick auf den Button wird der Vorgang abgebrochen. Eine neue Vorlage wird nicht angelegt.

Besonderheiten

Ordner

Anders als die Referenznamen von Vorlagen (vgl. Kapitel 2.2.1) müssen Ordnernamen nicht eindeutig vergeben werden.

Tabellenvorlagen

Für jede eingepflegte Tabelle im Datenbankmodell muss unterhalb des Schemas eine Tabellenvorlage angelegt werden. In diesen Tabellenvorlagen wird festgelegt, über welche Eingabekomponenten der Redakteur später die Daten in die entsprechenden Tabellen einpflegen kann und an wo diese Daten in der Datenbank gespeichert werden sollen.

Um eine neue Tabellenvorlage zu erstellen, wird zusätzlich zu den Anzeige- und dem Referenznamen die Auswahl einer Tabelle des Datenbankschemas benötigt (vgl. Abbildung 2-4), für welche die Vorlage erstellt werden soll (siehe auch: Kapitel 2.12.1 Seite 95)

Enthält das Schema keine Tabellen, ist die Combobox leer. Es kann dann keine Tabellenvorlage erstellt werden.

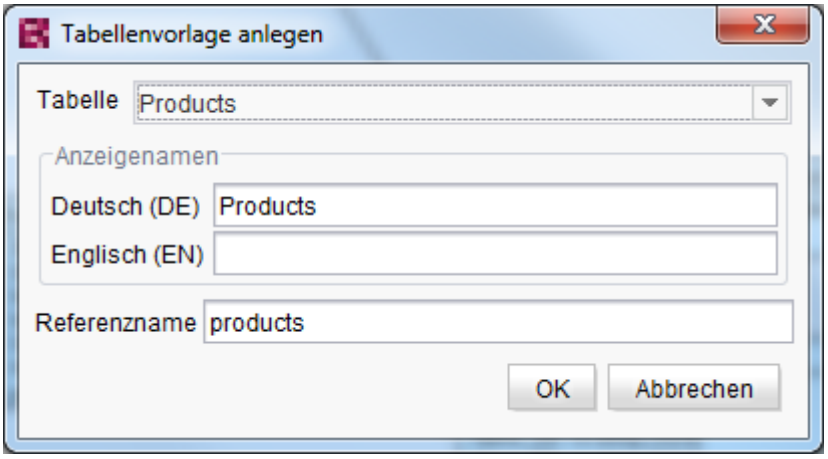


Abbildung 2-4: Tabellenvorlage anlegen

2.2.2 Bearbeiten an/aus

Abbildung 2-5: Funktion Bearbeiten an/aus

Um Änderungen an einem Objekt vornehmen zu können, ist es zunächst notwendig, den Bearbeitungsmodus einzuschalten. Dies beugt dem gleichzeitigen Bearbeiten durch einen anderen Benutzer vor und verhindert Konflikte, die beim gleichzeitigen Ändern eines Elements entstehen können.

Über den Kontextmenüeintrag "Bearbeiten an/aus" oder über das Icon "Bearbeiten an/aus" in der Symbolleiste können neue Objekte zum Bearbeiten gesperrt werden.

Nachdem die gewünschten Änderungen durchgeführt wurden, muss der Bearbeitungsmodus wieder ausgeschaltet werden, um das entsprechende Objekt zum Bearbeiten für andere Benutzer freizugeben. Bei Beendigung des Bearbeitungsmodus werden automatisch alle vorgenommenen Änderungen gespeichert.

2.2.3 Änderungen zurücksetzen

Abbildung 2-6: Funktion Änderungen zurücksetzen

Mit dieser Funktion können Änderungen rückgängig gemacht werden, die während des aktuellen Bearbeitungsvorgangs vorgenommen und *noch nicht gespeichert* worden sind. Damit nicht versehentlich Inhalte gelöscht werden, erfolgt vor dem Zurücksetzen eine Sicherheitsabfrage.

Die Funktion ist nur aktiv, wenn der Bearbeitungsmodus auf einem Objekt aktiviert ist (vgl. Kapitel 2.2.2).

Diese Funktion kann auf folgenden Elementen der Vorlagen-Verwaltung verwendet werden:

- Seiten- und Absatz-Vorlagen
- Format- (Ausnahme: System-Formatvorlagen), Tabellenformat- und Stilvorlagen
- Verweisvorlagen
- Skripte
- Datenbank-Schemata
- Tabellenvorlagen
- Abfragen
- Arbeitsabläufe

2.2.4 Ausschneiden

Abbildung 2-7: Funktion Ausschneiden

Mithilfe dieser Funktion kann ein Objekt an der aktuellen Baumposition ausgeschnitten und in der Zwischenablage abgelegt werden.

Die Funktion "Ausschneiden" wird erst dann ausgeführt, wenn das ausgeschnittene Objekt wieder eingefügt wird. Wird ein ausgeschnittenes Objekt nicht wieder eingefügt, bleibt es an der ursprünglichen Baumposition erhalten, wird also nicht gelöscht.

Über die Funktion Einfügen (siehe Kapitel 2.2.6) können diese Objekte dann an anderer Stelle wieder eingefügt werden.

Diese Funktion kann auf folgenden Elementen der Vorlagen-Verwaltung verwendet werden:

- Allen Ordnern der Vorlagen-Verwaltung
- Seiten- und Absatz-Vorlagen
- Format- (Ausnahme: System-Formatvorlagen und Ordner, die System-Formatvorlagen enthalten), Tabellenformat- und Stilvorlagen
- Skripte
- Datenbank-Schemata
- Tabellenvorlagen
- Abfragen
  Arbeitsabläufe

2.2.5 Kopieren

Abbildung 2-8: Funktion Kopieren

Mithilfe dieser Funktion wird eine Kopie des aktuellen Objektes erzeugt und in der Zwischenablage abgelegt. Über die Funktion Einfügen (siehe Kapitel 2.2.6) können diese Objekte dann an anderer Stelle wieder eingefügt werden.

Diese Funktion kann auf folgenden Elementen der Vorlagen-Verwaltung verwendet werden:

- Allen Ordnern der Vorlagen-Verwaltung
- Seiten- und Absatzvorlagen
- Format-, Tabellenformat- und Stilvorlagen
- Skripte
- Datenbank-Schemata
- Tabellenvorlagen
- Abfragen
- Arbeitsabläufe

Kopieren per ‚Drag & Drop': Weiterhin ist es in der Vorlagen-Verwaltung möglich, Ordner und Vorlagen mithilfe der Maus und gleichzeitig gedrückter Strg-Taste zu kopieren (gekennzeichnet durch ein kleines Plus am Mauszeiger).

Bei "Drag & Drop" (verschieben, kopieren) von Knoten in der Vorlagen-Verwaltung muss der Benutzer die erforderlichen Rechte besitzen. Andernfalls erscheint eine Fehlermeldung "Die Aktion kann nicht ausgeführt werden (fehlende Rechte)!"

2.2.6 Einfügen

Abbildung 2-9: Funktion Einfügen

Mithilfe dieser Funktion wird der Inhalt der Zwischenablage an der aktuellen Position der Baumstruktur eingefügt. Diese Funktion ist nur dann aktiv, wenn sich Daten in der Zwischenablage befinden, die an der aktuellen Position eingefügt werden dürfen.

Die Zwischenablage ist ein Bereich innerhalb des JavaClients, an welchem unterschiedlichste Objekte (Seiten, Seitenreferenzen, Bilder, Datensätze, Absätze, einzelne Eingabekomponenten, Texte, Dateien) ablegt werden können. Sie dient dem Redakteur als „Sammelbecken“, in dem Materialien und Inhalte zentral und übersichtlich für spätere Arbeitsschritte erfasst werden können.

Ausführlichere Informationen zur Zwischenablage finden Sie in den Releasenotes zu FirstSpirit 5.0.

Objekte dürfen nur in den dazu vorgesehenen Bereichen eingefügt werden. Es ist nicht möglich, beispielsweise eine Absatzvorlage unterhalb des Knotens Seitenvorlagen einzufügen. In diesem Fall ist der Eintrag "Einfügen" deaktiviert.

Diese Funktion kann auf folgenden Elementen der Vorlagen-Verwaltung verwendet werden:

- Allen Wurzelknoten der Vorlagen-Verwaltung
- Allen Ordnern der Vorlagen-Verwaltung
- Datenbank-Schemata (nur für Tabellenvorlagen)

2.2.7 Umbenennen

Abbildung 2-10: Funktion Umbenennen

Mit dieser Funktion ist es möglich, die sprachabhängigen Anzeigenamen des aktuellen Objektes in der Baumstruktur des FirstSpirit JavaClients zu ändern. Nach Aufruf der Funktion öffnet sich ein Fenster mit dem bisherigen Anzeigenamen, diese können nun geändert werden.

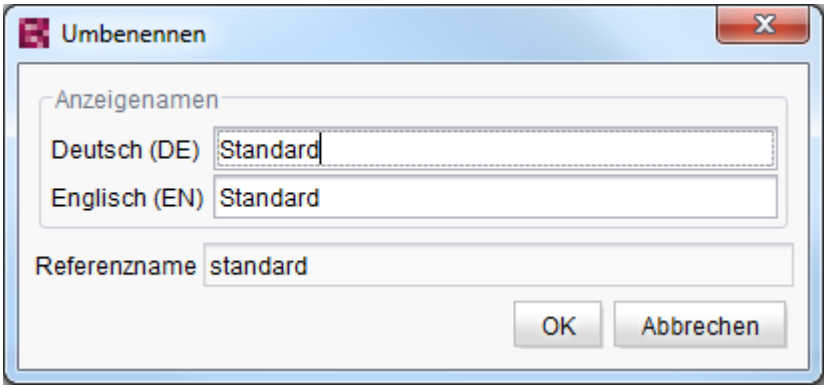


Abbildung 2-11: Umbenennen

Eindeutige Referenznamen dürfen nicht geändert werden, da ansonsten die Bezüge zu diesem Objekt im Projekt verloren gehen (z. B. Referenznamen von Absatz- oder Seitenvorlagen). Das Feld "Referenzname" ist in diesem Fall deaktiviert und kann nicht bearbeitet werden. Ordner besitzen keinen eindeutigen Referenznamen. Dieser Name kann daher jederzeit umbenannt werden.

Diese Funktion kann auf folgenden Elementen der Vorlagen-Verwaltung verwendet werden:

- Allen Ordnern der Vorlagen-Verwaltung
- Seiten- und Absatzvorlagen
- Format, Tabellenformat- und Stilvorlagen
- Verweisvorlagen
- Skripte
- Datenbank-Schemata
- Tabellenvorlagen

- Abfragen
- Arbeitsabläufe

2.2.8 Löschen

Abbildung 2-12: Funktion Löschen

Mit dieser Funktion ist es möglich, das aktuell markierte Objekt oder den aktuell markierten Teilbaum zu löschen. Versehentliches Löschen wird durch eine Sicherheitsabfrage unterbunden.

Für Projektadministratoren besteht die Möglichkeit, gelöschte Elemente wieder herzustellen (siehe Kapitel 2.3.4 Seite 41).

Die Funktion "Objekte löschen" steht auf folgenden Elementen zur Verfügung:

- Seitenvorlagen
- Absatzvorlagen (siehe Kapitel 2.2.8.1)
- Format-, Tabellenformat- und Stilvorlagen (System-Formatvorlagen können nicht gelöscht werden)
- Verweisvorlagen
- Skripte
- Arbeitsabläufe
- Datenbank-Schemata, Abfragen und Tabellenvorlagen

Die Funktion "Teilbäume löschen" steht auf folgenden Elementen zur Verfügung:

- Allen Ordnern der Vorlagen-Verwaltung

Weiterführende Informationen zum Löschen von Objekten und Teilbäumen siehe FirstSpirit Handbuch für Redakteure, Kapitel 3.2.8 "Löschen".

2.2.8.1 Löschen von verwendeten Objekten

Diese Funktion ist nur für Projekt- und Serveradministratoren verfügbar.

Vorlagen, die noch von anderen Objekten innerhalb des Clients verwendet werden, können ebenfalls gelöscht werden. In diesem Fall wird dem Benutzer vor dem

Löschen eines einzelnen Objekts eine Sicherheitsfrage angezeigt.

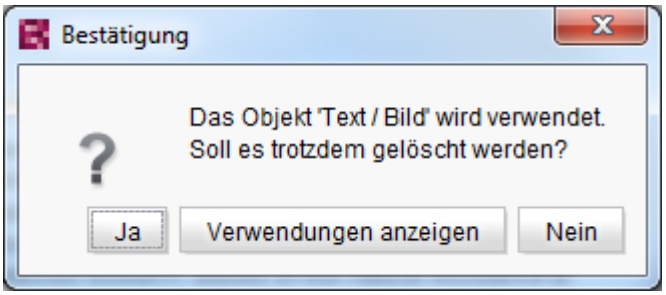
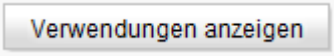


Abbildung 2-13: ein verwendetes Objekt löschen



Mit einem Klick auf den Button öffnet sich der Dialog "Das Objekt wird noch referenziert", in dem die Objekte angezeigt werden, die das zu löschende Objekt aktuell noch verwenden (siehe Abbildung 2-14).

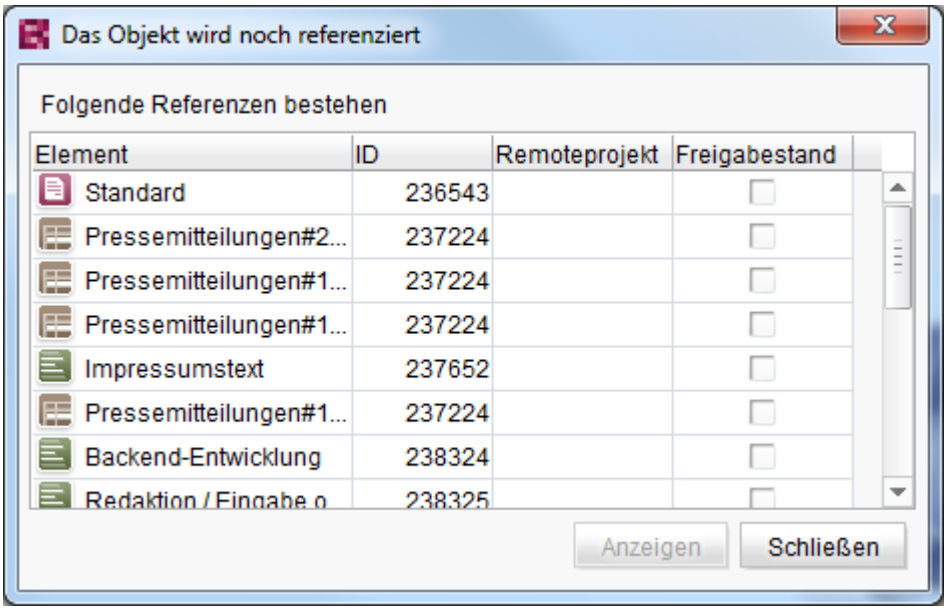


Abbildung 2-14: das Objekt wird noch referenziert

Mit einem Klick auf den Button wird das zu löschende Objekt trotz der bestehenden Verwendungen im Projekt gelöscht.

Im Referenzgraph wird die Referenz auch nach der Löschung noch dargestellt. Allerdings ist das gelöschte Objekt nun unbekannt. Die Referenz wird erst nach einer erneuten Bearbeitung des noch bestehenden Objekts aufgehoben.

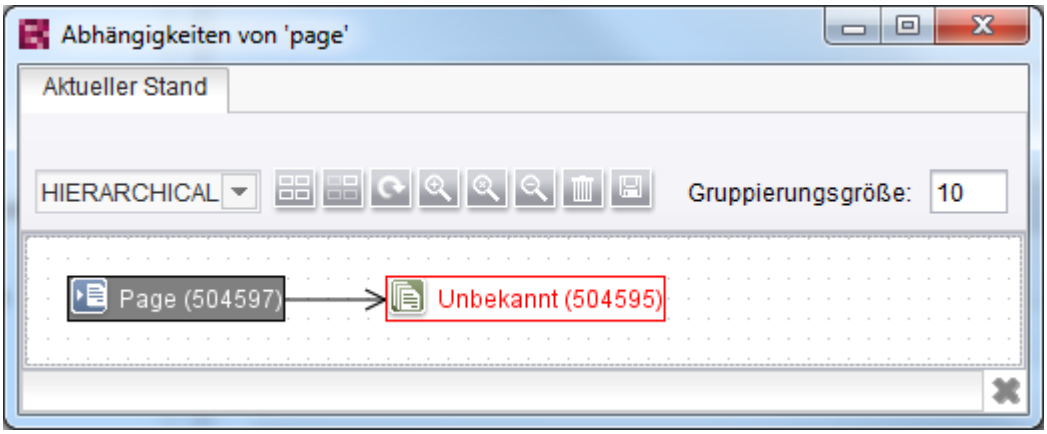


Abbildung 2-15: Referenzgraph nach der Löschung eines noch verwendeten Objekts

Vor dem Löschen mehrerer Objekte, von denen mindestens eines noch in Verwendung ist, wird dem Benutzer ebenfalls eine Sicherheitsabfrage angezeigt, die sich jedoch von dem Dialog einer Einzel-Löschung unterscheidet (siehe Abbildung 2-16).

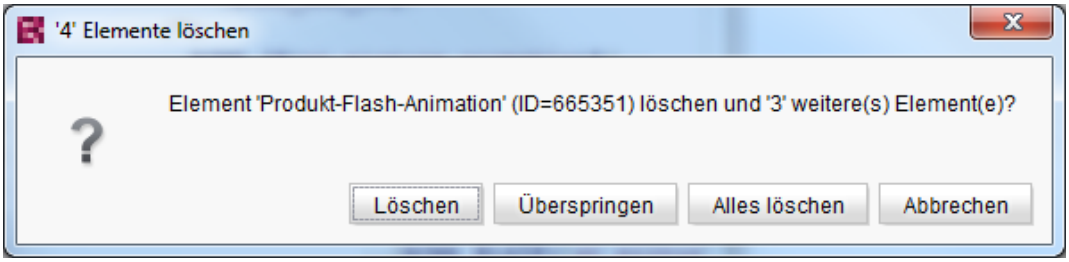


Abbildung 2-16: mehrere verwendete Objekte löschen

In diesem Fall lässt sich die Verwendung der entsprechenden Objekte nicht visualisieren. Über die Button „Löschen“ und „Alles Löschen“ kann der Löschvorgang differenziert werden.

Ein Klick auf „Alles Löschen“ löscht die ausgewählten Objekte gesammelt, nach einer zuvorigen Abfrage, ob verwendete Objekte ebenfalls gelöscht oder übersprungen werden sollen.

Ein Klick auf „Löschen“ entspricht der bisherigen Löschung mehrerer Objekte. In diesem Fall muss die Löschung für jedes der ausgewählten Objekte einzeln bestätigt werden.

Handelt es sich bei einem zu löschenden Objekt um eine Absatzvorlage, die als Absatzeinschränkung im Inhaltsbereich einer Seitenvorlage verwendet wird, so wird diese Einschränkung beim Löschen der Absatzvorlage automatisch aus der Liste der erlaubten Absatzvorlagen für den Inhaltsbereich entfernt. Handelte es sich dabei um die einzige Absatzeinschränkung des Inhaltsbereichs, sind anschließend nach einem

Refresh der Seitenvorlage alle Absatzvorlagen für diesen Inhaltsbereich erlaubt (siehe auch Kapitel 2.5.2 Seite 55).

2.3 Spezielle Kontextmenüs der Vorlagen-Verwaltung

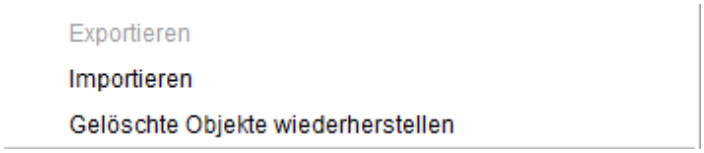


Abbildung 2-17: Spezielle Kontextmenüs – Seitenvorlagen (Wurzel)

Im mittleren Bereich des Kontextmenüs sind spezielle Funktionen für das jeweilige Objekt sowie eventuell auch lizenzabhängige Funktionen verfügbar. Die Funktionen sind also stark abhängig vom ausgewählten Objekttyp bzw. von dem Umfang der Lizenz.

2.3.1 Aktualisieren

Abbildung 2-18: Aktualisieren

Diese Funktion kann auf folgenden Elementen der Vorlagen-Verwaltung verwendet werden:

- Wurzelknoten der Vorlagen-Verwaltung

Mithilfe dieses Menüeintrags kann die Ansicht der Vorlagen-Verwaltung aktualisiert werden. Dies ist erforderlich, wenn mehrere Personen gleichzeitig an einem Projekt arbeiten und Änderungen vornehmen.

Wird ein Objekt aktuell bearbeitet und die Änderungen sind noch nicht gespeichert, darf diese Funktion nicht verwendet werden! Ansonsten werden die nicht gespeicherten Änderungen durch die Version des Objekts auf dem Server überschrieben und gehen damit verloren.

2.3.2 Exportieren

Abbildung 2-19: Funktion Exportieren

Über den Kontextmenüeintrag "Exportieren" können Objekte aus einem Projekt zu einer komprimierten Zip-Datei zusammengefasst und im lokalen Dateisystem gespeichert werden. Die Export-Dateien können anschließend verwendet werden, um die exportierten Inhalte eines Projekts (Quellprojekt) in andere FirstSpirit-Projekte (Zielprojekt) zu importieren (siehe Kapitel 2.3.3 Seite 36). Die zur Verfügung stehende Auswahl ist abhängig vom Objekttyp, auf dem das Kontextmenü bzw. die Funktion aufgerufen wurde.

Das im FirstSpirit JavaClient verfügbare Kontextmenü "Exportieren" ist eine clientseitige Funktion und stellt daher bei großen Datenmengen erhebliche Anforderungen an den Hauptspeicher des Client-Systems. Diese Funktion sollte also nur für den Export von kleineren Datenmengen verwendet werden.

Diese Funktion kann auf folgenden Elementen der Vorlagen-Verwaltung verwendet werden:

- Allen Ordnern der Vorlagen-Verwaltung (siehe Kapitel 2.3.2.1 Seite 34)
- Seiten- und Absatz-Vorlagen (siehe Kapitel 2.3.2.2 Seite 34)
- Formatvorlagen (keine System-Formatvorlagen)(siehe Kapitel 2.3.2.2 Seite 34)
- Stil- und Tabellenformatvorlagen (siehe Kapitel 2.3.2.3 Seite 34)
- Verweisvorlagen (siehe Kapitel 2.3.2.3 Seite 34)
- Skripte (siehe Kapitel 2.3.2.3 Seite 34)
- Datenbank-Schemata (siehe Kapitel 2.3.2.4 Seite 34)
- Tabellenvorlagen (siehe Kapitel 2.3.2.5 Seite 35)
- Abfragen (siehe Kapitel 2.3.2.5 Seite 35)
- Arbeitsabläufe (siehe Kapitel 2.3.2.6 Seite 35)

Beim Aufruf des Kontextmenüs öffnet sich zunächst ein Exportfenster zur Auswahl des gewünschten Speicherorts für die Exportdatei im lokalen Dateisystem des Arbeitsplatzrechners.

2.3.2.1 Ordner exportieren

Ordner können exportiert und zur Verwendung in anderen FirstSpirit-Projekten dort wieder importiert werden. Die Verzeichnisstruktur aus dem Zielprojekt wird beim Exportieren beibehalten.

2.3.2.2 Vorlagen exportieren

Vorlagen können exportiert und zur Verwendung in anderen FirstSpirit-Projekten dort wieder importiert werden. Sollen häufig Vorlagen aus einem Quellprojekt exportiert und in ein Zielprojekt importiert werden, wird die Verwendung der lizenzabhängigen Funktion „CorporateContent“ empfohlen, da dabei erweiterte Informationen für die Zuordnung und zum Aktualisierungsstatus im Zielprojekt zur Verfügung stehen.

2.3.2.3 Stil- und Tabellenformatvorlagen, Verweisvorlagen und Skripte exportieren

Das Exportieren von Stil- und Tabellenformatvorlagen sowie von Verweisvorlagen und Skripten funktioniert analog zum Exportieren von Vorlagen (siehe Kapitel 2.3.2.2 Seite 34).

Stil- und Tabellenformatvorlagen sind eng miteinander verbunden (siehe auch Kapitel 2.8 und 2.9 ab Seite 67) und sollten daher möglichst gemeinsam exportiert werden. Dazu werden sie am besten in einem Ordner zusammengefasst, der sich, wie in Kapitel 2.3.2.1 beschrieben, exportieren lässt. Stilvorlagen können aber auch problemlos einzeln exportiert und später wieder importiert werden. Zu Tabellenformatvorlagen müssen auch immer die als Standard Stilvorlage und die in den Darstellungsregeln verwendeten Stilvorlagen mit exportiert werden.

2.3.2.4 Schema exportieren

Schemata können exportiert und zur Verwendung in anderen FirstSpirit-Projekten dort wieder importiert werden (siehe Kapitel 2.3.3.1 Seite 37).

Nach der Anzeige des Exportfensters, in dem aus den lokalen Verzeichnissen des Arbeitsplatzrechners der gewünschte Speicherort für die Exportdatei auswählen werden kann, wird der folgende Dialog angezeigt:

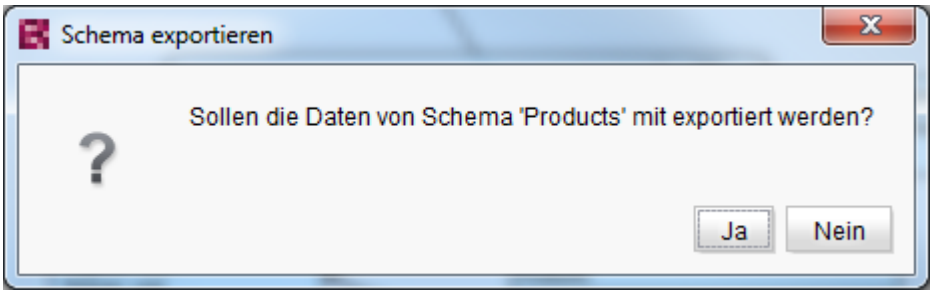
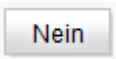


Abbildung 2-20: Daten beim Export eines Schemas exportieren

Mit einem Klick auf den Button werden die Daten des Schemas der Datenquellen-Verwaltung des aktuellen Projekts zur Export-Datei hinzugefügt. Beim Importieren der Export-Datei in ein weiteres FirstSpirit-Projekt stehen diese Daten dann den Anwendern des zweiten Projekts zur Verfügung.



Mit einem Klick auf den Button wird nur das Schema, nicht aber die Daten des Schemas aus der Datenquellen-Verwaltung des aktuellen Projekts zur Export-Datei hinzugefügt. Beim Importieren der Export-Datei in ein weiteres FirstSpirit-Projekt steht den Anwendern des zweiten Projekts zwar das Schema, nicht aber die Daten aus der Datenquellen-Verwaltung des ersten Projekts zur Verfügung.

Die zum Schema zugehörigen Tabellenvorlagen werden automatisch zur Export-Datei hinzugefügt.

2.3.2.5 Tabellenvorlagen und Abfragen exportieren

Wird ein Schema exportiert, werden die zugehörigen Tabellenvorlagen (und Abfragen) automatisch zur Export-Datei hinzugefügt. Tabellenvorlagen (und Abfragen) sollten möglichst immer zusammen mit dem zugehörigen Schema exportiert werden. Ist das nicht möglich, können diese auch einzeln exportiert werden.

In diesem Fall muss die Vorlage (und/oder Abfrage) im Zielprojekt auf dem passenden Schemaknoten importiert werden. Andernfalls kann es zu Fehlern im Projekt kommen, da das Mapping der Tabellenvorlage nicht mehr zu den Tabellen des Schemas passt ("Die referenzierte Tabelle 'xy' existiert nicht").

2.3.2.6 Arbeitsabläufe exportieren

Mithilfe dieser Funktion können sowohl einzelne Arbeitsabläufe als auch Ordner mit allen enthaltenen Unterordnern und Arbeitsabläufen in das Dateisystem des

Rechners exportiert werden. So können Arbeitsabläufe zu einem späteren Zeitpunkt z. B. in anderen Projekten verwendet werden.

Hierzu öffnet sich ein Exportfenster, in dem in den lokalen Verzeichnissen des Arbeitsplatzrechners der gewünschten Speicherort für die Exportdatei ausgewählt werden kann.

Werden innerhalb eines Arbeitsablaufs Skripte verwendet, so können diese ebenfalls der Exportdatei hinzugefügt werden. Dazu wird zunächst der gewünschte Arbeitsablauf und anschließend mit Strg + Klick das Skript in der Baumansicht selektiert. Wird nun über das Kontextmenü die Funktion "Exportieren" gewählt, sind beide Objekte in der Zip-Datei enthalten. Skripte können jedoch auch zu einem späteren Zeitpunkt gesondert exportiert werden (siehe Kapitel 2.3.2.3 Seite 34).

2.3.3 Importieren

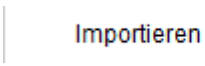


Abbildung 2-21: Funktion Importieren

Über den Kontextmenüeintrag "Importieren" können zuvor exportierte Objekte aus einem Quellprojekt in ein weitere FirstSpirit-Projekt (Zielprojekt) eingefügt werden. Dazu muss zunächst die gewünschte Zip-Datei aus dem lokalen Dateisystem des Arbeitsplatzrechners ausgewählt und an passender Stelle im Zielprojekt importiert werden.

Passen die importierten Inhalte nicht in den Kontext des Zielprojekts werden diese – sofern das möglich ist – automatisch im richtigen Kontext des Zielprojekts importiert. Der Import wird in diesem Fall unabhängig vom Objekt, auf dem das Kontextmenü "Importieren" ausgewählt wurde, ausgeführt. Wird beispielsweise versucht, die Exportdatei eines Skriptes in den Bereich "Arbeitsabläufe" zu importieren, wird das ausgewählte Skript dennoch in das Zielprojekt importiert, allerdings nicht in den Bereich "Arbeitsabläufe" sondern in den korrekten Bereich "Skripte" des Zielprojekts.

Diese automatische Korrektur greift nicht in allen Fällen. Kann nicht zugeordnet werden, zu welchem Objekt des Zielprojekts die Import-Datei passt, wird stattdessen eine Fehlermeldung angezeigt.

Die zur Verfügung stehende Auswahl ist abhängig vom Objekttyp, auf dem das Kontextmenü bzw. die Funktion aufgerufen wurde.

Diese Funktion kann auf folgenden Elementen der Vorlagen-Verwaltung verwendet werden:

- Allen Ordnern der Vorlagen-Verwaltung
- Stil- und Tabellenformatvorlagen (siehe Kapitel 2.3.3.1 Seite 37)
- Verweisvorlagen
- Datenbank-Schemata (siehe Kapitel 2.3.3.2 Seite 39)
- Arbeitsabläufe (siehe Kapitel 2.3.3.3 Seite 41)

Mithilfe dieser Funktion können im Dateisystem exportierte Seiten-/Absatzvorlagen aber auch exportierte Ordner mit allen enthaltenen Vorlagen und Unterordnern in den ausgewählten Ordner importiert werden.

Hierzu öffnet sich ein Importfenster, in dem in den lokalen Verzeichnissen des Arbeitsplatzrechners nach der gewünschten Exportdatei gesucht werden kann.

Zuordnung der Vorlagensätze im Zielprojekt: Beim Import einer Vorlage aus einem Quellprojekt in ein Zielprojekt, wird versucht, auch die Inhalte der Vorlagensätze zu übernehmen:

- Dabei wird zunächst versucht eine Zuordnung anhand des Namens (*Name im Quellprojekt* zu *Name im Zielprojekt*) durchzuführen. Das heißt, sind die Namen der Vorlagensätze im Quell- und im Zielprojekt identisch, werden die Inhalte ins Zielprojekt übernommen.
- Gelingt eine Zuordnung anhand des Namens nicht, weil die Vorlagensätze im Zielprojekt anders benannt sind, wird im nächsten Schritt versucht eine Zuordnung anhand des Präsentationskanals durchzuführen (*Name im Quellprojekt* zu *Präsentationskanal im Zielprojekt*). Dabei wird beispielsweise ein Vorlagensatz mit dem Namen "html" aus dem Quellprojekt, dem ersten gefundenen Präsentationskanal "HTML" im Zielprojekt zugeordnet (unabhängig vom Namen des Kanals im Zielprojekt).
- Kann weder eine Zuordnung anhand des Namens noch anhand des Präsentationskanals erfolgen, können die Inhalte des Vorlagensätze nicht aus dem Quellprojekt in das Zielprojekt importiert werden und müssen ggf. manuell angelegt oder kopiert werden.

2.3.3.1 Stil- und Tabellenformatvorlagen importieren

Stil- und Tabellenformatvorlagen können aus anderen FirstSpirit-Projekten importiert werden. Dazu muss zunächst eine Export-Datei der gewünschten Vorlagen aus einem anderen FirstSpirit-Projekt exportiert werden (siehe Kapitel 2.3.2.3 Seite 34).

Stilvorlagen können problemlos einzeln importiert werden. Tabellenformatvorlagen sollten zusammen mit den verwendeten Stilvorlagen (siehe Kapitel 2.8 ab Seite 67) importiert werden. Werden diese Stilvorlagen nicht mit exportiert, können Tabellenformatvorlagen trotzdem importiert werden, Referenzen zu den Stilvorlagen gehen jedoch verloren.

Zum Import von Stil- und Tabellenformatvorlagen wird das Kontextmenü auf dem Wurzelknoten "Formatvorlagen" bzw. einem Ordner unterhalb dieses Wurzelknotens aufgerufen und die Funktion "Importieren" ausgewählt. Nach der Auswahl der gewünschten Export-Datei erscheint der Import-Dialog:

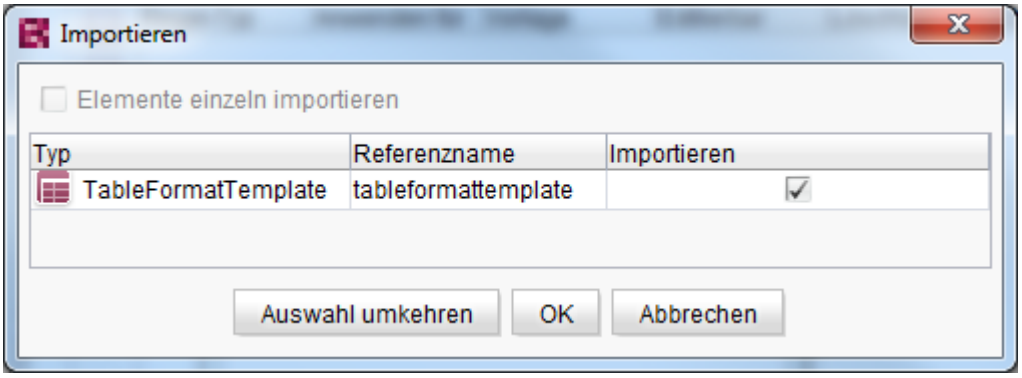


Abbildung 2-22: Tabellenformatvorlage importieren

Elemente einzeln importieren: Diese Funktionalität steht in der Vorlagen-Verwaltung nicht zur Verfügung.

Typ: Typ des in der Export-Datei enthaltenen Elements.

Referenzname: Name des in der Export-Datei enthaltenen Elements.

Importieren: Ist die Checkbox aktiviert, wird das betreffende Element ins Zielprojekt importiert, ist die Checkbox deaktiviert, wird das Element nicht importiert.

Auswahl umkehren Mit einem Klick auf den Button wird die innerhalb der Spalte "Importieren" getroffene Auswahl umgekehrt.

OK Mit einem Klick auf den Button wird die Auswahl des Dialogs bestätigt und der Dialog "Datenbank-Layer auswählen" öffnet sich (siehe Abbildung 2-24).

Abbrechen Mit einem Klick auf den Button wird der Import-Vorgang abgebrochen.

2.3.3.2 Schema importieren

Schemata können aus anderen FirstSpirit-Projekten importiert werden. Dazu muss zunächst eine Export-Datei des gewünschten Schemas aus einem weiteren FirstSpirit-Projekt exportiert werden (siehe Kapitel 2.3.2.4 Seite 34).

Die Export-Datei des ersten FirstSpirit-Projekts kann nun in das zweite Projekt importiert werden. Dazu wird das Kontextmenü auf dem Wurzelknoten "Datenbank-Schemata" bzw. einem Ordner unterhalb dieses Wurzelknotens aufgerufen und die Funktion "Importieren" ausgewählt. Nach der Auswahl der gewünschten Export-Datei erscheint der Import-Dialog:

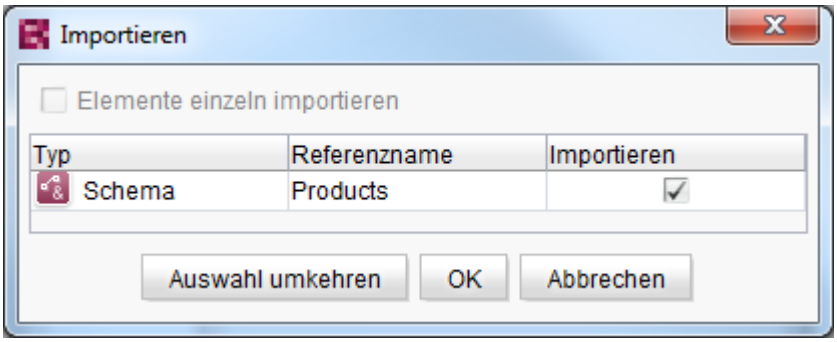


Abbildung 2-23: Schema mit Tabellenvorlagen importieren

Elemente einzeln importieren: Diese Funktionalität steht in der Vorlagen-Verwaltung nicht zur Verfügung.

Typ: Typ des in der Export-Datei enthaltenen Elements.

Name: Name des in der Export-Datei enthaltenen Elements.

Importieren: Ist die Checkbox aktiviert, wird das betreffende Element (und alle untergeordneten Elemente, z. B. Tabellenvorlagen) ins Zielprojekt importiert, ist die Checkbox deaktiviert, wird das Element nicht importiert.

Auswahl umkehren Mit einem Klick auf den Button wird die innerhalb der Spalte "Importieren" getroffene Auswahl umgekehrt.

OK Mit einem Klick auf den Button wird die Auswahl des Dialogs bestätigt und der Dialog "Datenbank-Layer auswählen" öffnet sich (siehe Abbildung 2-24).

Abbrechen Mit einem Klick auf den Button wird der Import-Vorgang abgebrochen.

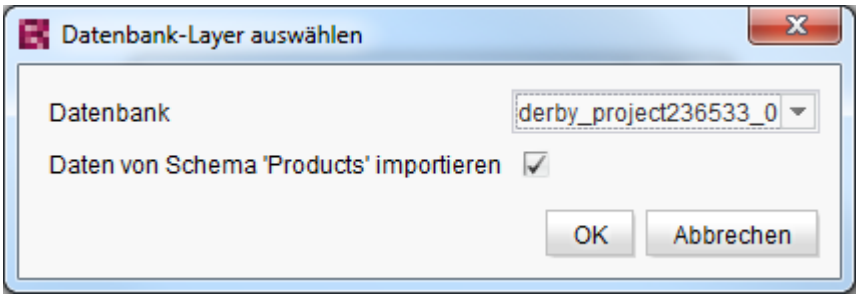


Abbildung 2-24: Schema importieren

Datenbank: Auswahl des gewünschten Datenbank-Layers. In der Klappliste werden alle Layer angezeigt, die vom Projektadministrator für das Projekt aktiviert wurden.

Daten von Schema ‚xy' importieren: Ist die Checkbox aktiviert, werden die bisherigen Daten, die zu diesem Schema in der Datenquellen-Verwaltung des Quellprojekts gepflegt wurden, in das Zielprojekt übernommen. Beim Anlegen einer Tabelle in der Datenquellen-Verwaltung des Zielprojekts, basierend auf den importierten Schema-Informationen, werden die bisher gepflegten, strukturierten Daten im Zielprojekt angezeigt:

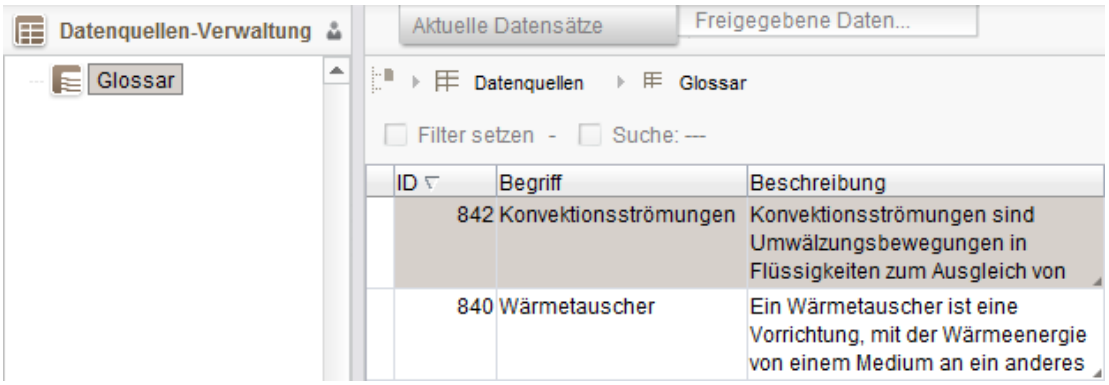


Abbildung 2-25: Tabellenansicht der strukturierten Daten im Zielprojekt

Die Daten können innerhalb des Zielprojekts auch geändert werden, sind also nicht schreibgeschützt.

Soll nur das Schema, nicht aber die bisherigen Daten, die zu diesem Schema in der Datenquellen-Verwaltung des Quellprojekts gepflegt wurden, übernommen werden, muss entweder der Export des Schemas im Quellprojekt ohne Daten erfolgen (siehe Abbildung 2-20) oder die Checkbox "Daten von Schema ‚xy' importieren" im Zielprojekt deaktiviert werden. In beiden Fällen werden die bisherigen Daten (basierend auf dem betreffenden Schema) beim Importieren ignoriert, sind also vor dem Zugriff aus dem Zielprojekt geschützt.

Sollen die bisherigen Daten aus dem Quellprojekt zwar im Zielprojekt angezeigt werden, die Änderung der strukturierten Daten aber unterbunden werden, muss der Schreibschutz auf dem ausgewählten Datenbank-Layer nach dem Import der Daten aktiviert werden.

2.3.3.3 Arbeitsabläufe importieren

Diese Funktion kann über die Kontextmenüs im Bereich Arbeitsabläufe und auf Ordnerebene aufgerufen werden.

Mithilfe dieser Funktion können im Dateisystem exportierte Arbeitsabläufe aber auch exportierte Ordner mit allen enthaltenen Unterordnern und Arbeitsabläufen (und ggf. enthaltene Skripte) in den ausgewählten Ordner importiert werden.

Hierzu öffnet sich ein Importfenster, in dem in den lokalen Verzeichnissen des Arbeitsplatzrechners nach der gewünschten Exportdatei gesucht werden kann.

Die Rechtekonfiguration muss beim Importieren ggf. projektspezifisch angepasst werden (siehe Kapitel 4.6 Seite 163).

2.3.4 Gelöschte Objekte wiederherstellen

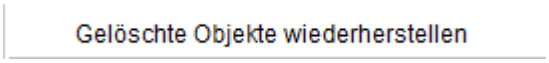


Abbildung 2-26: Gelöschte Objekte wiederherstellen

Die Funktion Gelöschte Objekte wiederherstellen kann bei Seiten-, Absatz-, Format-, Verweisvorlagen und Skripten sowie Arbeitsabläufen sowohl auf Wurzel- als auch auf Ordnerebene aufgerufen werden, bei Datenbank-Schemata zusätzlich auf Schemaebene. Wurde irrtümlich ein Objekt aus der Baumstruktur gelöscht, dann lässt es sich mithilfe dieser Funktion wiederherstellen. Nach dem Klicken öffnet sich ein Fenster mit den gelöschten Objekten.

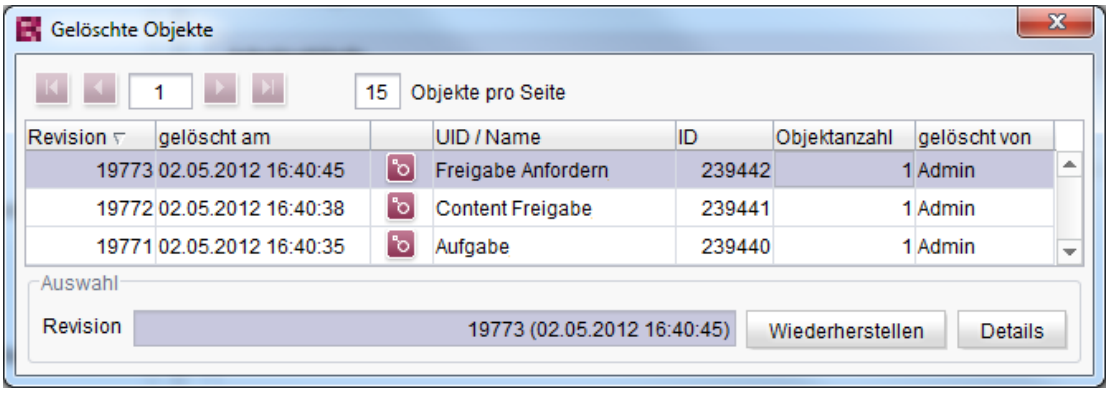


Abbildung 2-27: Gelöschte Objekte

Auf Wurzelebene werden alle Objekte angezeigt, von denen ein Backup vorhanden ist, während auf Ordnerebene lediglich die Objekte angezeigt werden, die sich unterhalb dieses Ordners befunden haben. Für jedes Objekt werden die folgenden Informationen angegeben:

Revision: Versionsnummer des gelöschten Objektes.

gelöscht am: Datum und Uhrzeit, an dem das Objekt gelöscht wurde.

UID / Name: Der Referenzname des gelöschten Objektes.

ID: Die eindeutige ID-Nummer des gelöschten Objektes.

Objektanzahl: Die Anzahl der Objekte, die sich in der Baumstruktur unterhalb des gelöschten Objektes befunden haben. Diese können über die Schaltfläche Details in einem Popup-Fenster angezeigt werden. Diese hierarchisch untergeordneten Objekte werden bei der Wiederherstellung ebenfalls wieder eingefügt.

gelöscht von: Name des Benutzers, der das Objekt gelöscht hat.

Zum Wiederherstellen muss lediglich das gewünschte Objekt ausgewählt und die Schaltfläche Wiederherstellen aktiviert werden.

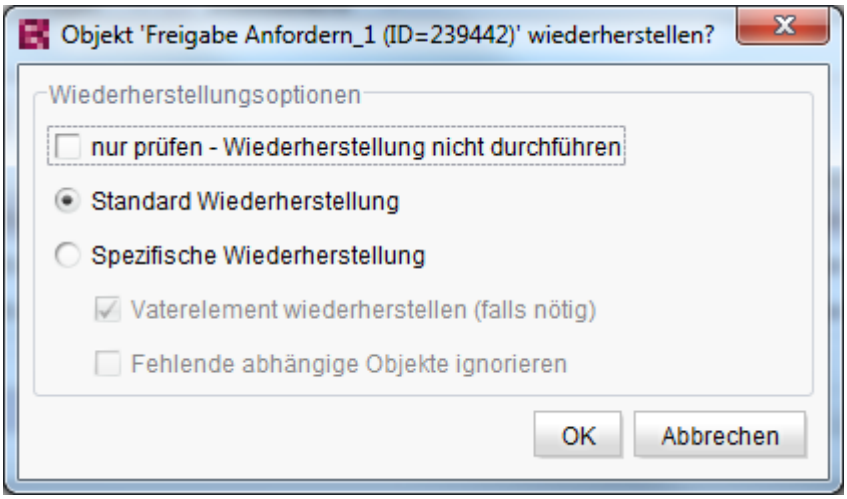


Abbildung 2-28: Gelöschte Objekte wiederherstellen

nur prüfen – Wiederherstellung nicht durchführen: Ist diese Option ausgewählt, wird überprüft, ob eine Wiederherstellung fehlerfrei durchgeführt werden kann. Dazu wird die Wiederherstellung simuliert, das gelöschte Objekt wird aber nicht wiederhergestellt. In einem Popup-Fenster wird anschließend angezeigt, ob eine Wiederherstellung möglich ist oder nicht.

Standard Wiederherstellung: Diese Option ist standardmäßig voreingestellt. Wird die Wiederherstellung mit dieser Option durchgeführt, wird die Wiederherstellung direkt objektabhängig durchgeführt. Je nach Objekt können daher im Bereich "Spezifische Wiederherstellung" unterschiedliche Optionen ausgewählt sein.

Spezifische Wiederherstellung: Diese Option kann ausgewählt werden, um die Optionen der Standard Wiederherstellung manuell anzupassen.

Spezifische Wiederherstellung – Vaterelement wiederherstellen (falls nötig): Ist diese Option ausgewählt, wird auch das Vaterelement wenn nötig wiederhergestellt.

Spezifische Wiederherstellung – Fehlende abhängige Objekte ignorieren: Ist diese Option ausgewählt, werden fehlende Referenzen zum ausgewählten Objekt bei der Wiederherstellung ignoriert.

Diese Option steht nur Projektadministratoren zur Verfügung.

Im nächsten Dialog kann dann die Position ausgewählt werden, an der das gelöschte Objekt eingefügt werden soll. In der Regel ist in diesem Dialog ein Unterordner als Wiederherstellungspunkt ausgewählt. Soll der Wurzelknoten einer

Verwaltung ausgewählt werden, muss der in der mittleren Spalte markierte Unterordner durch STRG und Klick auf den Unterordner abgewählt werden.

2.3.5 Extern bearbeiten

Abbildung 2-29: Funktion Extern bearbeiten

Diese Funktion kann über das Kontextmenü auf Seitenvorlagen- und Absatzvorlagen aufgerufen werden und untergliedert sich in mehrere Bereiche: Es werden alle **Vorlagensätze** aufgelistet, die in der Server- und Projektkonfiguration für dieses Projekt eingestellt wurden, außerdem gibt es noch die Bereiche **Formular** und **Regeln**.

Wird einer der vorhandenen Editierbereiche aktiviert, dann öffnet sich die entsprechende Quelldatei in einem externen Editor. Zur Bearbeitung einer Quelldatei in einem externen Editor sollte in den Benutzereinstellungen der Globalen Einstellungen ein Editor eingetragen sein. Zusätzlich erscheint noch ein weiteres Fenster, in dem alle geöffneten Vorlagen angezeigt werden.

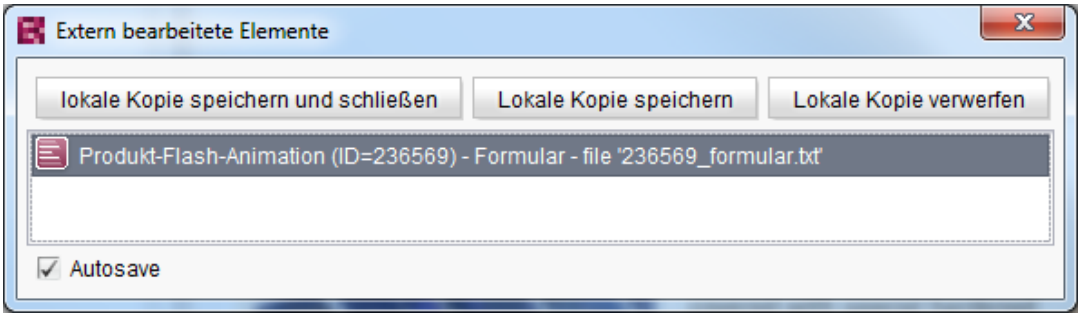
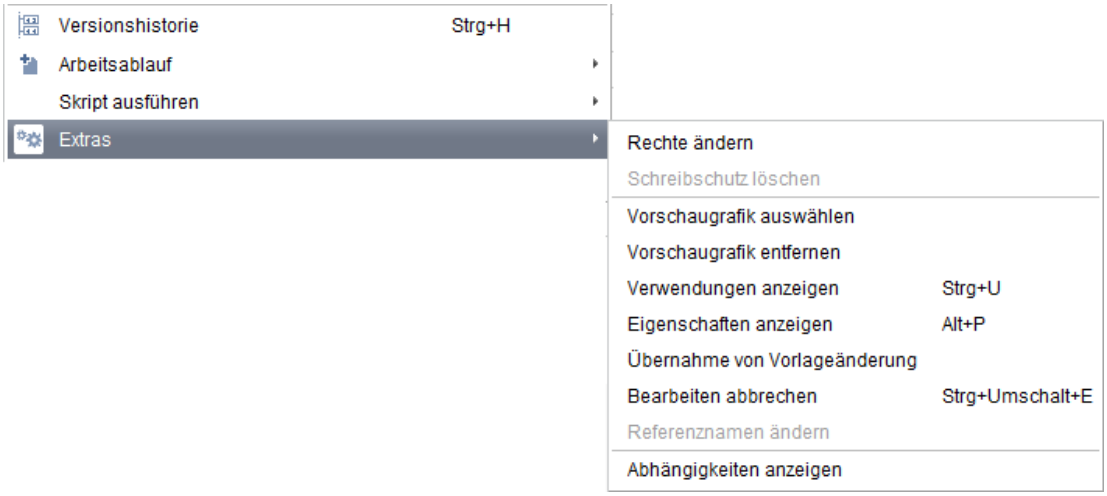


Abbildung 2-30: Extern bearbeiten

Änderungen am Quelltext werden, nach Markierung der Vorlagen, über die Schaltflächen "Lokale Kopie speichern und schließen" oder "Lokale Kopie speichern" gesichert. Im ersten Fall wird der Editor anschließend beendet. Ebenso können noch nicht gespeicherte Änderungen über "Lokale Kopie verwerfen" rückgängig gemacht werden.

Autosave: Ist dieser Haken gesetzt, dann werden alle Änderungen, die im externen Editor gespeichert werden, automatisch auch im FirstSpirit-Client gespeichert.

2.4 Administrative Kontextmenüs der Vorlagen-Verwaltung



2.4.1 Versionshistorie

Über diese Funktion kann die Versionshistorie zu jedem Objekt der Vorlagen-Verwaltung aufgerufen werden.

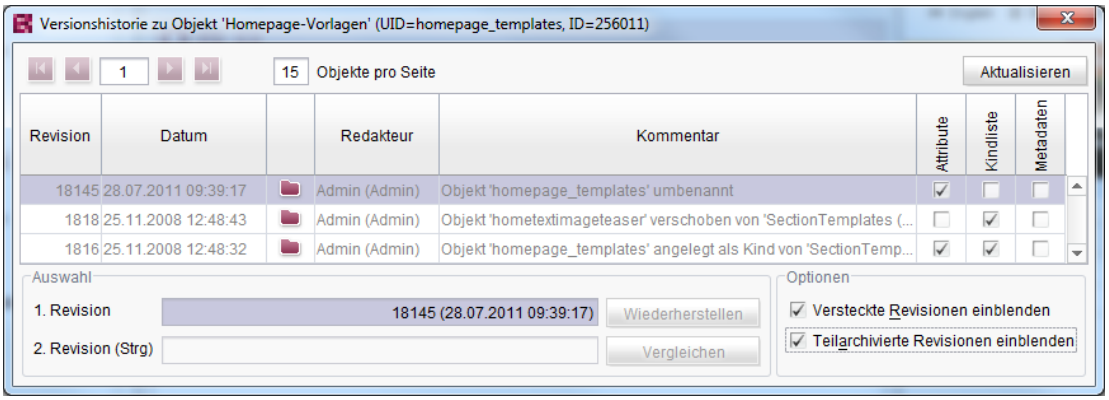


Abbildung 2-31: Versionshistorie auf Absatzvorlagen-Ordner

Allgemeine Informationen zur FirstSpirit Versionshistorie sowie zu den Funktionen des Dialogs in Abbildung 2-31 befinden sich im *FirstSpirit Handbuch für Redakteure*, Kapitel 11.10.

Zusätzlich zu den allgemein verfügbaren Informationen einer Revision (Revision, Datum, Redakteur, Kommentar) wird im rechten Teil der Listendarstellung angezeigt, an welchem Element des Objekts eine Änderung, die zur Vergabe einer neuen

Revisionsnummer geführt hat, vorgenommen wurde (z. B. Attribute, Kindliste, Vorschau, Ausgabekanäle). Dies ist abhängig davon, auf welchem Objekt die Versionshistorie aufgerufen wurde.

2.4.2 Arbeitsablauf starten

Ist auf dem ausgewählten Objekt noch kein Arbeitsablauf aktiv, dann werden unter diesem Menüpunkt alle Arbeitsabläufe aufgelistet, die im Rechte-System für diesen Knoten in der Baumstruktur definiert wurden. Der benötigte Arbeitsablauf kann unter diesem Menüpunkt gestartet werden.

Ist bereits ein Arbeitsablauf für das ausgewählte Objekt aktiv, dann kann er unter diesem Menüpunkt weitergeschaltet werden.

Eine ausführliche Dokumentation von Arbeitsabläufen befindet sich in Kapitel 4 ab Seite 123 sowie im *FirstSpirit Handbuch für Redakteure*, Kapitel 12.

2.4.3 Skript ausführen

Unter diesem Menüpunkt werden alle Skripte aufgelistet, die an dieser Position im JavaClient aufgerufen werden können. Mit Skripten ist es möglich, vorprogrammierte Aktionen oder Berechnungen ausführen zu lassen. Informationen zur Skriptentwicklung in FirstSpirit befinden sich in der *FirstSpirit Online-Dokumentation*.

2.4.4 Suche in Vorlagen

Diese Funktion ist identisch mit der Funktion "Suche in Vorlagen" im Menü "Suche" der FirstSpirit Menüleiste. Weitere Informationen zu dieser Suche befinden sich *FirstSpirit Handbuch für Redakteure*, Kapitel 3.

2.4.5 Extras – Rechte ändern

Mithilfe dieser Funktion können die Rechte für den aktuellen Knoten in der Baumstruktur definiert werden. Sie kann über das Kontextmenü auf allen Knoten aufgerufen werden.

Die Einträge der Listen in den Bereichen "Geerbte Rechte" und "In diesem Objekt definierte Rechte" werden automatisch alphabetisch sortiert angezeigt, zunächst Gruppen, dann Benutzer.

Eine ausführliche Dokumentation zur Definition von Rechten befindet sich im *FirstSpirit Handbuch für Redakteure*, Kapitel 13.

2.4.6 Extras – Schreibschutz löschen

Falls durch einen aktiven Arbeitsablauf ein Schreibschutz für den ausgewählten Knoten besteht, kann der Schreibschutz mithilfe dieser Funktion aufgehoben werden. (Der Schreibschutz wird durch kursive Schrift im Baum dargestellt.) Detaillierte Informationen zum Schreibschutz innerhalb von Arbeitsabläufen befindet sich in Kapitel 4.7 ab Seite 176.

2.4.7 Extras – Vorschaugrafik auswählen / entfernen

Diese Funktion kann über das Kontextmenü auf Seiten-/Absatzebene aufgerufen werden. Das jeweilige Objekt muss sich dazu im Bearbeiten-Modus befinden.

Mithilfe dieser Funktion kann eine Vorschaugrafik für das Register Vorschau des jeweiligen Objekts ausgewählt bzw. eine vorhandene entfernt werden. Hierzu öffnet sich ein Dateifenster, in dem in den lokalen Verzeichnissen des Arbeitsplatzrechners nach der gewünschten Vorschaugrafik gesucht werden kann. Die Grafikdatei muss die Endung "gif", "jpg" oder "png" besitzen.

2.4.8 Extras – Eigenschaften anzeigen

Mithilfe dieser Funktion können die Eigenschaften eines Objektes mit folgenden Informationen angezeigt werden. Die Informationen können dabei je nach Objekt-Typ variieren.

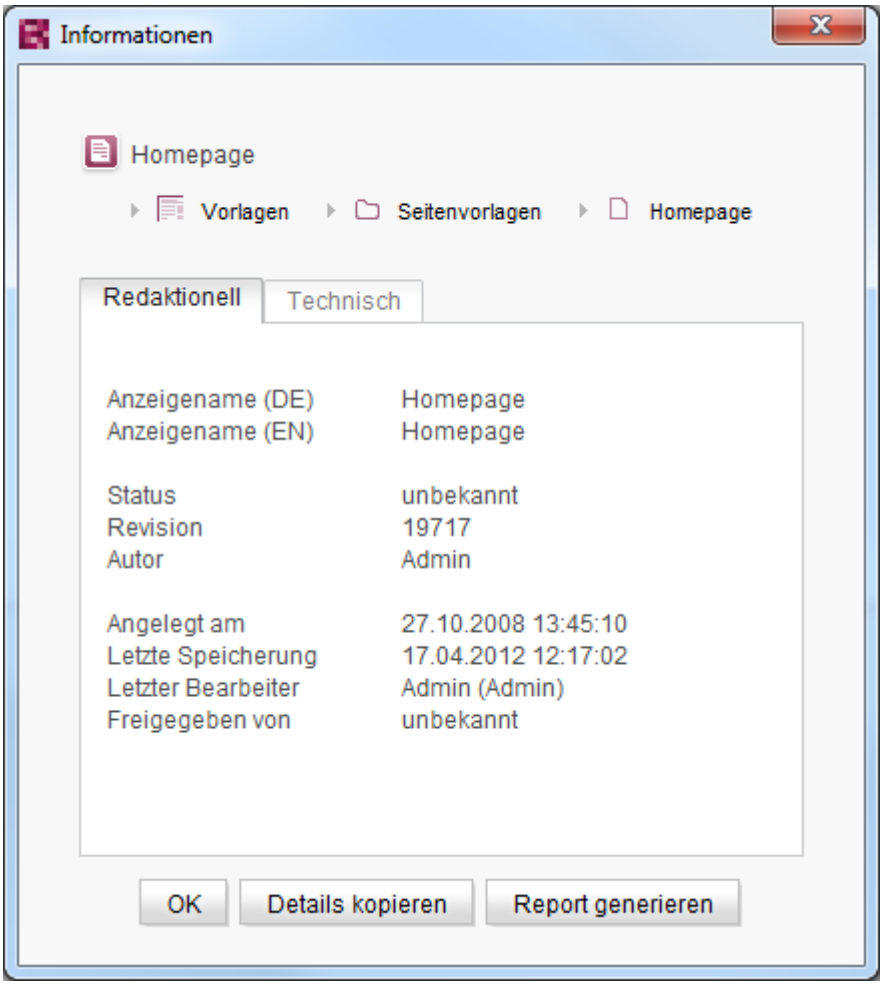


Abbildung 2-32: Eigenschaften einer Seitenvorlage – Redaktionell

Über den Pfad können die Eigenschaften zu weiteren Objekten angezeigt werden.

Register Redaktionell

In diesem Register werden die redaktionell relevanten Eigenschaften eines Objekts angezeigt:

Anzeigename: Anzeigenamen des Objektes (sprachabhängig)

Status: zeigt den Status an (z. B. "Nicht freigegeben", "Freigegeben", "Geändert (nicht freigegeben)")

Revision: zeigt die Revision an

Autor: Name des Benutzers, der das Objekt angelegt hat

Angelegt am: Zeitpunkt, zu dem das Objekt im JavaClient angelegt wurde, mit Datum und Uhrzeit

Letzte Speicherung: Zeitpunkt, zu dem das Objekt zuletzt gespeichert wurde, mit Datum und Uhrzeit

Letzter Bearbeiter: Name des Benutzers, der das Objekt zuletzt bearbeitet hat

Freigegeben von: Name des Benutzers, der das Objekt freigegeben hat

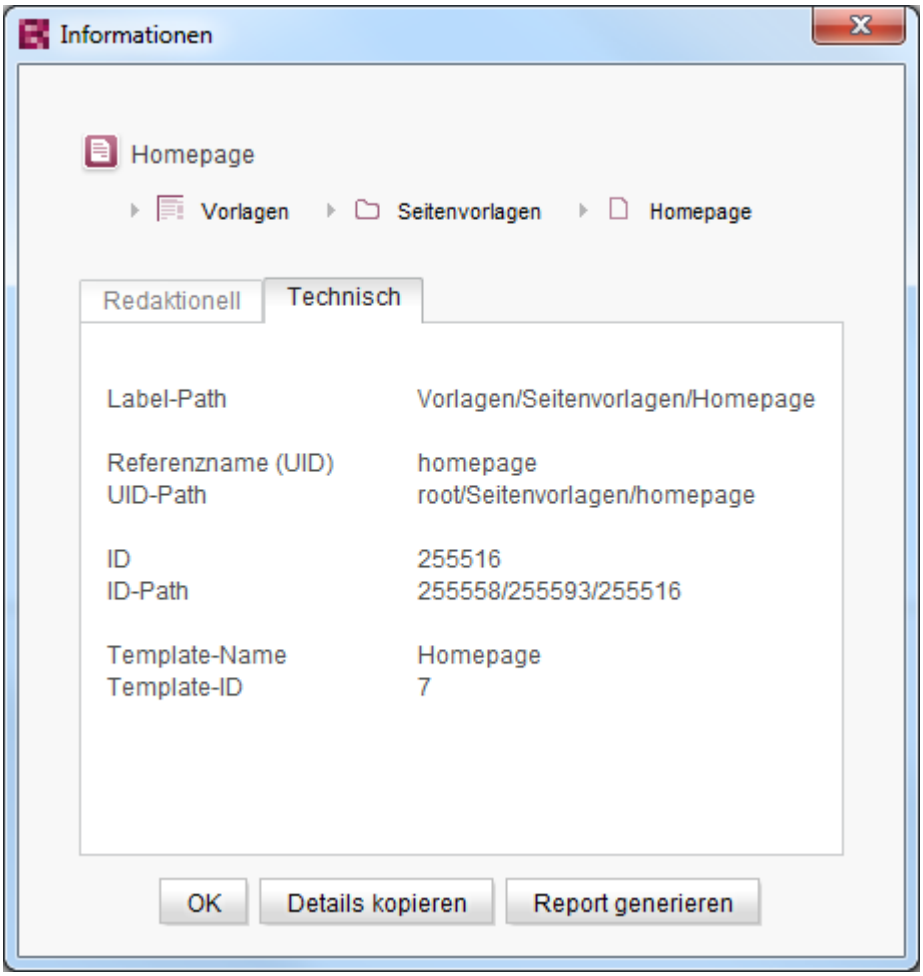


Abbildung 2-33: Eigenschaften einer Seitenvorlage – Technisch

Register Technisch

In diesem Register werden die technisch relevanten Eigenschaften eines Objekts angezeigt:

Label-Path: Pfad zum aktuellen Objekt (Anzeigenamen)

Referenzname (UID): Referenzname (UID) des Objekts

UID-Path: Pfad zum aktuellen Objekt (Referenznamen)

ID: ID des Objekts

ID-Path: Pfad zum aktuellen Objekt (IDs)

Die Pfad-Informationen können auch über das Tastenkürzel Strg + Umschalt + Q angefordert werden.

Template-Name: Anzeigename der zugrundeliegenden Vorlage

Template-ID: ID der zugrundeliegenden Vorlage

Über den Button "OK" schließt sich der Dialog wieder. Über den Button "Details kopieren" können alle Informationen des Dialogs in die Zwischenablage übertragen werden. Über den Button "Report generieren" können die Informationen als HTML-Seite ausgegeben werden. Dazu kann ein zusätzlicher Kommentar, beispielsweise eine Fehlerbeschreibung, eingegeben werden.

2.4.9 Extras – Verwendungen anzeigen

Diese Funktionalität kann über das Kontextmenü auf Seiten-, Absatz-, Formatvorlagen, Skripten und Tabellenvorlagen aufgerufen werden.

Mit ihrer Hilfe kann automatisch zu Knoten in anderen Verwaltungen gesprungen werden, die auf dem Objekt beruhen, auf dem diese Funktionalität ausgeführt wurde. Wird das Objekt mehrfach verwendet, öffnet sich ein neues Fenster, das alle Knoten (z. B. Absätze aus der Inhalte-Verwaltung) zeigt, die auf dem aktuellen Objekt beruhen. Ein Doppelklick auf einen dieser Einträge zeigt den entsprechenden Knoten im Verzeichnisbaum.

2.4.10 Extras – Übernahme von Vorlagenänderungen

Mithilfe dieser Funktion können Änderungen an der Definition der Inhaltsbereiche in einer Seitenvorlage für bestehende Seiten übernommen werden.

Die Funktionalität steht nur auf Seitenvorlagen in der Vorlagen-Verwaltung zur Verfügung.

Wird beispielsweise innerhalb einer Seitenvorlage ein Inhaltsbereich hinzugefügt, so wirkt sich diese Änderung nicht automatisch auf bestehende Seite aus. Um bei einer Änderung an einer Seitenvorlage die bestehenden Seiten zu aktualisieren, kann die Funktion "Übernahme von Vorlagenänderungen" verwendet werden. Die Funktion prüft die Definition der Inhaltsbereiche in der Seitenvorlage, gegen die Inhaltsbereiche der Seiten, die diese Vorlage verwenden:

- Werden Inhaltsbereiche gefunden, die in der Vorlage definiert, aber auf der bestehenden Seite nicht vorhanden sind, werden diese Inhaltsbereiche in der Seite hinzugefügt.
- Werden umgekehrt Inhaltsbereiche gefunden, die in der Vorlage fehlen, aber in der Seite vorhanden sind, so
  - werden diese aus der Seite entfernt, wenn sie keine Absätze enthalten.
  - bleiben diese auf der Seite erhalten, wenn sie Absätze enthalten.

2.4.11 Extras – Bearbeiten abbrechen

Mithilfe dieser Funktion kann der Bearbeitungsmodus auf Knoten beendet werden, ohne vorgenommene Änderungen zu speichern. Änderungen, die bereits über STRG + S oder die Speichern-Funktion der Icon-Leiste gespeichert wurden, können allerdings nicht rückgängig gemacht werden.

2.4.12 Extras – Referenznamen ändern

Über diese Funktion kann der Referenzname von Objekten im Nachhinein geändert werden.

Da möglicherweise noch Referenzen für das Objekt vorhanden sind, die nach einer Änderung des Referenznamens ungültig sind, wird zunächst ein Warnhinweis angezeigt. Wird "Trotzdem ändern" gewählt, kann im nächsten Dialog der Referenzname des Objekts geändert werden.

2.4.13 Extras – Abhängigkeiten anzeigen

Wesentliche Funktionalitäten von FirstSpirit basieren auf dem sogenannten Referenzgraph eines Projekts. Der Referenzgraph eines Projekts wird verwendet, um Abhängigkeiten innerhalb des Projekts zu erkennen und ist damit essentieller Bestandteil komplexer Funktionalitäten, beispielsweise der Serverseitigen Freigabe (siehe Kapitel 6 Seite 222).

Die Visualisierung des Referenzgraphs für ein Objekt kann von Projektadministratoren über das Kontextmenü "Extras – Abhängigkeiten anzeigen" angefordert werden. Damit ist es auch in komplexen Projekten möglich, die Abhängigkeiten eines Objekts zu identifizieren.

Referenzgraphen einzelner Datensätze der Datenquellen-Verwaltung werden über das Kontextmenü des jeweiligen Datensatzes aufgerufen.

Die Register zeigen die Abhängigkeiten des Objekts, in Form von eingehenden und ausgehenden Kanten, sowohl für den aktuellen Stand als auch den zuletzt freigegebene Stand an (siehe Abbildung 2-34).

Die Darstellung kann auf eine hierarchische Ansicht umgestellt werden, die besonders für komplexe Abhängigkeiten empfehlenswert ist (siehe Abbildung 2-34). Eine direkte Aktualisierung bei Änderungen und das Zoomen innerhalb der Ansicht ist über die Buttons im oberen Fensterbereich möglich. Für eine spätere Verwendung kann die Ansicht außerdem als Bild gespeichert werden.

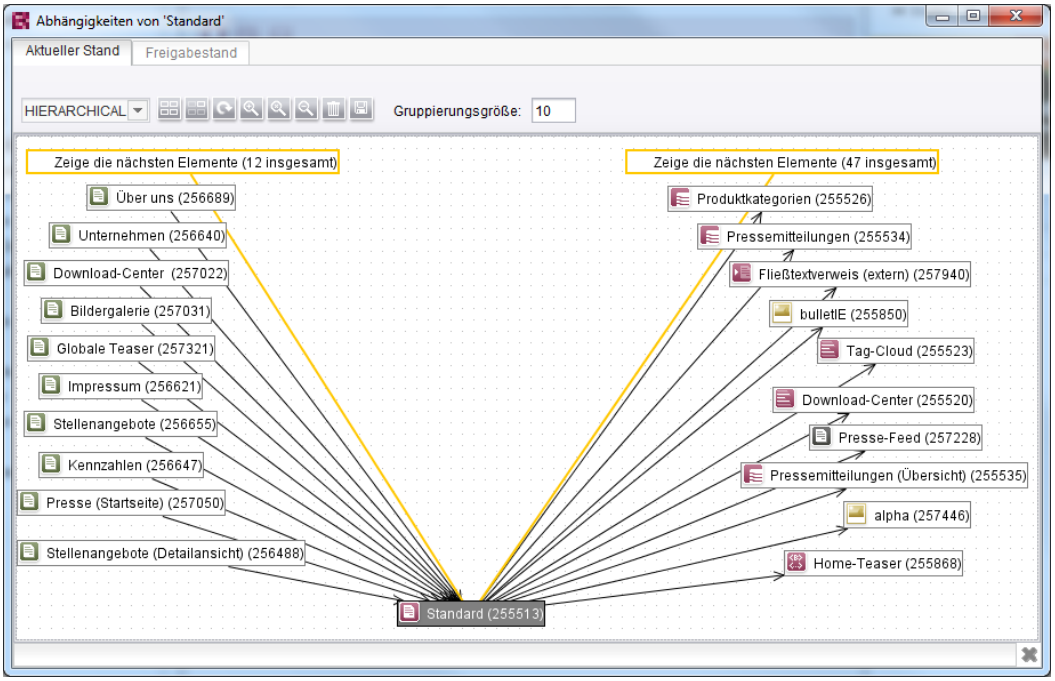


Abbildung 2-34: Anzeigen von Abhängigkeiten über den Referenzgraph

2.4.14 Extras – Kopie dieses Arbeitsablaufs erstellen

Diese Funktion kann auf Arbeitsabläufen aufgerufen werden. Sie erstellt eine Kopie des ausgewählten Arbeitsablaufs unterhalb des Knotens "Arbeitsabläufe".

2.5 Seitenvorlagen

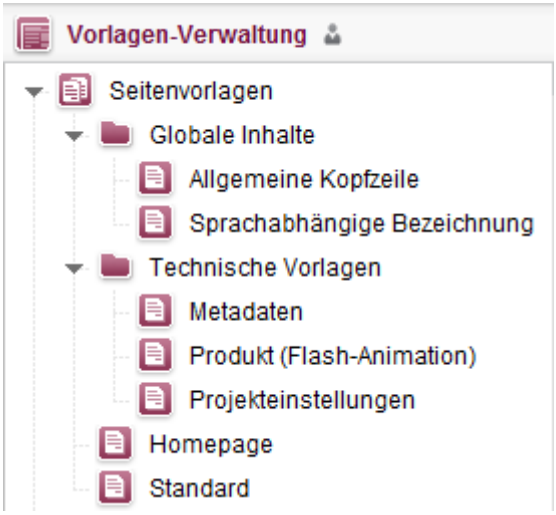


Abbildung 2-35: Baumansicht Vorlagen-Verwaltung – Seitenvorlagen

Seitenvorlagen erstellen das Grundgerüst einer Seite. In den Seitenvorlagen werden z. B. die Platzierungen von Logos und Navigationen sowie ähnliche allgemeine Einstellungen festgelegt. Außerdem definieren die Seitenvorlagen, an welchen Stellen ein Redakteur Inhalte einfügen kann.

Baumelemente innerhalb der Seitenvorlagen:

Wurzelelement der Seitenvorlagen

Ordner innerhalb des Knotens Seitenvorlagen

Seitenvorlage

Mit FirstSpiritVersion 5.0 werden als Lesehilfe an vielen Stellen der Vorlagen-Verwaltung Zeilennummern angezeigt. Mithilfe des Tastaturkürzels STRG + L können sie im JavaClient ein- und ausgeblendet werden.

2.5.1 Register Vorschau

Um eine Vorstellung zu erhalten, wie eine Vorlage später im Browser dargestellt wird, lässt sich auf dem Register "Vorschau" eine zuvor angefertigte Vorschaugrafik (z. B. ein Screenshot) anzeigen. Somit kann jeder Anwender sofort erkennen, welche Vorlage er gerade markiert hat.

Das Einfügen einer Grafik auf diesem Register kann auf drei verschiedene Arten geschehen:

1. Die Vorlage sperren, dann aus dem Kontextmenü Vorlage "Vorschaugrafik auswählen" anklicken und eine Datei vom Typ "gif", "jpg" oder "png" auswählen.

2. Die Vorlage sperren, dann aus dem Datei-Explorer eine Datei vom Typ "gif", "jpg" oder "png" auswählen, mit der Maus an die Vorschaustelle ziehen und dort ablegen.

3. Die Vorlage sperren, von einer Webseite (nur MS-IE) eine linkfreie Grafik (Ctrl-Taste gedrückt) auswählen, mit der Maus auf das Register Vorschau ziehen und dort ablegen.

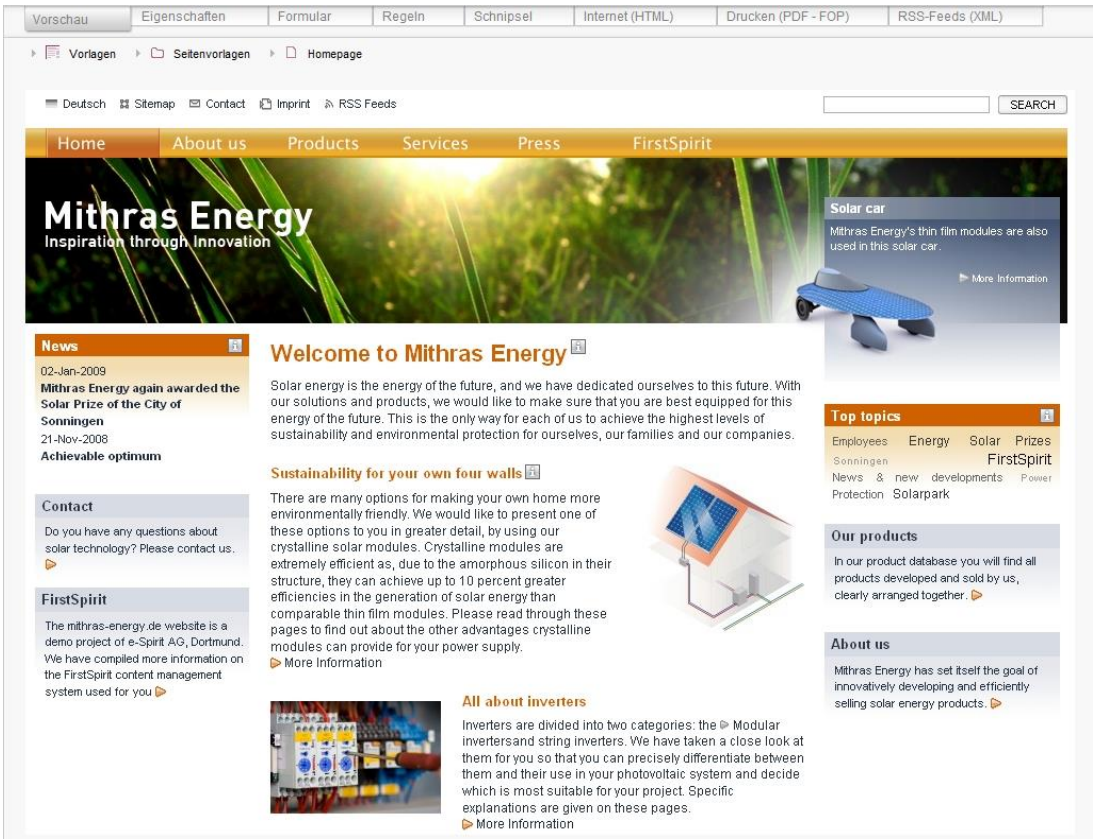


Abbildung 2-36: Seitenvorlage – Register "Vorschau"

2.5.2 Register Eigenschaften

Das Register "Eigenschaften" enthält unterschiedliche Einträge für Seiten- und Absatzvorlagen. Folgende Abbildung zeigt die Eigenschaften einer **Seitenvorlage**:

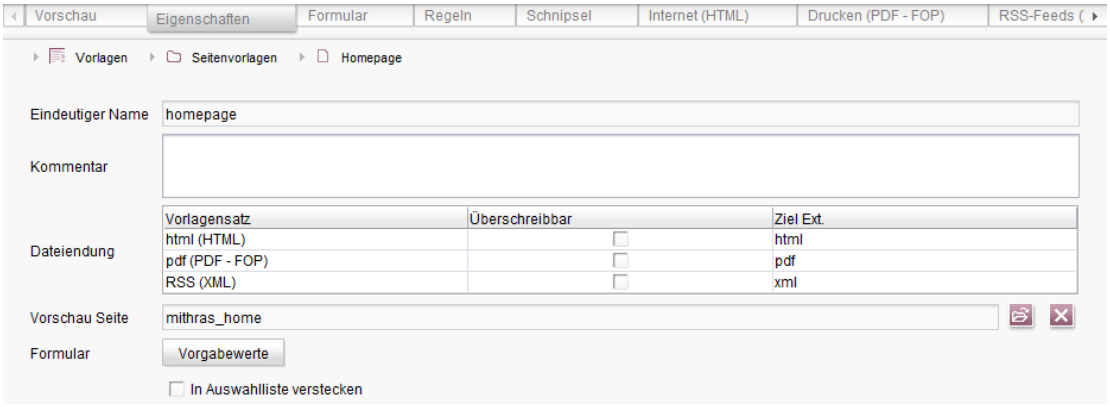


Abbildung 2-37: Seitenvorlage – Register "Eigenschaften"

Eindeutiger Name: In diesem Feld wird eine eindeutige Bezeichnung angegeben, unter der die Vorlage im Dateiverzeichnis gespeichert wird (siehe "Referenzname" in Kapitel 2.2.1 Seite 21).

Kommentar: Hier kann ein Kommentar eingegeben werden, der die Seiten- oder Absatzvorlage näher beschreibt.

Datei-Endung – Vorlagensatz: Name und Art der Vorlagensätze, die der Projektadministrator in der Server- und Projektkonfiguration für das aktuelle Projekt definiert hat. Deaktivierte Vorlagensätze werden ausgegraut dargestellt und lassen sich nicht bearbeiten.

Datei-Endung – Überschreibbar: Wird der Haken gesetzt, bedeutet das, dass die Endungen einer Seitenvorlage, die in einem der nächsten beiden Eingabefelder angegeben werden, von einer Absatzvorlage überschrieben werden können.

Datei-Endung – Ziel Ext.: Die Endung der Vorlage, auf die verlinkt werden soll. Durch Doppelklicken in das Feld kann die Endung bearbeitet werden.

Vorschau Seite: Hier kann eine Seite aus der Struktur-Verwaltung ausgewählt werden, in der die Vorlage verwendet wird. Vorgenommene Änderungen an der Vorlage können so direkt in der Vorlagen-Verwaltung mit der Vorschau-Funktion überprüft werden.

Vorlage in Auswahlliste verstecken: Durch Aktivieren dieser Option wird verhindert, dass ein Redakteur beim Anlegen einer neuen Seite diese Vorlage

verwenden kann.

Formular: An dieser Stelle steht der Button „Vorgabewerte“ zur Verfügung, über den Vorgabewerte für die Vorlage definiert werden können. Es öffnet sich der Pflegedialog für die Festlegung der Vorbelegungen. Die sprachabhängigen Vorgabewerte werden direkt nach dem Speichern der Eigenschaften im Vorschaubereich angezeigt.

Inhaltsbereiche / Absatzeinschränkungen: Mit einem Klick auf die Icons oben rechts kann hier ein neuer Inhaltsbereich zur Seitenvorlage hinzugefügt, ein bestehender Inhaltsbereich gelöscht oder die Liste der Inhaltsbereiche umsortiert werden.

Genauso wie alle anderen Vorlagen lassen sich auch Inhaltsbereiche sprachabhängig angelegen und können somit mit einem (oder mehreren) sprachabhängigen sowie einem eindeutigen Referenznamen versehen werden.

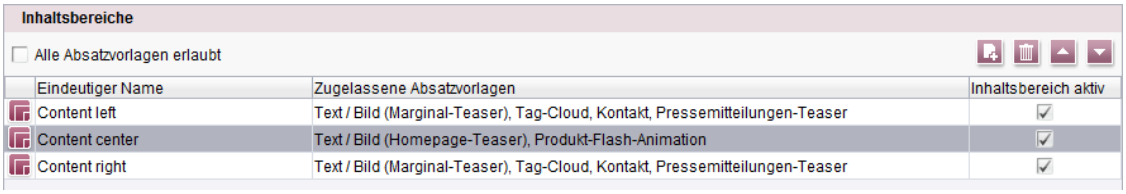


Abbildung 2-38: Inhaltsbereiche für eine Seitenvorlage definieren

Mit einem Klick auf einen Inhaltsbereich können **Absatzeinschränkungen** für die Seitenvorlage definiert werden:

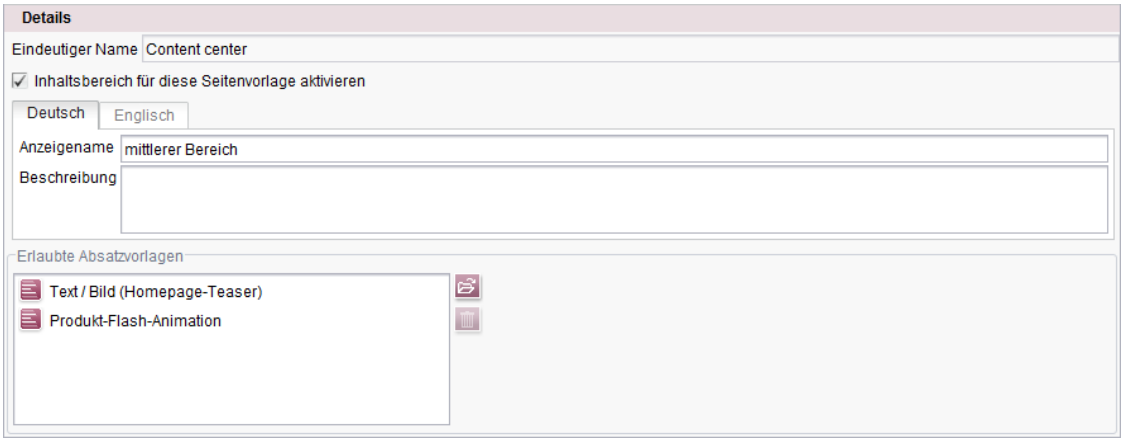


Abbildung 2-39: Absatzeinschränkungen definieren

Dazu können die gewünschten Absatzvorlagen durch das Hinzufügen zu bzw. das Entfernen aus einer Liste (für einen Inhaltsbereich) entweder erlaubt oder verboten werden. Für den entsprechenden Inhaltsbereiche bedeutet dies, dass nur noch die

jeweils ausgewählten Absatzvorlagen zugelassen werden. Das Hinzufügen erfolgt über die Selektion des Inhaltsbereichs und einen Klick auf das Symbol des geöffneten Ordners. Das Löschen erfolgt, indem die Einschränkung unterhalb der Seitenvorlage markiert und anschließend per Tastatur über die Entfernen-Taste oder per Maus mit einem Klick auf das Mülleimer-Symbol entfernt wird.

Optional können auch alle Absatzvorlagen für alle Inhaltsbereiche einer Seitenvorlage erlaubt werden. Damit wird jede Absatzeinschränkung für die Seitenvorlage aufgehoben. Dies ist über die Aktivierung der Checkbox „Alle Absatzvorlagen erlauben“ möglich.

Werden alle Absatzeinschränkungen unterhalb einer Seitenvorlage gelöscht, bestehen damit keine Einschränkungen mehr für das Anlegen von Absätzen innerhalb einer Seite. Dabei kann unter Umständen das Layout der Seite beschädigt werden, indem beispielsweise Absätze angelegt werden, die für das Layout der Seite nicht geeignet sind.

2.5.3 Register Formular

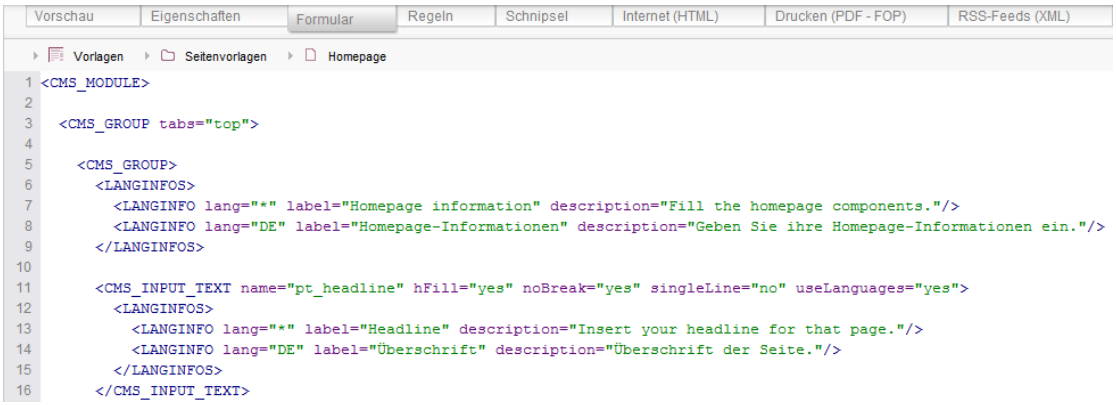


Abbildung 2-40: Seitenvorlage – Register "Formular"

Das Register "Formular" zeigt die Datei GUI.XML. Ist die Vorlage gesperrt, können auch hier direkt Änderungen vorgenommen werden.

Beim Abspeichern der GUI.XML wird eine DTD-Validierung durchgeführt. Verletzungen der Wohlgeformtheit sind dabei fatal, die GUI.XML lässt sich in diesem Fall nicht speichern. Sonstige Fehler werden nur angezeigt.

Wird im gesperrten Zustand eine Vorschau für die GUI.XML angefordert, dann werden die Änderungen vorher automatisch gespeichert. (Es wird keine neue Version angelegt – das erfolgt nur beim Entsperren!)

Zur Aktualisierung von Inhaltsbereichen in bestehenden Seiten (bei Änderung der Definition innerhalb der Seitenvorlage) siehe Kapitel 2.4.10 Seite 50.

*In der "FirstSpirit Online Dokumentation" ist eine Auflistung aller zur Verfügung stehenden Eingabeelemente zu finden. Unter dem Menüpunkt **Vorlagenentwicklung – Formulare** werden die Eingabeelemente mit allen Attributen und einem schematischen Beispiel erläutert.*

Das Hinzufügen von Eingabekomponenten im Register „Formular“ ist durch die Code-Vervollständigung vereinfacht worden (siehe Kapitel 7 Seite 239).

2.5.4 Register Vorlagensätze

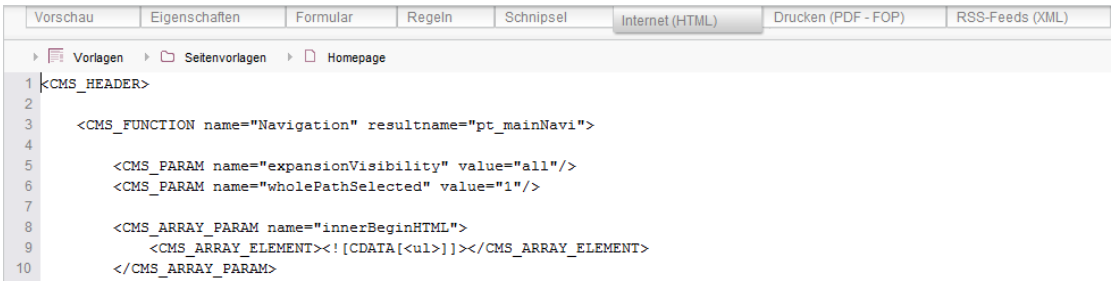
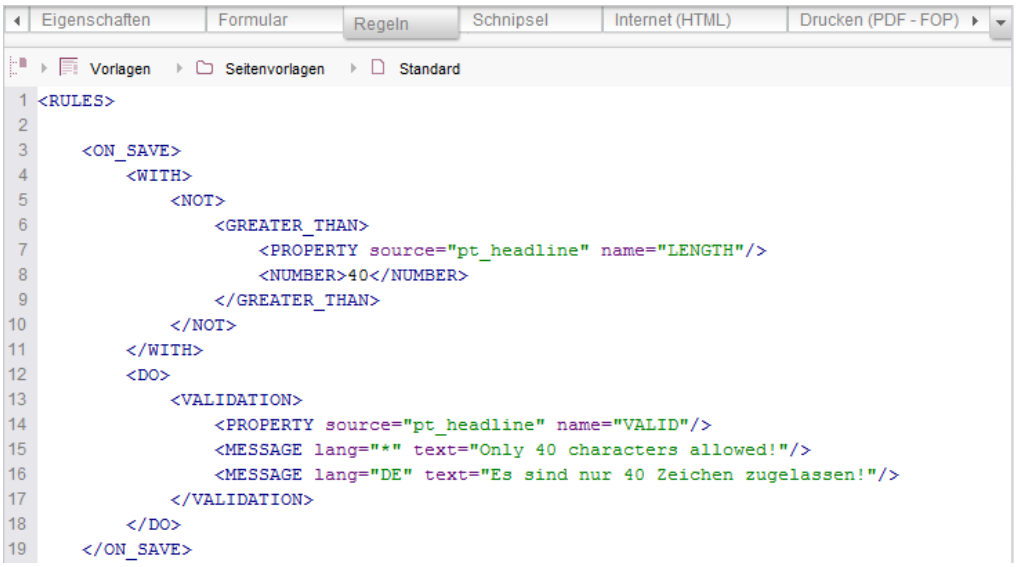


Abbildung 2-41: Seitenvorlage – Register Vorlagensätze

Die Register "Internet", "Drucken" und "RSS-Feeds" sind Vorlagensätze, die der Projektadministrator in der Server- und Projektkonfiguration für dieses Projekt angelegt hat. Die Register zeigen den Quelltext der verschiedenen Vorlagensätze für die aktuelle Vorlage an. Ist die Vorlage gesperrt, können hier direkt Änderungen vorgenommen werden.

Wurde eine Änderung im Quelltext vorgenommen, wird die Veränderung beim Speichern auf die Wohlgeformtheit des CMS_HEADER geprüft. Sollte ein Fehler auftreten, wird dieser sofort über ein neues Fenster angezeigt.

2.5.5 Register Regeln



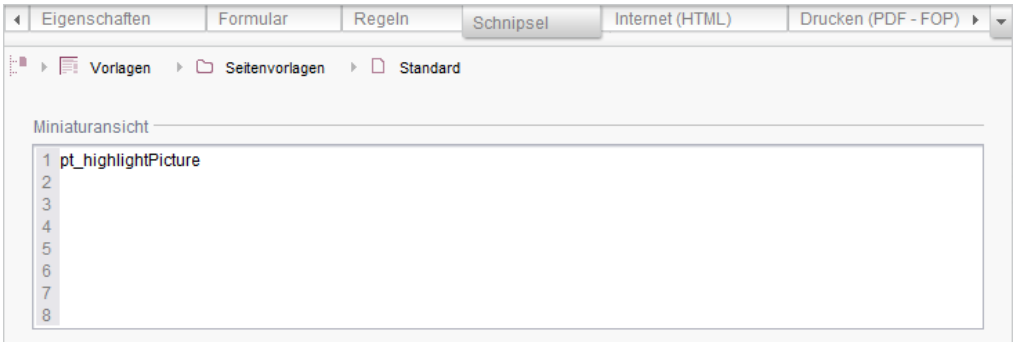

```
<RULES>

    <ON_SAVE>
        <WITH>
            <NOT>
                <GREATER_THAN>
                    <PROPERTY source="pt_headline" name="LENGTH"/>
                    <NUMBER>40</NUMBER>
                </GREATER_THAN>
            </NOT>
        </WITH>
        <DO>
            <VALIDATION>
                <PROPERTY source="pt_headline" name="VALID"/>
                <MESSAGE lang="*" text="Only 40 characters allowed!"/>
                <MESSAGE lang="DE" text="Es sind nur 40 Zeichen zugelassen!"/>
            </VALIDATION>
        </DO>
    </ON_SAVE>
```

Ein Vorlagenentwickler kann über die Definition von Regeln innerhalb der (Formular)-Vorlage bestimmte Elemente oder Eigenschaften des Formulars beeinflussen und so ein "Dynamisches Formular" erstellen. Zur Regeldefinition dient der neue Registerbereich „Regeln“. Hier kann beispielsweise eine Eingabekomponente, die im Registerbereich „Formular“ der Vorlage definiert wurde, mit einer Regel verknüpft werden.

Eine genaue Beschreibung der Regeln befindet sich in der FirstSpirit Online-Dokumentation.

Übergeordnete Objekte, also Objekte die außerhalb des eigentlichen Formulars liegen, können nicht einbezogen werden. Über eine Regel kann also beispielsweise nicht beeinflusst werden, welche Vorlagen in welchen Bereichen der Struktur-Verwaltung verwendet werden dürfen.

2.5.6 Register Schnipsel



Über das Register "Schnipsel" kann für einige Vorlagentypen bestimmt werden, wie Suchergebnisse dargestellt werden sollen, die auf diesen Vorlagen basieren. Dazu wird auf die Variablennamen der Eingabekomponenten der Vorlage zurückgegriffen. Diese Darstellung wird sowohl im JavaClient als auch im WebClient verwendet.

Ziel ist es, Suchtreffer nicht nur mit dem Objektnamen darzustellen, sondern mit

- einem Bild,
- einem Titel und
- einem Textausschnitt,

die das jeweilige Objekt möglichst treffend widerspiegeln. Auf diese Weise soll der Redakteur eine klare Vorstellung vom Inhalt des Suchtreffers erhalten, um so den relevantesten Treffer besser ausmachen zu können und schneller zum gesuchten Objekt zu gelangen.

Um die Ausgabe stärker an die Bedürfnisse des jeweiligen Projekts und der Redakteure anpassen zu können, können auch mehrere Eingabekomponenten kombiniert werden. Zusätzlich können Methoden, die mit $CMS_VALUE(...)$ verwendet werden können, eingesetzt werden. So kann die Verwendung von eingepflegten redaktionellen Inhalte für die Suche abhängig von den Eingaben gemacht werden. Ist eine Eingabekomponente, die auf dem Register „Schnipsel“ angegeben ist, nicht durch den Redakteur gefüllt, wird standardmäßig als Titel der Name und als Textausschnitt der Pfad zum Suchtreffer dargestellt. Im WebClient wird unabhängig in jedem Fall der Pfad angezeigt.

Eine genaue Beschreibung der Schnipsel befindet sich in der FirstSpirit Online-Dokumentation.

2.6 Absatzvorlagen

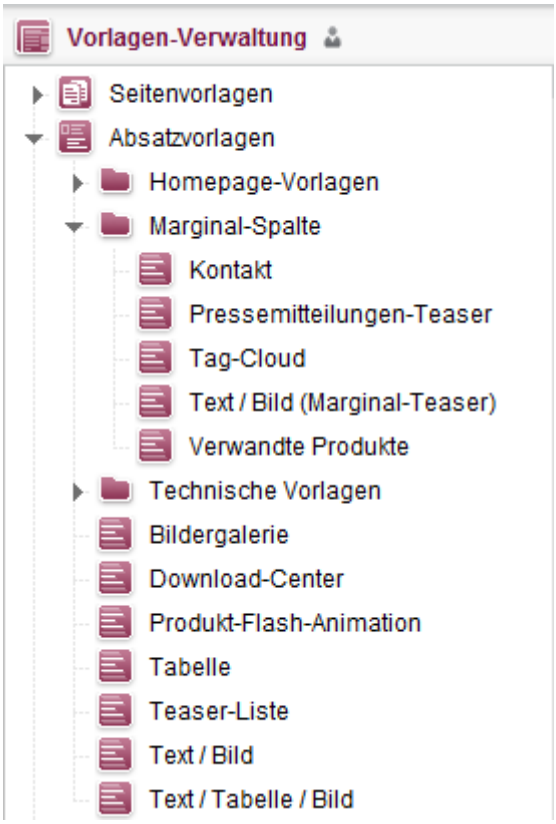


Abbildung 2-42: Baumansicht Vorlagen-Verwaltung – Absatzvorlagen

Absatzvorlagen werden verwendet, um Inhalte in das Grundgerüst einer Seite einzufügen, das von den Seitenvorlagen definiert wird. Innerhalb einer Absatzvorlage werden alle Eingabeelemente definiert, die dynamische Inhalte der Seite (Text, Tabellen, Bilder, Datensätze,...) aufnehmen sollen. In jeden Absatzbereich einer Seite können beliebig viele Absätze eingehängt werden. Für die verschiedenen möglichen Inhalte einer Seite stehen meist auch mehrere unterschiedliche Absatzvorlagen zur Verfügung.

Baumelemente innerhalb der Absatzvorlagen:

Wurzelelement der Absatzvorlagen

Ordner innerhalb des Knotens Absatzvorlagen

Absatzvorlagen

2.6.1 Register Vorschau, Eigenschaften, Formular, Vorlagensätze, Regeln & Schnipsel

Die Register Vorschau, Eigenschaften, Formular, Regeln, Schnipsel und Vorlagensätze der Absatzvorlagen sind in ihrer Funktion identisch zu den gleichnamigen Registern der Seitenvorlagen und können ebenso bearbeitet werden.

Informationen zu den einzelnen Registern können Sie den Kapiteln 2.5.1 (Seite 54) bis Kapitel 2.5.6 (Seite 60) entnehmen.

Es besteht die Möglichkeit, innerhalb einer Seite HTML-Sprungmarken zu verwenden, wie sie beispielsweise bei umfangreichen Seiteninhalten zum Einsatz kommen. Hierfür muss im Eigenschaften-Register der Absatzvorlage das <a>-Tag aktiviert werden. Die Sprungmarke wird daraufhin automatisch aus dem Referenznamen der Absatzvorlage erzeugt und kann anschließend innerhalb einer Verweisvorlage verwendet werden.

Eine genaue Beschreibung zu der Verwendung von Sprungmarken befindet sich in der FirstSpirit Online-Dokumentation.

Für Absatzvorlagen steht die Option "Vorlage in Auswahlliste verstecken" (vgl. Kapitel 2.5.2) nicht zur Verfügung.

2.7 Formatvorlagen

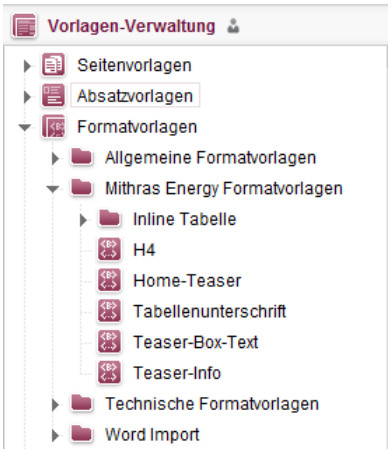


Abbildung 2-43: Baumansicht Vorlagen-Verwaltung – Formatvorlagen

Über Formatvorlagen werden Textformatierungen definiert, die anschließend in den Eingabeelementen DOM-Editor und DOM-Tabelle verwendet werden können. Einige Formatvorlagen werden standardmäßig mit ausgeliefert, z. B. Formatvorlagen für Absätze ("Standard"), Zeilenumbrüche (Kürzel "br"), "Fett", "Kursiv", zur Darstellung von Tabellen (Kürzel "table", "tr" und "td") usw. Sie befinden sich unterhalb des Ordners "Allgemeine Formatvorlagen" im Bereich "Formatvorlagen" in der Vorlagen-Verwaltung.

Diese Standard-Formatvorlagen sind für den korrekten Betrieb in vielerlei Hinsicht erforderlich und dürfen nicht gelöscht werden.

Die Formatvorlagen

- "Gelöscht" (Kürzel "deleted"),
- "Gelöscht (Absatz)" (Kürzel "deleted_block"),
- "Eingefügt" (Kürzel "inserted") und
- "Eingefügt (Absatz)" (Kürzel "inserted_block")

werden für die Darstellung von Versionsvergleichen ("Versionshistorie") verwendet. Durch Ändern der Eigenschaften dieser Formatvorlagen (Farben, Schriftgröße, Rahmen) (siehe Kapitel 2.7.1 Seite 64) kann somit die Darstellung von Änderungen in Versionsvergleichen beeinflusst werden.

Auch die Darstellung der anderen Standard-Formatvorlagen im DOM-Editor kann geändert werden.

Zusätzlich zu den Standard-Formatvorlagen können Vorlagenentwickler weitere projektspezifische Formatvorlagen anlegen.

2.7.1 Register Eigenschaften

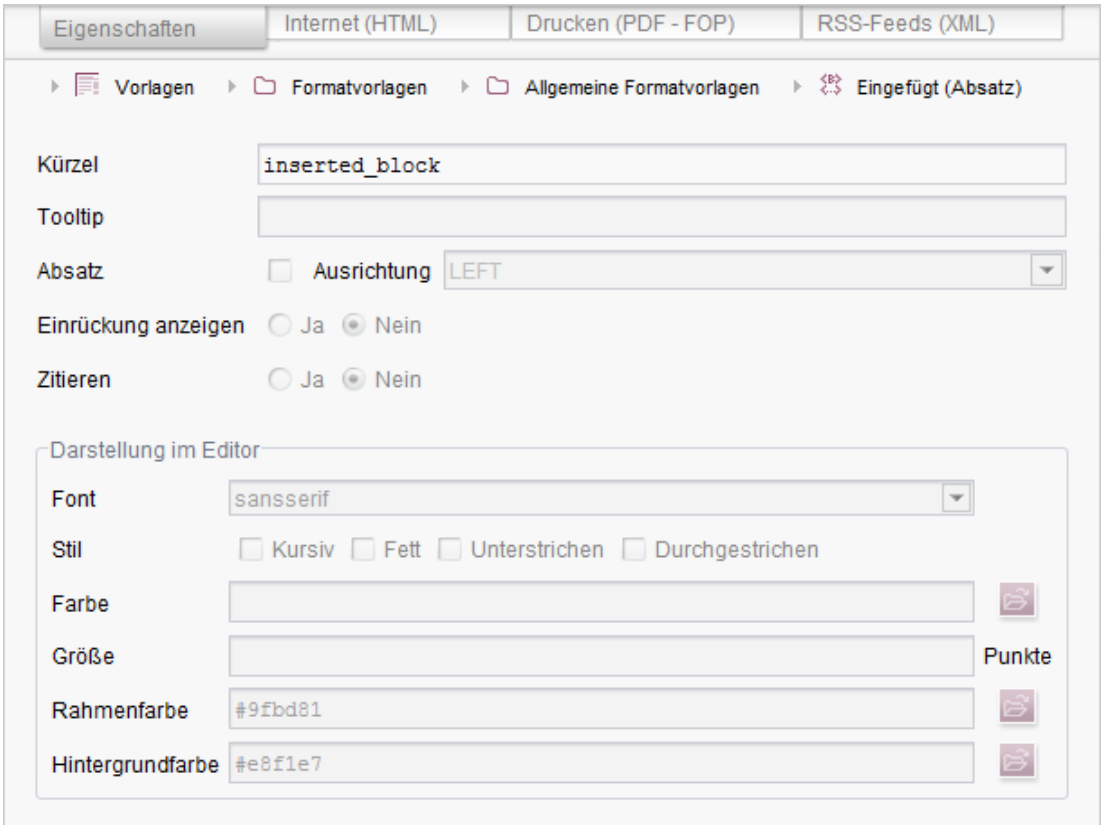


Abbildung 2-44: Formatvorlage – Register Eigenschaften

In der Registerkarte **Eigenschaften** werden die grundlegenden Eigenschaften einer Formatvorlage definiert. Die einzelnen Felder haben hierbei folgende Bedeutungen:

Kürzel: Benötigt wird das Kürzel im Formularbereich der Seiten- oder Absatzvorlage, um die gültigen Formatvorlagen für die Eingabekomponente anzugeben. Aus dem Kürzel wird der entsprechende XML-Tag-Name gebildet, z. B. für die Formatvorlage "Bold" (siehe dazu auch FirstSpirit-Online-Dokumentation – Bereich Vorlagensyntax / Formatvorlagen). Der Name muss eindeutig sein und darf keine Sonderzeichen enthalten. Er wird automatisch nach dem eindeutigen Referenznamen der Formatvorlage vergeben. Um die Eindeutigkeit nicht zu gefährden, sollte das Kürzel nicht manuell verändert werden (siehe Kapitel 2.2.1

Seite 21).

Ein eindeutiger Name bzw. das Kürzel einer Formatvorlage kann nach dem Anlegen nicht mehr geändert werden, da ansonsten alle Bezüge innerhalb des Projekts verloren gingen!

Tooltip: Der in diesem Feld eingetragene Text erscheint als Hilfetext beim Überfahren der Formatierung mit der Maus, beispielsweise im DOM-Editor.

Absatz: Wird die Checkbox "Absatz" aktiviert, wird immer der gesamte Absatz formatiert. Ist die Checkbox nicht aktiviert, wird die Formatierung nur für einzelne, markierte Zeichen angewendet.

Ausrichtung: Wurde die Checkbox "Absatz" aktiviert, kann hier die Ausrichtung des Textes, beispielsweise im DOM-Editor, angegeben werden.

Einrückung anzeigen, definiert, wie der entsprechend formatierte Text angezeigt werden soll. Wird diese Option aktiviert, so werden alle Leerzeichen angezeigt und der Text nicht mehr automatisch umgebrochen. Wird diese Option deaktiviert, werden Leerzeichen in "HTML-Notation" angezeigt.

Zitieren: Durch die Auswahl von **Ja** werden auf die einzelnen Vorlagensätze die vollständigen Konvertierungsregeln angewendet (convert-Teil und quote-Teil). Bei der Auswahl von **Nein** wird auf die einzelnen Vorlagensätze nur der convert-Teil der Konvertierungsregeln angewendet.

Über das Feld **Darstellung im Editor** können weitere, nur im Editor dargestellte Formatierungen definiert werden.

Font: Hier kann eine Schriftart ausgewählt werden, durch die der Text dargestellt werden soll. (Diese Schriftart muss auf jedem Clientrechner installiert sein, ansonsten wird eine ähnliche Schriftart genommen.)

Stil: Hier kann ausgewählt werden, ob der Text im DOM-Editor fett, kursiv oder unterstrichen dargestellt werden soll.

Farbe: Hier kann eine Farbauswahl für den dargestellten Text vorgenommen werden.

Größe: Die Größe, in der ein Text im DOM-Editor dargestellt werden soll, wird hier bestimmt. Hier sind auch relative Angaben (+2, -1, etc.) möglich.

Rahmenfarbe: Hier kann eine Farbauswahl für einen Rahmen innerhalb der Eingabekomponente vorgenommen werden.

Hintergrundfarbe: Hier kann eine Farbauswahl für den Hintergrund innerhalb der Eingabekomponente vorgenommen werden.

2.7.2 Register Vorlagensätze

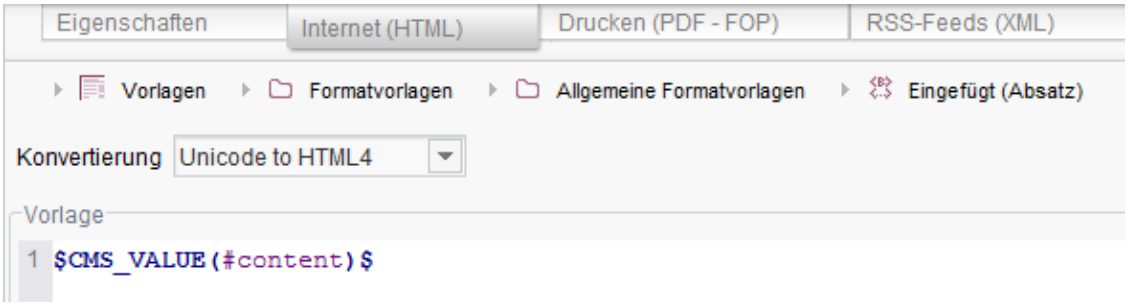


Abbildung 2-45: Formatvorlage – Register Vorlagensätze

Konvertierung: Hier kann eine der Konvertierungsregeln ausgewählt werden, die in der Server- und Projektkonfiguration in den Servereigenschaften konfiguriert wurde.

Vorlage: Hier wird der HTML-Code eingetragen, der die gewünschten Formatierungen für den Text auf der Website erzeugt. Der Ausdruck #content steht für den Text, der im DOM-Editor eingegeben wurde.

Eine genaue Beschreibung der Formatvorlagen und der Konvertierungsregeln befindet sich in der FirstSpirit Online-Dokumentation.

Die selektierte Konvertierungsregel wird nur bei der Ausgabe der Eingabekomponenten CMS_INPUT_DOM bzw. CMS_INPUT_DOMTABLE über das Systemobjekt #content angewendet, z. B. über $CMS_VALUE(#content)$.

Mit FirstSpiritVersion 5.0 werden als Lesehilfe an vielen Stellen der Vorlagen-Verwaltung Zeilennummern angezeigt. Mithilfe des Tastaturkürzels STRG + L können sie im JavaClient ein- und ausgeblendet werden.

2.8 Stilvorlagen

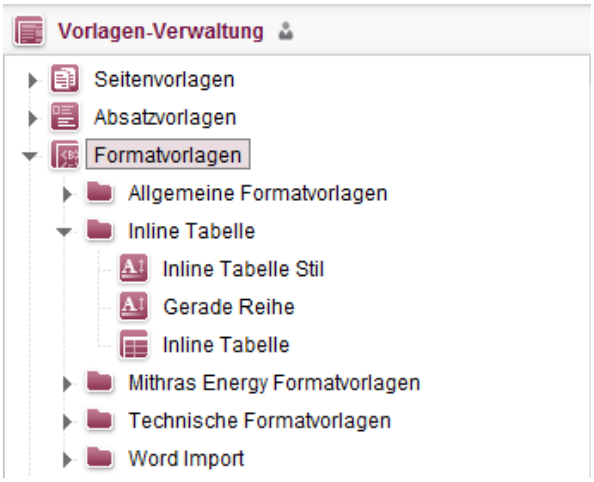


Abbildung 2-46: Baumansicht Vorlagen-Verwaltung – Stilvorlagen

2.8.1 Einleitung: Inline-Tabellen

Über den Dom-Editor (Eingabekomponente CMS_INPUT_DOM) können sogenannte "Inline-Tabellen" in den Textfluss integriert werden. Dabei können dem Redakteur beliebig viele Gestaltungsmöglichkeiten bis auf Zellenebene zur Verfügung gestellt werden.

Das Tabellenlayout wird einerseits von Tabellenformatvorlagen (siehe Kapitel 2.9 Seite 77), andererseits von Stilvorlagen (ab Kapitel 2.8.2 Seite 68) bestimmt. Über Stilvorlagen werden Tabellenlayoutmerkmale, also z. B. Hintergrundfarbe, Textausrichtung, Schriftart, Umbruchsteuerung, Rahmen und Rahmenabstände, festgelegt.

Jeder Tabellenformatvorlage können genau eine Standard-Stilvorlage (für die gesamte Tabelle) und mehrere weitere Stilvorlagen für eine gesonderte Darstellung einzelner Zellen der Tabelle zugeordnet werden (siehe Kapitel 2.9.1 Seite 80). Die Stilvorlagen definieren das Layout der einzelnen Tabellenzelle, z. B. die Hintergrundfarbe ("bgcolor"), die Ausrichtung von Text in der Zelle ("align") oder die Farbe der Schrift innerhalb der Zelle ("color").

Zur Verwendung der Inline-Tabellen im DOM-Editor muss daher zunächst eine Stilvorlage angelegt werden (siehe Kapitel 2.8.2 Seite 68).

Zur besseren Übersicht sollten Stil- und Tabellenformatvorlagen in einem Ordner (z. B. "Tabellen") zusammengefasst werden.

Icons der Inline-Tabellen:

Ordner

Tabellenformatvorlage

Stilvorlage

Die Inline-Tabellen-Funktionalität steht auch in WebEdit zur Verfügung.

Mit FirstSpiritVersion 5.0 werden als Lesehilfe an vielen Stellen der Vorlagen-Verwaltung Zeilennummern angezeigt. Mithilfe des Tastaturkürzels STRG + L können sie im JavaClient ein- und ausgeblendet werden.

2.8.2 Stilvorlage anlegen

Stilvorlagen werden im Bereich "Formatvorlagen" angelegt. Im Kontextmenü wird dazu der Punkt **Neu – Stilvorlage anlegen** ausgewählt. In dem sich öffnenden Fenster muss ein Referenzname für die Stilvorlage eingegeben werden. Die Angabe eines Anzeigenamens ist optional.

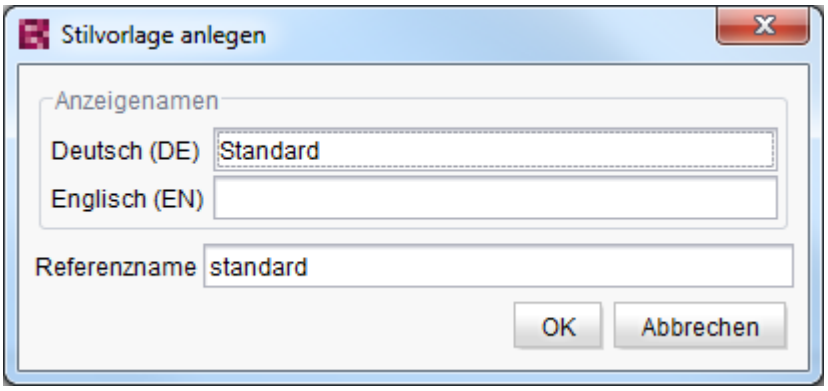


Abbildung 2-47: Neu – Stilvorlage anlegen

Mit einem Klick auf OK wird die neue Stilvorlage angelegt. Über den Formularbereich einer Stilvorlage können Eingabekomponenten angelegt werden, welche die Eigenschaften des Layouts, z. B. Hintergrundfarbe, Textausrichtung, Schriftart, Umbruchsteuerung, Rahmen und Rahmenabstände, beeinflussen (siehe Kapitel 2.8.3 Seite 69).

2.8.3 Formularbereich einer Stilvorlage

Anders als andere Formatvorlagen verfügen Stilvorlagen über ein Register "Formular". Innerhalb des Formularbereichs einer Stilvorlage können Eingabekomponenten zur Pflege von Layout-Attributen angelegt werden:

Abbildung 2-48: Formularbereich einer Stilvorlage

Einige vorgegebene Layout-Attribute (mit reservierten Bezeichnern) wirken sich direkt auf die Darstellung der Tabelle innerhalb des DOM-Editors aus:

- `bgcolor`: legt die Hintergrundfarbe einer Tabellenzelle fest
  (Beispiel siehe Kapitel 2.8.7.1 Seite 75)
- `color`: legt die Schriftfarbe eines Textes innerhalb der Tabellenzelle fest
  (Beispiel siehe Kapitel 2.8.7.2 Seite 75)
- `align`: legt die Ausrichtung eines Textes in der Tabellenzelle fest
  (Beispiel siehe Kapitel 2.8.7.3 Seite 76)

Die vorgegebenen Bezeichner dürfen nicht geändert werden. Die Attribute müssen in der Eingabekomponente immer mit name="Bezeichner" angegeben werden, z. B. <CMS_INPUT_TEXT name="bgcolor" .../>

Natürlich können neben diesen Standard-Attributen noch weitere frei definierte Attribute über Eingabekomponenten des Formularbereichs gepflegt werden, z. B. CSS-Attribute.

Unterstützte Eingabekomponenten zur Pflege der Layout-Attribute:

- `CMS_INPUT_TEXT` / `CMS_INPUT_TEXTAREA`: Textfeld für die Angabe eines Wertes, z. B. für die Hintergrundfarbe.
  (Beispiel siehe Kapitel 2.8.7.1 Seite 75)

- `CMS_INPUT_COMBOBOX`: Auswahl aus einer vorgegebenen Menge von Werten, z. B. für die Angabe einer Hintergrundfarbe oder einer Ausrichtung
  (Beispiel siehe Kapitel 2.8.7.2 Seite 75)
- `CMS_INPUT_RADIOBUTTON`: Auswahl aus einer vorgegebenen Menge von Werten, z. B. für die Angabe einer Hintergrundfarbe oder einer Ausrichtung
  (Beispiel siehe Kapitel 2.8.7.3 Seite 76)
- `CMS_INPUT_NUMBER`: Angabe eines Zahlenwerts (z. B. Wert für die Hintergrundfarbe einer Zelle)
- `FS_BUTTON`: Schaltfläche zur Aktivierung eines Skriptes oder zur Ausführung einer Klasse

Das Hinzufügen von Eingabekomponenten im Register „Formular“ ist durch die Code-Vervollständigung vereinfacht worden (siehe Kapitel 7 Seite 239).

Für alle Eingabekomponente, die im Formularbereich einer Stilvorlage verwendet werden, gilt: die Komponenten sollten sprachunabhängig definiert werden (`useLanguages="no"`). Die Sprachabhängigkeit der Komponente wird in diesem Fall durch die Sprachauswahl innerhalb der DOM-Editor-Instanz abgedeckt, die vom Redakteur bearbeitet wird.

Weitere Eingabekomponenten zur Pflege von Layout-Attributen (in Stilvorlagen) werden nicht unterstützt.

2.8.3.1 Bearbeitung des Layouts für Redakteure unterbinden

Die Pflege der Layout-Attribute kann für Redakteure unterbunden werden. Dazu muss innerhalb der Eingabekomponente das Attribut `hidden="yes"` oder eine entsprechende Regel definiert werden. Das Attribut `hidden="yes"` bewirkt, dass die Eingabekomponente nur innerhalb der Vorlagen-Verwaltung sichtbar ist, nicht aber bei der Pflege der Tabelle in der Inhalte-Verwaltung. Über das Attribut kann der Vorlagen-Entwickler also eine Bearbeitung des Layouts durch den Redakteur unterbinden und stattdessen definierte Werte für das Layout vorgeben, beispielsweise für die Hintergrundfarbe der Zellen (siehe Kapitel 2.8.4 Seite 71).

Ist das Attribut `hidden="yes"` (für alle Eingabekomponenten der Stilvorlage) definiert, besteht für den Redakteur keine Möglichkeit, die Layout-Eigenschaften

einer Tabellenzelle in der Inhalte-Verwaltung zu verändern. Der entsprechende Button "Eigenschaften Zelle" ist in diesem Fall inaktiv.

*Wurde die Pflege der Layout-Eigenschaften durch die Verwendung von Validatoren unterbunden, wird der Button **niemals** inaktiv. Auch dann nicht, wenn alle Eingabekomponenten ausgeblendet werden. In beiden Clients erscheint daraufhin ein leerer Dialog. Dieses Verhalten gilt ebenfalls für den Dialog zur Vorbelegung von Layout-Attributen (siehe Kapitel 2.8.4 Seite 71).*

Weiterführende Informationen zu Validatoren finden sie in der FirstSpirit Online-Dokumentation.

Sind dagegen einzelne Komponenten "sichtbar" (`hidden="no"`) und andere "versteckt" (`hidden="yes"`), so ist der Button "Eigenschaften Zelle" innerhalb des DOM-Editors (in der Inhalte-Verwaltung) aktiv, im folgenden Dialog werden dem Redakteur aber nur die "sichtbaren" Komponenten angezeigt.

Alle Komponenten der Stilvorlage werden nur dann angezeigt, wenn keine Einschränkungen durch den Vorlagen-Entwickler definiert wurden.

Abbildung 2-49: Button "Eigenschaften Zelle" innerhalb des DOM-Editors

Mit einem Klick auf den Button öffnet der Redakteur einen Dialog zum Bearbeiten der projektspezifischen Layout-Attribute (siehe *FirstSpirit Handbuch für Redakteure*).

2.8.4 Vorbelegung der Layout-Attribute

Innerhalb der Vorlagen-Verwaltung können Rückgriffwerte für die Layout-Attribute definiert werden. Die Vorbelegung einer Eingabekomponente (z. B. mit einem Farbwert) erfolgt in einem separaten Dialog, der im Register „Eigenschaften“ über den entsprechenden Button geöffnet werden kann.

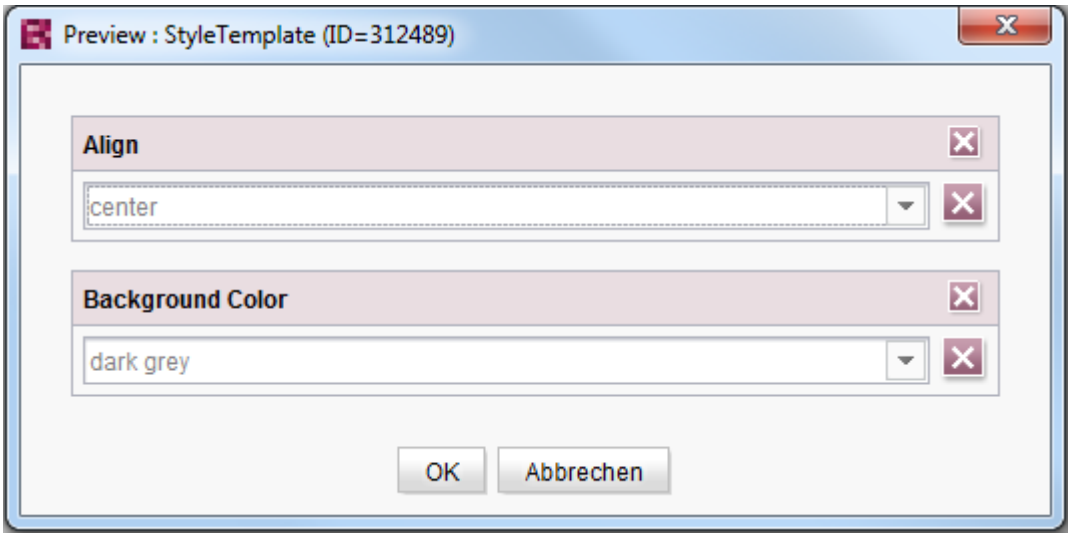


Abbildung 2-50: Vorschau Stilvorlage

Innerhalb des Dialogs kann eine Vorbelegung für die Eingabekomponente durch den Vorlagen-Entwickler definiert werden. In der Eingabekomponente für "Background Color" kann beispielsweise ein Farbwert ausgewählt werden (vgl. Abbildung 2-50). Dieser Farbwert wird beim Anlegen einer Inline-Tabelle im DOM-Editor für alle Tabellenzellen übernommen, die auf der entsprechenden Stilvorlage basieren.

Abhängig vom definierten Wert für das Attribut `hidden` innerhalb der Definition der Eingabekomponente, kann diese Vorbelegung vom Redakteur geändert werden (siehe Kapitel 2.8.3.1 Seite 70).

Ist die Bearbeitung möglich (`hidden="no"`), kann der Redakteur beim Bearbeiten einer Tabellenzelle innerhalb des DOM-Editors diesen Vorgabewert überschreiben (siehe *FirstSpirit Handbuch für Redakteure*).

Die Werte im Dialog können nur dann bearbeitet bzw. gespeichert werden, wenn die Vorlage gesperrt ist (Button "In Bearbeitungs-Modus wechseln")!

2.8.5 Ausgabekanal einer Stilvorlage

Innerhalb eines Ausgabekanals (z. B. "HTML") einer Stilvorlage können die Werte, die innerhalb der Eingabekomponenten eingepflegt wurden (siehe Kapitel 2.8.4 Seite 71), wieder ausgelesen werden.

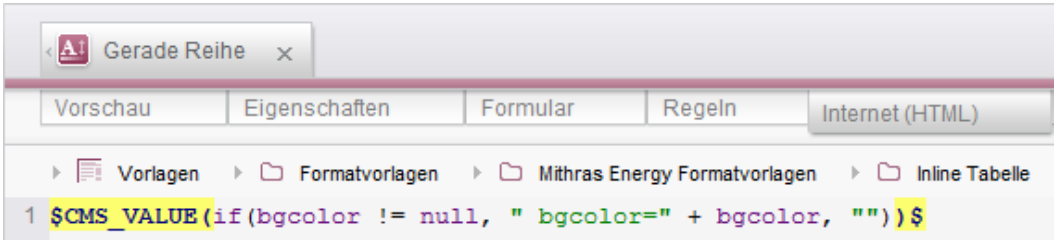


Abbildung 2-51: Ausgabekanal "HTML" einer Stilvorlage

Dazu muss der Name der Eingabekomponente mittels der Anweisung `$CMS_VALUE(...)$` ausgegeben werden:

```
$CMS_VALUE(if(bgcolor != null, " bgcolor=" + bgcolor, ""))$
```

oder

```
$CMS_IF(!bgcolor.isEmpty)$$CMS_VALUE(bgcolor)$$CMS_END_IF$
```

Weiterführende Informationen zur Ausgabe von Variablen siehe FirstSpirit Online-Dokumentation[3].

2.8.6 Verknüpfung mit Standard-Tabellen-Formatvorlagen

Über das Systemobjekt `#style` können die Werte der Stilvorlage mit den Standard-Formatvorlagen für die Generierung (und Vorschau) von Tabellen im Projekt verknüpft werden. Die von FirstSpirit zur Verfügung gestellten Standard-Formatvorlagen für Tabellen sind:

- Table (Kürzel: table): Formatierung für Tabellen
- Table-Cell (Kürzel: td): Formatierung für Tabellenzellen
- Table-Row (Kürzel: tr): Formatierung für Tabellenreihen

Wird beispielsweise innerhalb der Standard-Formatvorlage `td`, das Systemobjekt `#style` verwendet, werden die Werte, die innerhalb des Dialogs "Eigenschaften Zelle" vom Redakteur definiert wurden (siehe *FirstSpirit Handbuch für Redakteure*) bzw. die Werte, die vom Vorlagen-Entwickler für die Stilvorlage vorbelegt wurden (siehe Kapitel 2.8.4 Seite 71), bei der Generierung der Tabelle berücksichtigt.

[3] FirstSpirit Online-Dokumentation im Bereich Vorlagenentwicklung / Variablen

Beispiel für die Ausgabe im HTML-Kanal der Standard-Formatvorlage `td`:

```
<td$CMS_VALUE(#style)$
$CMS_VALUE(if(#cell.rowspan != 0, " rowspan='" + #cell.rowspan +
"'"))$
$CMS_VALUE(if(#cell.colspan != 0, " colspan='" + #cell.colspan +
"'"))$>
$CMS_VALUE(if(#content.isEmpty, " ", #content))$
</td>
```

Weiterführende Informationen darüber, wie auf Eigenschaften und Informationen von Tabellen und ihren Inhalten zugegriffen werden kann, siehe FirstSpirit Online-Dokumentation, Systemobjekte #cell, #content, #table und #tr im Bereich Vorlagenentwicklung / Vorlagensyntax / Systemobjekte.

Die Werte der Layout-Attribute, die durch den Redakteur (bzw. den Vorlagen-Entwickler) im Dialog "Eigenschaften Zelle" definiert wurden, werden nun bei der Generierung der Tabelle berücksichtigt (siehe Abbildung 2-52):

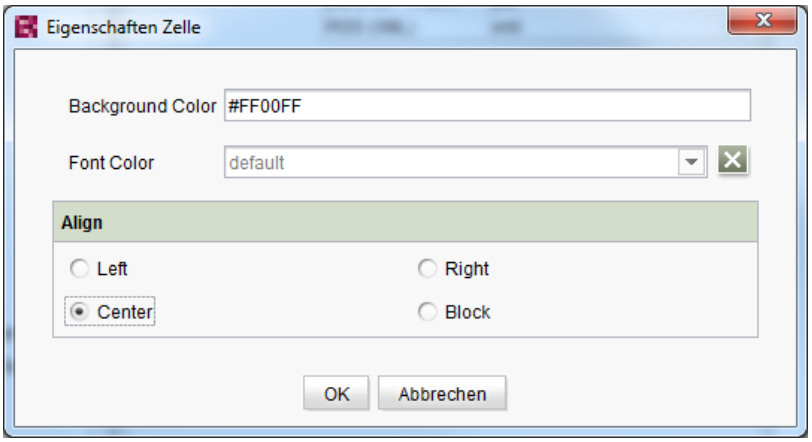


Abbildung 2-52: Eigenschaften der Tabellenzelle

Der Quelltext der Tabellenzelle wird nun beispielsweise folgendermaßen generiert:

```
<table>
<tr>
<td bgcolor="#ff00ff" align="center" color="#00ddee" rowspan='1'
colspan='1'>Dies ist ein Text.</td>
..
</tr>
</table>
```

2.8.7 Beispiele

2.8.7.1 Beispiel: Text-Eingabekomponente zur Eingabe einer Hintergrundfarbe

Definition der Komponente im Formularbereich:

```
<CMS_MODULE>
  <CMS_INPUT_TEXT name="bgcolor" useLanguages="no">
    <LANGINFOS>
      <LANGINFO lang="*" label="Background color:"/>
    </LANGINFOS>
  </CMS_INPUT_TEXT>
</CMS_MODULE>
```

`name="bgcolor"`: Die Eingabekomponente verwendet den Schlüsselwert "bgcolor" für die Definition einer Hintergrundfarbe. Dieser Name darf nicht geändert werden, da es sich um einen festen Schlüsselwert handelt.

Eingabe eines Farbwertes über die Eingabekomponente:

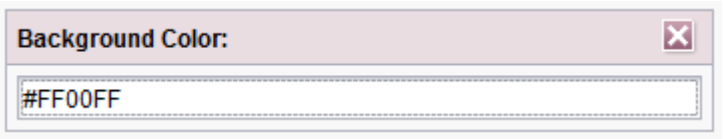


Abbildung 2-53: Eingabekomponente zur Eingabe einer Hintergrundfarbe

Zur Ausgabe des Wertes innerhalb des Ausgabekanals der Stilvorlage siehe Kapitel 2.8.5 Seite 72.

2.8.7.2 Beispiel: Eingabekomponente zur Eingabe einer Schriftfarbe

Definition der Komponente im Formularbereich:

```
<CMS_MODULE>
  <CMS_INPUT_COMBOBOX name="color" useLanguages="no">
    <ENTRIES>
      <ENTRY value="">
        <LANGINFOS>
          <LANGINFO lang="*" label="default"/>
        </LANGINFOS>
      </ENTRY>
      <ENTRY value="#ee00ff">
        <LANGINFOS>
          <LANGINFO lang="*" label="superior"/>
        </LANGINFOS>
```

```
      </ENTRY>
      <ENTRY value="#00ddee">
        <LANGINFOS>
          <LANGINFO lang="*" label="lightGrey"/>
        </LANGINFOS>
      </ENTRY>
    </ENTRIES>
    <LANGINFOS>
      <LANGINFO lang="*" label="Font Color"/>
    </LANGINFOS>
  </CMS_INPUT_COMBOBOX>
</CMS_MODULE>
```

Auswahl eines Farbwertes über Eingabekomponente:

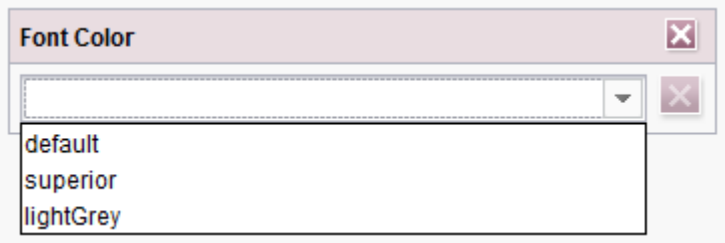


Abbildung 2-54: Eingabekomponente zur Auswahl eines Farbwertes für die Schriftfarbe

Zur Ausgabe des Wertes innerhalb des Ausgabekanals der Stilvorlage siehe Kapitel 2.8.5 Seite 72.

2.8.7.3 Beispiel: Eingabekomponente zur Eingabe einer Textausrichtung

Definition der Komponente im Formularbereich:

```
<CMS_MODULE>
  <CMS_INPUT_RADIOBUTTON name="align" useLanguages="no">
    <ENTRIES>
      <ENTRY value="">
        <LANGINFOS>
          <LANGINFO lang="*" label="Left"/>
        </LANGINFOS>
      </ENTRY>
      <ENTRY value="right">
        <LANGINFOS>
          <LANGINFO lang="*" label="Right"/>
        </LANGINFOS>
      </ENTRY>
```

```
      <ENTRY value="center">
        <LANGINFOS>
          <LANGINFO lang="*" label="Center"/>
        </LANGINFOS>
      </ENTRY>
      <ENTRY value="block">
        <LANGINFOS>
          <LANGINFO lang="*" label="Block"/>
        </LANGINFOS>
      </ENTRY>
    </ENTRIES>
    <LANGINFOS>
      <LANGINFO lang="*" label="Align:"/>
    </LANGINFOS>
  </CMS_INPUT_RADIOBUTTON>
</CMS_MODULE>
```

Auswahl einer Textausrichtung über die Eingabekomponente:

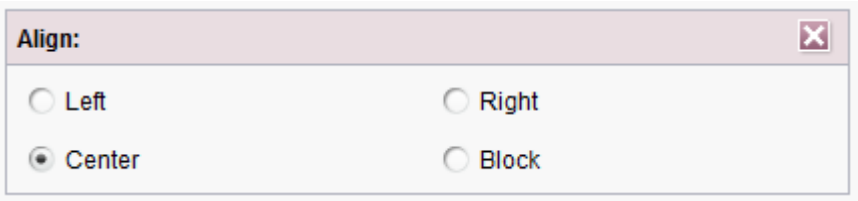


Abbildung 2-55: Eingabekomponente zur Auswahl einer Textausrichtung

Zur Ausgabe des Wertes innerhalb des Ausgabekanals der Stilvorlage siehe Kapitel 2.8.5 Seite 72.

2.9 Tabellenformatvorlagen

Tabellenformatvorlagen sind für die Erstellung von so genannten Inline-Tabellen erforderlich (siehe Kapitel 2.8.1 Seite 67). Für jedes gewünschte Tabellen-Layout muss im Bereich "Formatvorlagen" eine Tabellenformatvorlage angelegt werden.

Dazu wird im Kontextmenü der Punkt **Neu – Tabellenformatvorlage anlegen** ausgewählt. In dem sich öffnenden Fenster muss ein Referenzname für die Tabellenformatvorlage eingegeben werden. Die Angabe eines Anzeigenamens ist optional.

Eine ausführliche Beschreibung zu Referenz- und Anzeigenamen befindet sich in Kapitel 2.2.1 Seite 21.

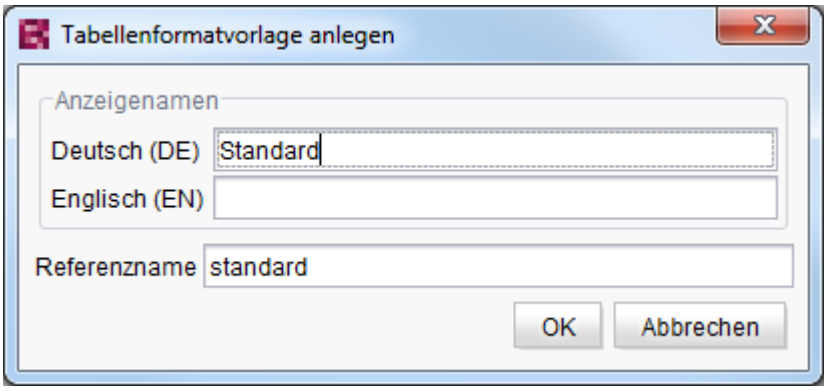


Abbildung 2-56: Neu – Tabellenformatvorlage anlegen

Nach einem Klick auf OK öffnet sich das Register "Eigenschaften". Hier können die Größe der Tabelle definiert sowie die in Kapitel 2.8.2 angelegten Stilvorlagen zugewiesen werden.

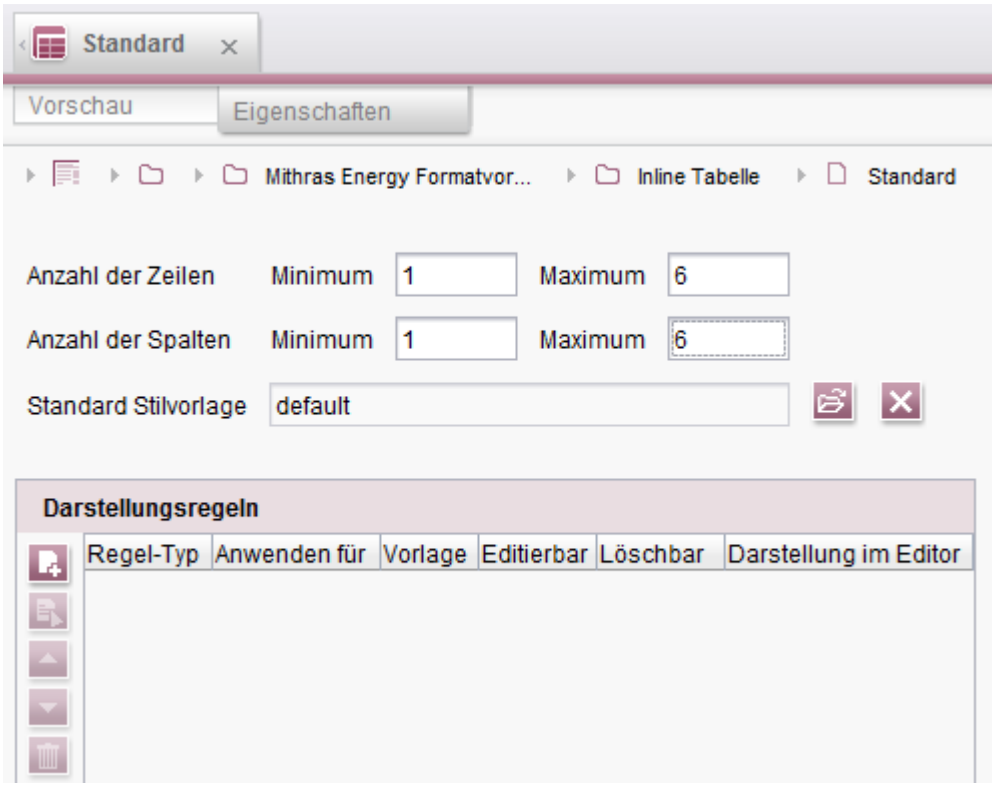


Abbildung 2-57: Tabellenformatvorlage

Anzahl der Zeilen / Spalten: Über die Felder **Minimum** und **Maximum** wird festgelegt, wie viele Zeilen bzw. Spalten die Tabelle mindestens umfassen muss

bzw. höchstens umfassen darf.

Die Mindestanzahl für Zeilen und Spalten ist dabei standardmäßig 2.

Fügt der Redakteur später eine Inline-Tabelle mit dieser Tabellenformatvorlage in den DOM-Editor ein, wird sie dort automatisch mit der hier angegebenen Mindestanzahl von Zeilen bzw. Spalten angelegt. Die Standardwerte können beim Bearbeiten der Tabelle durch den Redakteur nicht über- bzw. unterschritten werden. Die entsprechenden Buttons werden in diesem Fall deaktiviert. Ist beispielsweise eine Minimal-Zeilenanzahl von vier Zeilen definiert, wird der Button "Zeile löschen" im DOM-Editor deaktiviert, sobald die Tabelle nur noch vier Zeilen enthält.

Standard Stilvorlage: In diesem Feld kann durch einen Klick auf das Icon die gewünschte Stilvorlage ausgewählt werden, die der gesamten Tabelle zugrunde liegen soll. In dem sich öffnenden Fenster werden alle zur Verfügung stehenden Stilvorlagen angezeigt.

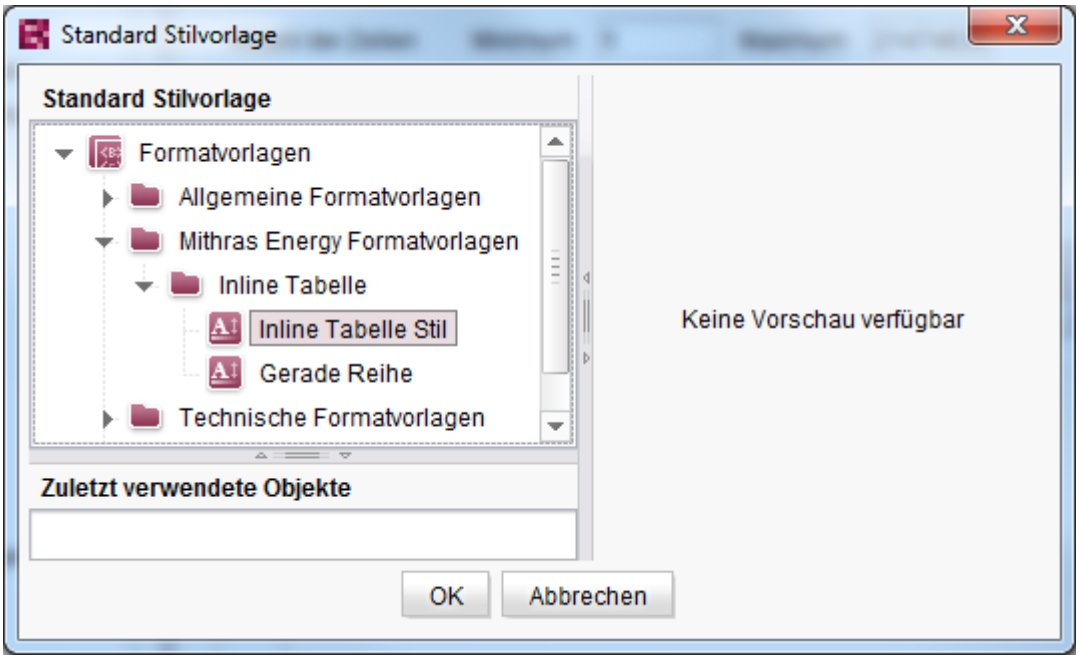


Abbildung 2-58: Tabellenformatvorlage – Standard Stilvorlage

Die gewünschte Stilvorlage kann aus der Baumstruktur ausgewählt und die Auswahl durch OK bestätigt werden.

2.9.1 Erstellen und Bearbeiten von Darstellungsregeln

Als Grundlage für das Layout einer Tabelle gilt die in der Tabellenformatvorlage definierte Standard-Stilvorlage (vgl. Abbildung 2-58). Darüber hinaus können dem Redakteur im Bereich **Darstellungsregeln** weitere Layoutmöglichkeiten zur Formatierung von Zeilen, Spalten und einzelner Zellen zur Verfügung gestellt werden, die die Layoutvorgaben der Standard-Stilvorlage überschreiben. Diese Layoutmöglichkeiten basieren auf den zuvor erstellten Stilvorlagen (siehe Kapitel 2.8.2 Seite 68).

Über das Icon  werden neue Darstellungsregeln angelegt. Nach einem Klick darauf öffnet sich folgendes Fenster:

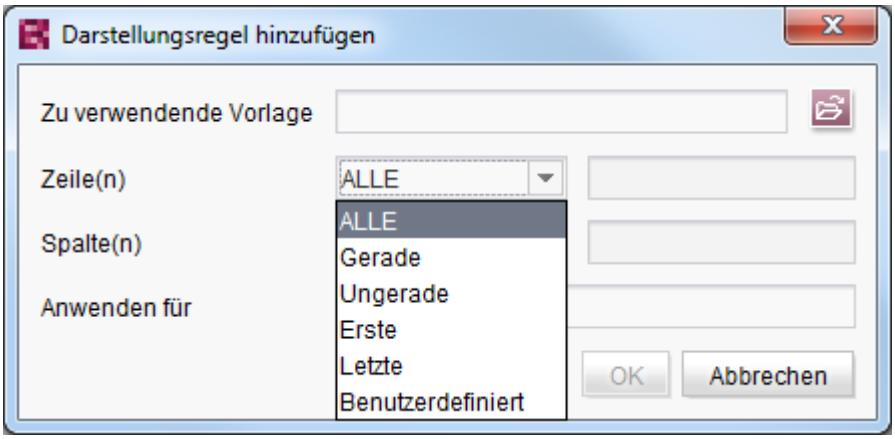


Abbildung 2-59: Tabellenformatvorlage – Darstellungsregel

Mit einem Klick auf das Icon  kann die gewünschte Stilvorlage ausgewählt werden, die für die Darstellungsregel angewendet werden soll. In dem sich öffnenden Fenster werden alle zur Verfügung stehenden Stilvorlagen angezeigt.

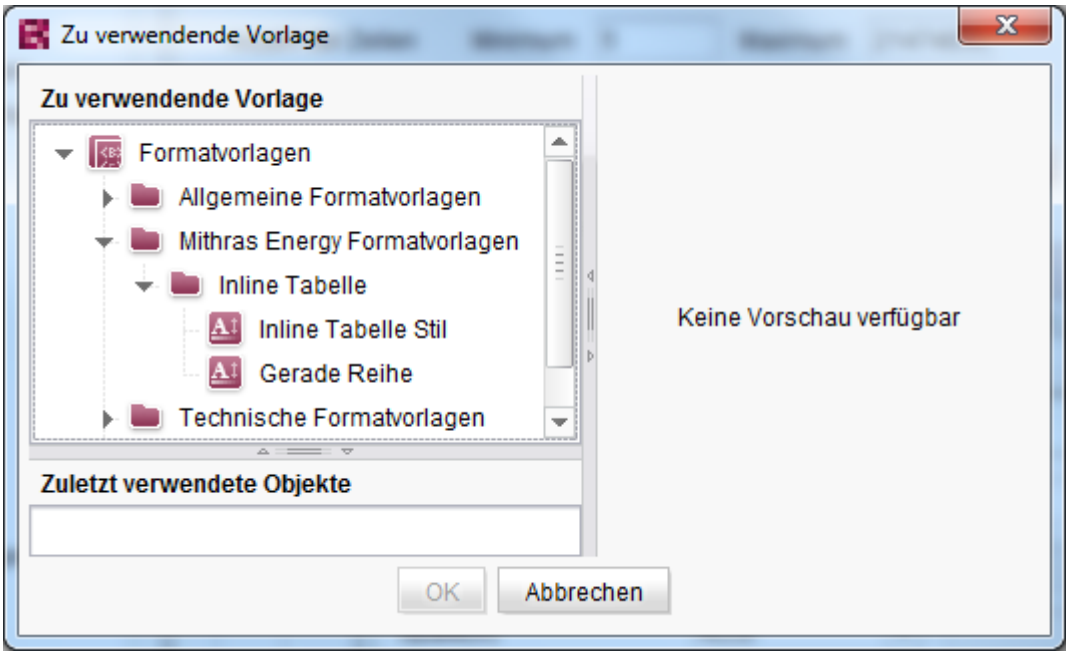


Abbildung 2-60: Tabellenformatvorlage – Stilvorlage

Die gewünschte Stilvorlage kann aus der Baumstruktur ausgewählt und die Auswahl durch OK bestätigt werden.

Über die Comboboxen **Zeile(n)** und **Spalte(n)** werden Bedingungen für die Anwendung der Regel und damit der Anwendung ausgewählten Stilvorlage definiert. Beide Bedingungen müssen erfüllt sein, damit die Regel angewendet wird.

ALLE: Die Darstellungsregel trifft für alle Zeilen bzw. Spalten ohne Einschränkung zu. Wird beispielsweise die Option "ALLE Spalten" ausgewählt, berücksichtigt die Darstellungsregel nur die unter der Option "Zeilen" definierte Einschränkung und umgekehrt.

Es ist nicht möglich, eine Darstellungsregel zu definieren, die ALLE Spalten und ALLE Zeilen betrifft. Eine solche Regel würde der Standard Stilvorlage entsprechen.

Gerade: Die Darstellungsregel trifft für gerade Zeilen bzw. Spalten zu (beginnend mit der zweiten Zeile/Spalte).

Ungerade: Die Darstellungsregel trifft für ungerade Zeilen bzw. Spalten zu (beginnend mit der ersten Zeile/Spalte).

Erste: Die Darstellungsregel trifft für die erste Zeile bzw. Spalte zu.

Letzte: Die Darstellungsregel trifft für die letzte Zeile bzw. Spalte zu.

Benutzerdefiniert: Die Darstellungsregel trifft für eine bestimmte Zeile bzw. Spalte zu. Ist diese Option aktiviert, muss in das Feld rechts neben der Combobox die Zahl der gewünschten Zeile/Spalte eingetragen werden.

Im Feld "Anwenden für" wird noch angezeigt, für welche Zeile(n) und Spalte(n) die gewählte Stilvorlage gilt.

Beispiele:

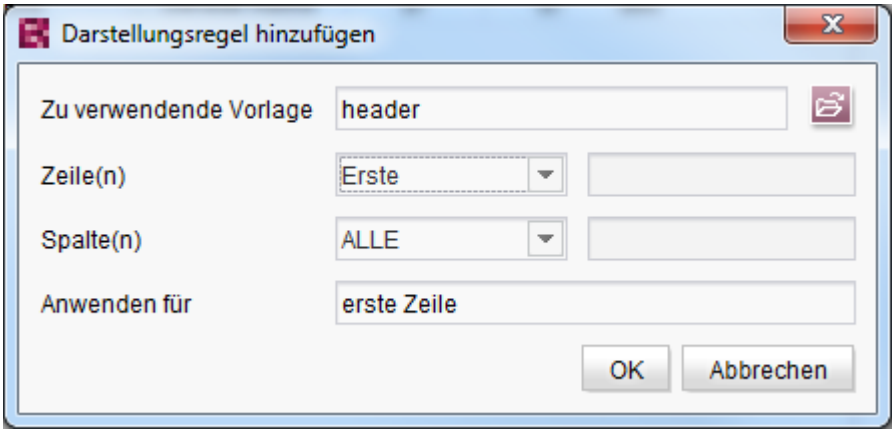


Abbildung 2-61: Darstellungsregeln – Beispiel 1

In diesem Beispiel wird die Stilvorlage "header" für die **Erste Zeile** in **ALLEN Spalten** angewendet, also in allen Zellen der ersten Zeile.

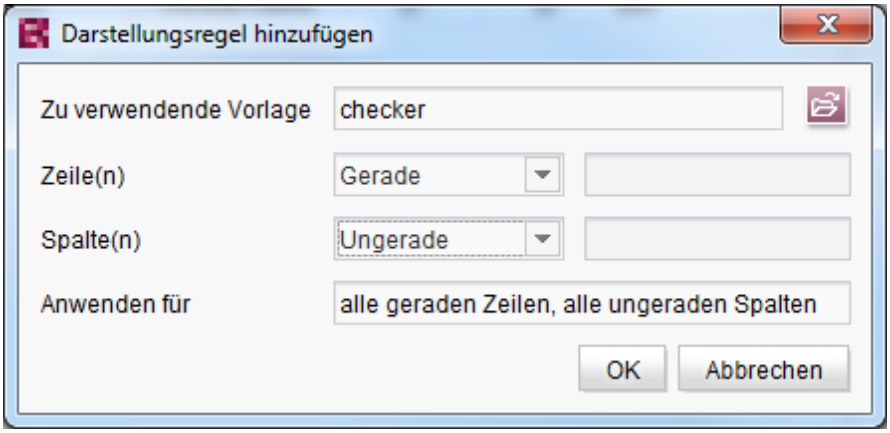


Abbildung 2-62: Darstellungsregeln – Beispiel 2

In diesem Beispiel wird die Stilvorlage "checker" für **Geraden Zeilen** in **Ungeraden Spalten** angewendet, also in allen Zellen, für die beide Bedingungen zutreffen.

Die Einstellungen für die neue Darstellungsregel werden mit einem Klick auf OK gespeichert. Die Darstellungsregel erscheint anschließend in folgender Liste:

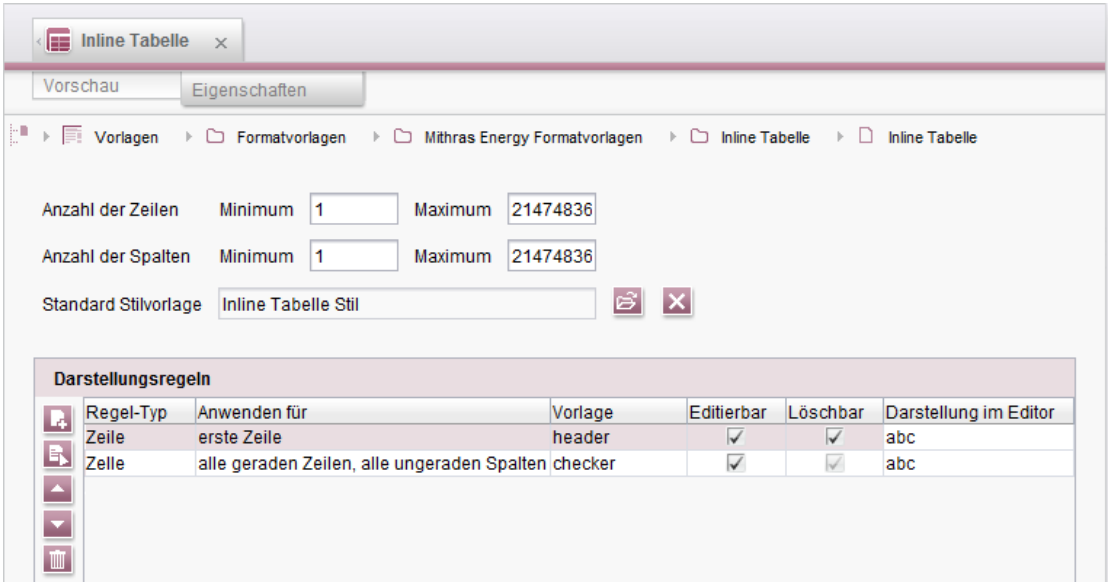


Abbildung 2-63: Liste der Darstellungsregeln

Regel-Typ: Gibt an, ob die Regel für Zeilen, Spalten oder Zellen gültig ist.

Anwenden für: Gibt an, auf welche Bereiche der Tabelle die Regel angewendet wird.

Vorlage: Gibt an, welche Stilvorlage angewendet wird.

Editierbar: Diese Checkbox ist standardmäßig aktiviert, so dass der Redakteur die Eigenschaften der Zelle(n), für die diese Darstellungsregel gilt, ändern kann. Wird die Checkbox deaktiviert, kann der Redakteur die Eigenschaften die betreffenden Zelle(n) nicht ändern.

Löschbar: Diese Checkbox ist standardmäßig aktiviert, wenn es sich um eine Regel für Zeilen oder Spalten handelt. Der Redakteur kann die Zeile(n) bzw. Spalten, für die diese Darstellungsregel gilt, löschen. Wird die Checkbox deaktiviert, kann der Redakteur die betreffenden Zeile(n) bzw. Spalten nicht löschen. Gilt die Regel für eine Zelle, ist die Checkbox aktiviert. Sie kann nicht deaktiviert werden, da einzelne Zellen nicht aus Tabellen gelöscht werden können.

Darstellung im Editor: Diese Spalte zeigt die Hintergrundfarbe sowie die Textausrichtung der Zeile/Spalte/Zelle, sofern entsprechende Werte definiert sind.

 Bearbeiten: Durch einen Klick auf dieses Icon (oder durch einen Doppelklick auf die Darstellungsregel) kann eine bereits vorhandene Darstellungsregel zur Bearbeitung geöffnet werden.

 Eine Position aufwärts: Sind mehrere Darstellungsregeln vorhanden, können sie über dieses Icon um eine Position in der Liste nach oben verschoben werden.

 Eine Position abwärts: Sind mehrere Darstellungsregeln vorhanden, können sie über dieses Icon um eine Position in der Liste nach unten verschoben werden.

 Löschen: Durch einen Klick auf dieses Icon wird die markierte Darstellungsregel gelöscht.

Die Spalten dieser Liste können je nach Bedarf per Klick auf die Spaltenlinien und mit gedrückter Maustaste in der Breite verändert werden.

 *Sind mehrere Regeln in der Liste vorhanden, werden diese von oben nach unten ausgewertet.*

2.9.2 Auswertungsreihenfolge

Die Tabellenformatvorlagen, Stilvorlagen und Darstellungsregeln, die in den Kapiteln 2.8.2 bis 2.9.1 erstellt wurden, enthalten Formatierungsangaben. Diese werden folgendermaßen ausgewertet:

1. Zunächst werden die **Darstellungsregeln** in der Liste (siehe Abbildung 2-63: Liste der Darstellungsregeln) von oben nach unten ausgewertet.
2. Auf Zellen, auf die keine Darstellungsregel zutrifft, wird die **Standard Stilvorlage** angewendet.

2.9.3 Inline-Tabelle in den DOM-Editor einfügen

Um dem Redakteur Inline-Tabellen im DOM-Editor zur Verfügung zu stellen, muss dieser Eingabekomponenten-Typ in die gewünschte Absatzvorlage eingefügt werden. Dazu muss der Eingabekomponente CMS_INPUT_DOM der Parameter `table="yes"` hinzugefügt werden.

Beispiel:

```
<CMS_INPUT_DOM name="st_inlinetable" table="yes">
  <LANGINFOS>
      <LANGINFO lang="*" label="Table"/>
  </LANGINFOS>
</CMS_INPUT_DOM>
```

Die Eingabekomponente kann folgendermaßen aussehen:

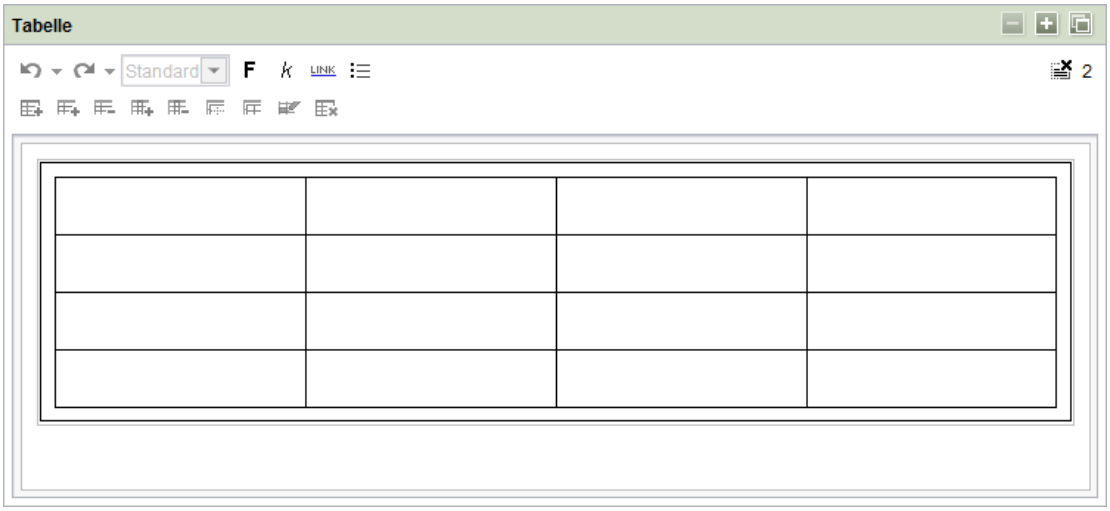


Abbildung 2-64: DOM-Editor mit Inline-Tabelle

2.10 Verweisvorlagen

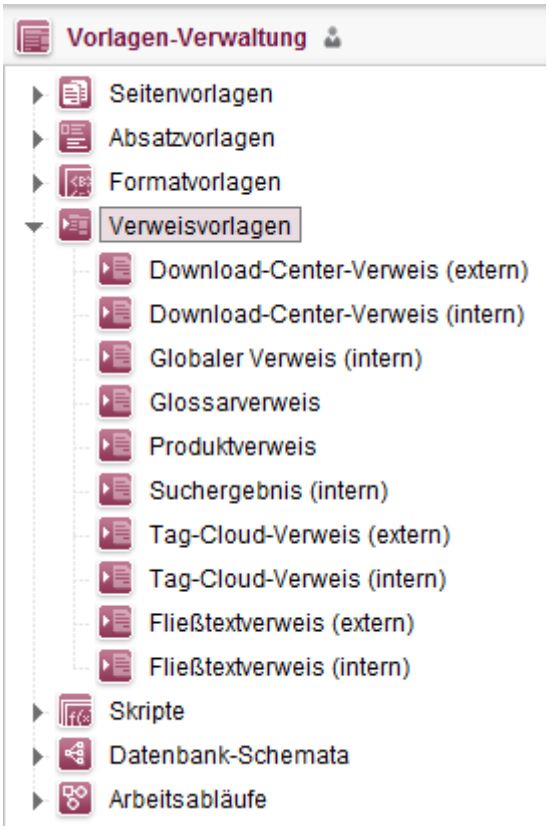


Abbildung 2-65: Baumansicht Vorlagen-Verwaltung – Verweisvorlagen

Mithilfe von Verweisvorlagen können Vorlagenentwickler das Layout von Verweisen innerhalb eines FirstSpirit Projekts detailliert vorgeben. Die Redakteure geben alle erforderlichen Inhalte über eine Eingabemaske ein. Welche Felder in der Eingabemaske ausgefüllt werden können, ist abhängig von der Konfiguration der Verweisvorlagen. Mit FirstSpirit Version 5.0 wurden die zuvor statischen Link-Editoren vollständig auf generische Links umgestellt, von denen unterhalb des Knotens „Verweisvorlagen“ beliebig viele Instanzen angelegt werden können. Jede Instanz muss einen eindeutigen Namen besitzen.

Durch die Möglichkeit, mehrere Instanzen (Verweisvorlagen) zu definieren, können im Formularbereich einer Seiten- oder Absatzvorlage für unterschiedliche Eingabekomponenten unterschiedliche Verweisvorlagen definiert werden. Auf diese Weise können interne Verweise, die vom Redakteur beispielsweise innerhalb der Eingabekomponente DOM-Editor eingepflegt werden, anders konfiguriert und dargestellt werden, als Verweise, die beispielsweise innerhalb der Eingabekomponente FS_LIST gepflegt werden.

Weiterführende Informationen zu Verweisvorlagen siehe "FirstSpirit Online Dokumentation".

Nachträgliche Änderungen an Verweisvorlagen können dazu führen, dass die Vorschau von bereits in die zugehörigen Eingabemasken eingegebenen Inhalten nicht aktuell ist. Dies kann durch nochmaliges Speichern der Eingabemasken behoben werden.

2.10.1 Standard-Verweistypen

Da in FirstSpirit Version 5.0 die statischen Links nicht mehr unterstützt werden und somit nur noch generische Links existieren, ist die Auswahl eines Verweistypen nicht mehr notwendig. Die entsprechende Auswahl im Neu-Dialog wurde aufgrund dessen entfernt.

Nähere Informationen hierzu finden Sie in den Releasenotes zu FirstSpirit 5.0.

2.10.2 Generische Link-Editoren

Die Konfigurationsmöglichkeiten von Verweisvorlagen wurden durch die Einführung der "generischer Link-Editoren" erheblich erweitert. Die Konfiguration wird nun, analog zu Seiten- und Absatzvorlagen, mithilfe von Eingabekomponenten im Formularbereich erstellt. Alle Eingabemöglichkeiten für die Pflege von Verweisen können dabei über die reguläre Formularsyntax von FirstSpirit abgebildet werden.

Das Hinzufügen von Eingabekomponenten im Register „Formular“ ist durch die Code-Vervollständigung vereinfacht worden (siehe Kapitel 7 Seite 239).

Im Zuge der Einführung der generischen Link-Editoren können die Verweisvorlagen nun auch in Ordnern strukturiert werden.

Die herkömmlichen Eingabemöglichkeiten für Verweise (der statischen Link-Editoren) können natürlich auch über die neuen, generischen Editoren erzeugt werden. Um alle Funktionalitäten der bisherigen statischen Link-Editoren auf die neuen generischen Editoren abbilden zu können, wurden einige neue Eingabekomponenten eingeführt. Beispielsweise konnte die Auswahl über das Feld "mediaref" der statischen Link-Editoren nicht auf die zuvor vorhandenen Eingabekomponenten abgebildet werden. Die Eingabekomponenten

CMS_INPUT_PICTURE und CMS_INPUT_FILE unterstützten jeweils nur die Auswahl eines Referenztyps, also entweder Bilder oder Dateien, nicht aber beide. Daher ist die Eingabekomponenten FS_REFERENCE eingeführt worden, die beliebige Referenztypen unterstützt.

Analog dazu wurden für die Abbildung eines Links auf Datenbankinhalte Erweiterungen zur Eingabekomponenten CMS_INPUT_ OBJECTCHOOSER zur Auswahl von Datensätzen aus *einer* bestimmten Datenbank-Tabelle und die neue Eingabekomponente FS_DATASET zur Auswahl von Datensätzen aus *beliebigen* Datenbank-Tabellen eingeführt.

Für generische Link-Editoren wird die Trennung zwischen Definition (Formular) und Ausgabe aufgehoben. Unterhalb des Wurzelknotens „Verweisvorlagen“ oder unterhalb eines Ordners braucht nun nur eine neue Vorlage angelegt werden.

Das Register "Eigenschaften" kann folgendermaßen aussehen:

Abbildung 2-66: Generischer Link – Register "Eigenschaften"

Auf dem Register "Formular" kann jede Eingabekomponente verwendet werden.

Weiterführende Informationen zu den Generischen Link-Editoren siehe "FirstSpirit Online Dokumentation" – Kapitel "Verweisvorlagen" / "Generische Link-Editoren".

2.11 Skripte

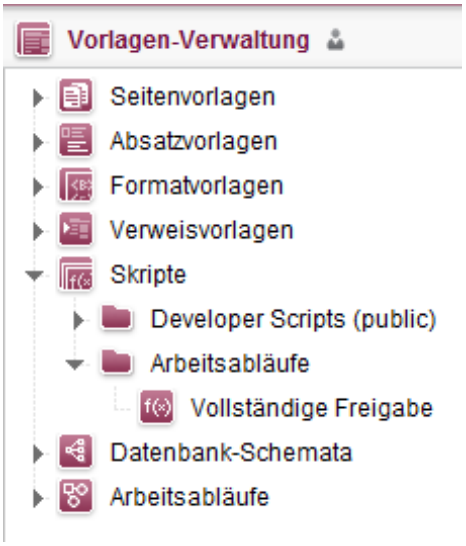


Abbildung 2-67: Baumansicht Vorlagen-Verwaltung – Skripte

Mithilfe von Skripten lassen sich unterschiedliche Arten von Bedienungsabläufen in FirstSpirit automatisieren. Ein Skript dient dabei zur Beschreibung des auszuführenden Ablaufes und kann bei Bedarf Veränderungen an FirstSpirit Datenstrukturen vornehmen. Skripte ermöglichen eine schnelle Umsetzung von Funktionalitäten, die in FirstSpirit noch nicht vorhanden sind. Weitere Einsatzgebiete sind zum Beispiel komplexe Migrationsszenarien und die Anbindung externer Systeme.

Die unterstützte Skriptsprache in FirstSpirit ist BeanShell[4]. Die BeanShell-Syntax ist stark an JAVA angelehnt, bietet aber zahlreiche Vereinfachungen, beispielsweise dynamische statt statische Typisierung globaler Variablen und Funktionen sowie einen (eingeschränkten) reflexiven Zugriff auf das Programm selbst und viele weitere Funktionalitäten.

Scripting mit BeanShell bietet eine hohe Flexibilität für den Vorlagenentwickler. Das Arbeiten mit Skripten ist jedoch nicht trivial. Daher sollte vor dem Einsatz eines Skripts genau geprüft werden, ob nicht bereits eine entsprechende Funktionalität in FirstSpirit vorhanden ist.

[4] Weitere Informationen über diese Skriptsprache befinden sich auf der Website www.beanshell.org, die zudem ein ausführliches Manual (EN) zur Verfügung stellt.

Weiterführende Informationen zur Skriptentwicklung in BeanShell siehe "FirstSpirit Online Dokumentation" – Kapitel "Scripting".

Beispiele für die Verwendung von Skripten in Arbeitsabläufen (siehe Kapitel 4.8 Seite 178).

2.11.1 Register Eigenschaften

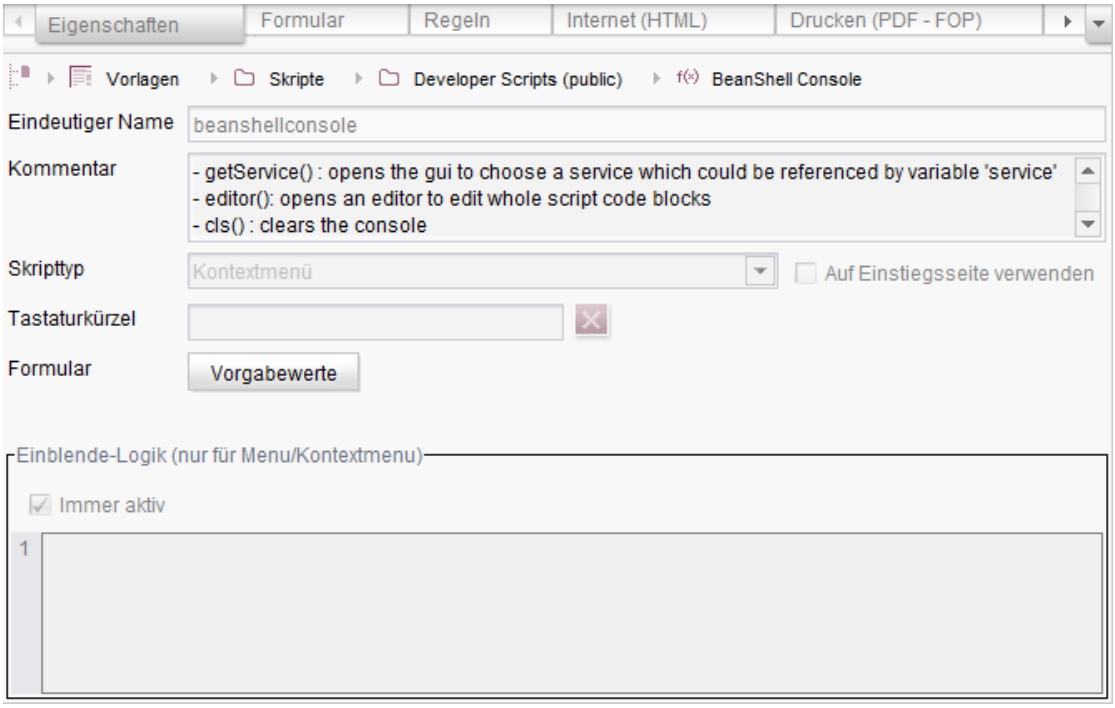


Abbildung 2-68: Skripte – Register "Eigenschaften"

Eindeutiger Name: Referenzname des Skriptes.

Kommentar: Hier kann ein optionaler Kommentar eingegeben werden, der die Skripte näher beschreibt.

Skripttyp: Hier kann der Kontext eingestellt werden, in dem ein Skript ausgeführt werden soll:

- Vorlage: Das Skript kann innerhalb einer Vorlage über `$CMS_RENDER(script:..)$` aufgerufen und ausgeführt werden, z. B. zum Rendern bestimmter Inhalte für den PDF-Ausgabekanal:

```
<fo:table table-layout="fixed" width="170mm">
$CMS_RENDER(script:"fotablecolumns",colWidth:set_cw,colNumbers:set_cn)$
<fo:table-body>
          $CMS_VALUE(#content)$
</fo:table-body>
</fo:table>
```

- Menü: Das Skript kann über das Menü "Extras" – "Skript ausführen" ausgeführt werden.

Skripte des Typs „Menü“ werden auch im WebEdit-Menü „Aktionen“ angezeigt.

- Kontextmenü: Das Skript kann über das Kontextmenü auf einem bestimmten Element in der Baumansicht des FirstSpirit JavaClients aufgerufen und ausgeführt werden.
- Uninterpretiert: das Skript wird beim Speichern nicht auf die BeanShell-Syntax geprüft. Es kann damit beispielsweise auch HTML-Syntax gespeichert werden (beispielsweise um Listenelemente darzustellen). **Diese Skripte sollten nicht mehr verwendet werden.**

In Projekten, die uninterpretierte Skripte verwenden, sollen die entsprechenden Vorlagen auf Formatvorlage umgestellt werden. Aus diesem Grund ist das Speichern von Skripten dieses Typs zukünftig nicht mehr möglich, bis eine Umstellung auf einen anderen Skripttyp erfolgt ist. In neu angelegten Skripten steht dieser Typ nicht mehr zur Verfügung.

Auf Einstiegsseite verwenden: Diese Option kann für Skripte vom Typ "Menü" aktiviert werden. Dieses Skript wird dann, abhängig von den Einstellungen im Bereich "Einblendelogik" (siehe unten), auf der Projekteinstiegsseite im Bereich "Meine Aktionen" angezeigt und kann direkt per Klick ausgeführt werden.

Tastaturkürzel: Für Skripte kann in diesem Feld ein eindeutiges Tastaturkürzel definiert werden. Das Skript muss in diesem Fall nicht mehr über das Kontextmenü oder das Menü "Extras" ausgeführt werden, sondern kann direkt über das festgelegte Tastaturkürzel aufgerufen werden. Um ein neues Tastaturkürzel zu definieren, muss sich der Cursor innerhalb des Feldes befinden. Anschließend genügt es, die gewünschte Tastenkombination über die Tastatur einzugeben. Die Eingabe wird dann in das Eingabefeld übernommen. Eine Texteingabe ist nicht möglich. Zum Ändern des Tastenkürzels den Cursor erneut im Feld positionieren

und anschließend die neue Tastenkombination aufrufen. Um ein definiertes Tastenkürzel für das Skript zu löschen, das Icon drücken.

Die Tastaturkürzel können nur für Skripte vom Typ "Kontextmenü" oder "Menü" verwendet werden.

Einblendelogik (nur für Menü/Kontextmenü): Über die Einblende-Logik können Skripte abhängig von bestimmten Eigenschaften eingeblendet oder ausgeblendet werden (analog zur Einblendelogik von Arbeitsabläufen, vgl. Kapitel 4.5.2 Seite 149). Beispielsweise kann ein Skript vom Skripttyp "Kontextmenü" nur dann angezeigt werden, wenn das Kontextmenü auf einer Seitenreferenz in der Struktur-Verwaltung aufgerufen wird:

```
//!Beanshell
import de.espirit.firstspirit.access.store.sitestore.PageRef;
e = context.getStoreElement();
return (e instanceof PageRef);
```

Immer aktiv: Soll die Einblende-Logik deaktiviert werden, kann die Checkbox "Immer aktiv" aktiviert werden. Das Skript wird in diesem Fall, unabhängig von der Einblende-Logik, immer eingeblendet. Die hinterlegte Einblendelogik wird zwar nicht mehr ausgewertet, bleibt aber enthalten und kann durch das Deaktivieren der Checkbox wieder aktiviert werden.

2.11.2 Register Formular

Im Register "Formular" lassen sich analog zu Seiten- und Absatzvorlagen individuelle Eingabekomponenten definieren, die zur Laufzeit des Skripts aufgerufen werden können. Die Werte der Eingabekomponenten können an das Skript zur Verarbeitung zurückgegeben werden (analog zur Formularunterstützung innerhalb von Arbeitsabläufen, vgl. Kapitel 4.4 Seite 145).

Das Hinzufügen von Eingabekomponenten im Register „Formular“ ist durch die Code-Vervollständigung vereinfacht worden (siehe Kapitel 7 Seite 239).

Mit FirstSpiritVersion 5.0 werden als Lesehilfe an vielen Stellen der Vorlagen-Verwaltung Zeilennummern angezeigt. Mithilfe des Tastaturkürzels STRG + L können sie im JavaClient ein- und ausgeblendet werden.

*In der "FirstSpirit Online Dokumentation" ist eine Auflistung aller zur Verfügung stehenden Eingabeelemente zu finden. Unter dem Menüpunkt **Vorlagenentwicklung – Formulare** werden die Eingabeelemente mit allen Attributen und einem schematischen Beispiel erläutert.*

2.11.3 Register Vorlagensätze

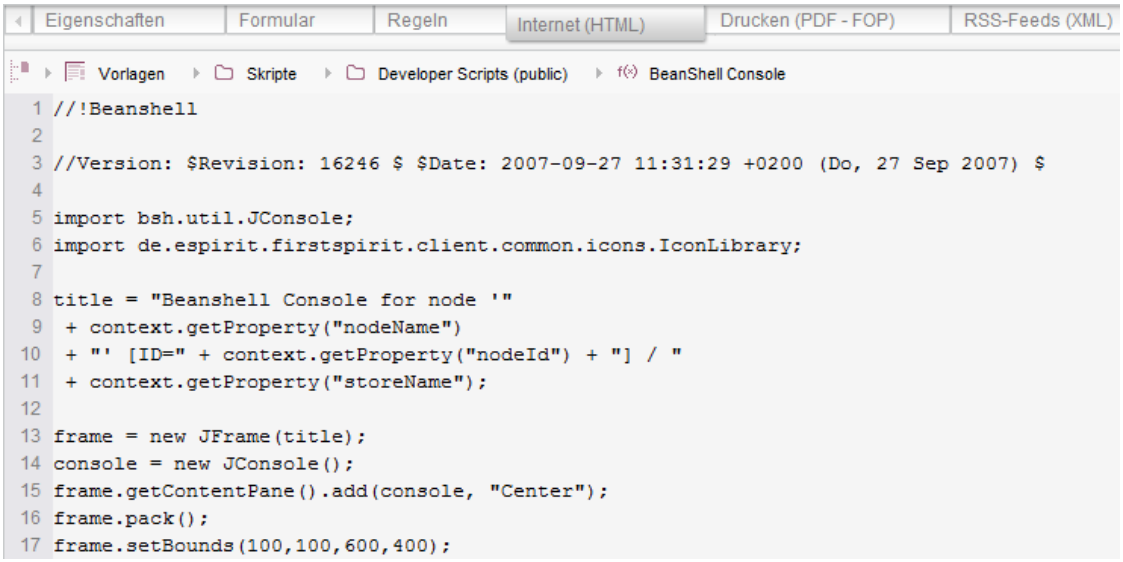


Abbildung 2-69: Ausgabekanäle "Skripte"

Der BeanShell-Quellcode wird in den Ausgabekanälen definiert, in denen das Skript ausgeführt werden soll. Durch die Angabe der Zeichenkette `//! Beanshell` in der ersten Zeile des Skripts, wird der nachfolgende Quelltext als BeanShell-Skript interpretiert.

Beispiele zur Skriptentwicklung innerhalb von Arbeitsabläufen siehe (siehe Kapitel 4.8 Seite 178).

Allgemeine Informationen zur Skriptentwicklung innerhalb von FirstSpirit siehe "FirstSpirit Online-Dokumentation".

Mit FirstSpiritVersion 5.0 werden als Lesehilfe an vielen Stellen der Vorlagen-Verwaltung Zeilennummern angezeigt. Mithilfe des Tastaturkürzels STRG + L können sie im JavaClient ein- und ausgeblendet werden.

2.12 Datenbank-Schemata

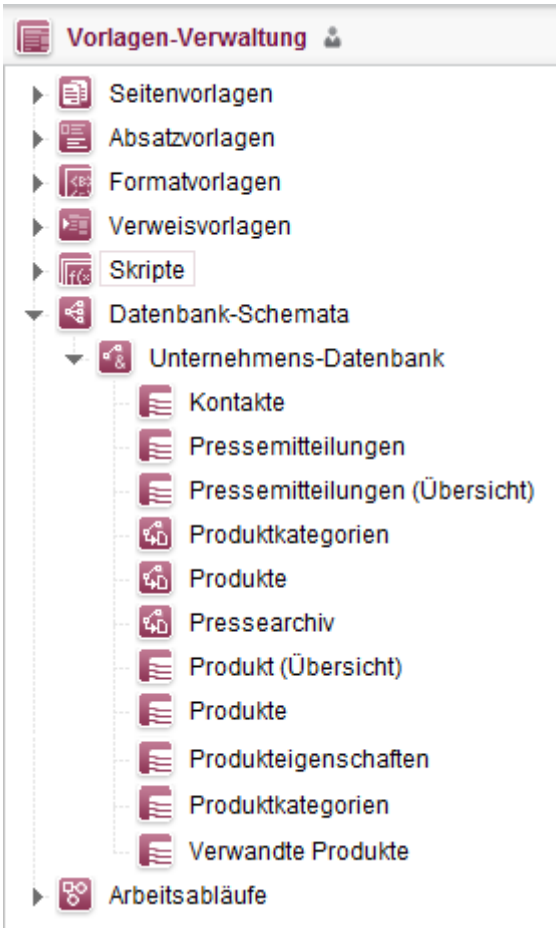


Abbildung 2-70: Baumansicht Vorlagen-Verwaltung – Datenbank-Schemata

FirstSpirit verfügt über leistungsfähige Mechanismen für die Anbindung von Datenbanken (siehe Kapitel 3 Seite 117).

In der Vorlagen-Verwaltung können über einen graphischen Schema-Editor Datenbank-Tabellen angelegt und modifiziert (siehe Kapitel 2.12.1 Seite 95), Vorlagen für die Pflege und Darstellung der Datensätze festgelegt (siehe Kapitel 2.12.4.1 Seite 109) und Abfragen zur Filterung der Datensätze formulieren werden (siehe Kapitel 2.12.5.1 Seite 112). Dazu wurde von FirstSpirit eine Datenbank-Abstraktionsschicht implementiert, die das universelle FirstSpirit-Content-Typsystem auf das konkret zu verwendende Datenbanksystem abbildet (siehe Kapitel 3 Seite 117).

2.12.1 Neu: Schema anlegen

In einem Datenbankschema wird festgelegt, welche Daten in einer Datenbank in welcher Form gespeichert werden und wie diese Daten in Beziehung zueinander stehen. Über den grafischen Editor im FirstSpirit JavaClient können Datenbanktabellen mit den zugehörigen Spalten sowie die Beziehungen zwischen den einzelnen Tabellen modelliert werden (siehe Kapitel 2.12.1 Seite 95).

Das Anlegen eines neuen Schemas im JavaClient erzeugt – neben dem Hinzufügen eines Schema-Knotens unterhalb des Wurzelknotens "Datenbank-Schemata" – eine neue Datenbank (siehe Abbildung 2-71 "Database_1") oder ein neues Datenbank-Schema (siehe Abbildung 2-71 "Schema") in der – für das betreffende Projekt konfigurierten – Datenbank. Ist beispielsweise vom Projektadministrator die "Standard-Datenbank" für das Projekt konfiguriert, wird über das Kontextmenü "Neu – Schema anlegen" eine neue Datenbank innerhalb der Standard-Datenbank (Derby) erzeugt. (Das Verhalten ist abhängig von der Konfiguration des Datenbank-Layers – siehe "FirstSpirit Handbuch für Administratoren" zur Einstellung "Kein Schema-Sync".) Über die zugehörigen Tabellen in der Datenquellen-Verwaltung von FirstSpirit erhält der Redakteur Zugriff auf die Datenbank und kann in den betreffenden Tabellen Inhalte einpflegen, die in die Datenbank geschrieben werden (sofern die Datenbank vom Projektadministrator nicht als "schreibgeschützt" definiert wurde) (siehe Kapitel 3 Seite 117).

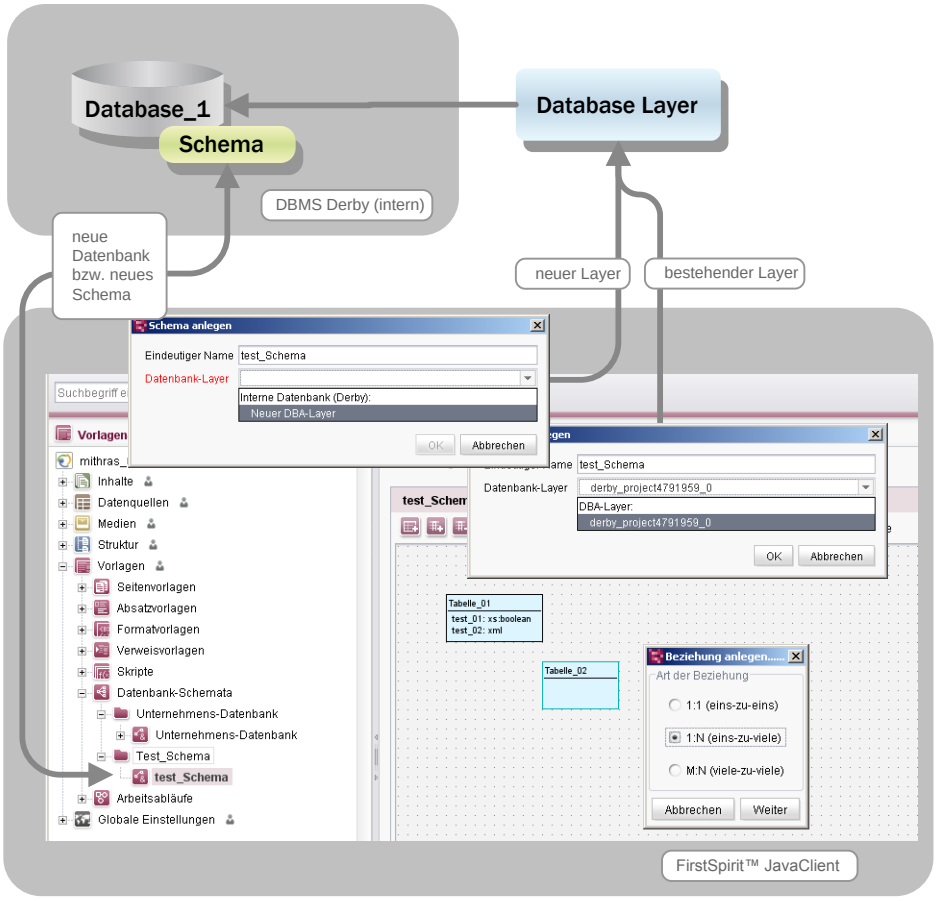


Abbildung 2-71: Anlegen eines neuen Schemas

Wird in FirstSpirit ein neues Schema für eine Datenquelle angelegt, so sollte im Vorfeld entschieden werden, in welcher Datenbank dieses im Produktivbetrieb abgelegt werden soll und welche Berechtigungen der von FirstSpirit verwendete DBMS Account im Produktivbetrieb haben soll. Eine spätere Umwandlung des Layer-Typen ist nicht ohne Weiteres möglich (vgl. "FirstSpirit Handbuch für Administratoren"). Im Zweifelsfall sollte für jedes FirstSpirit-Schema im Projekt ein eigener Standard-Layer angelegt werden (siehe Kapitel 3.2 Seite 119).

Generell empfehlen wir, die Entwicklung stets in einer dem Produktivbetrieb entsprechenden Umgebung vorzunehmen. Insbesondere das in FirstSpirit enthaltene Derby-DBMS ist nicht für den Produktivbetrieb geeignet und sollte daher lediglich für Tests verwendet werden.

Um ein neues Schema zu erstellen, wird neben den (sprachabhängigen) Anzeige- und dem Referenznamen die Auswahl eines Datenbank-Layers benötigt:

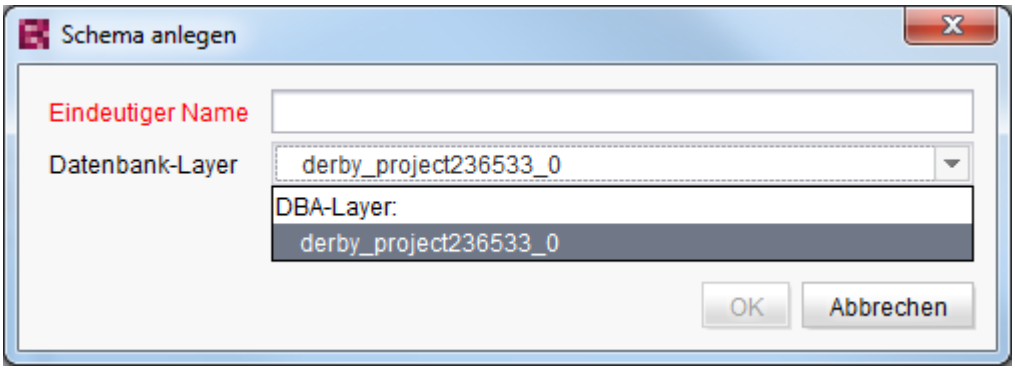


Abbildung 2-72: Schema anlegen

Eine ausführliche Beschreibung zu Referenz- und Anzeigenamen befindet sich in Kapitel 2.2.1 auf Seite 21.

Der Referenzname, der innerhalb eines FirstSpirit-Projekts für ein Schema definiert wird, entspricht nicht dem physikalischen Namen des Schemas in der Datenbank. Der physikalische Name wird datenbankabhängig automatisch vergeben – beispielsweise für die Standard-Datenbank (Derby) nach folgendem Muster: derby_projectID_schemaID

Datenbank-Layer: Im Feld "Datenbank-Layer" muss ein bestehender Datenbank-Layer zu einer Datenbank ausgewählt werden, in der die einzelnen Datenbanktabellen für dieses Schema gespeichert werden sollen. Dabei stehen so genannte "DBA-Layer" und (optional) "Standard-Layer" zur Auswahl:

- Standard-Layer: Soll direkt auf einem vorhandenen Datenbank-Schema gearbeitet werden, muss ein Standard-Layer ausgewählt werden (vgl. Abbildung 2-75). Dieser Layer kann in mehreren FirstSpirit-Projekten verwendet werden, die alle in die gleiche Datenbank schreiben bzw. Inhalte daraus lesen. Damit es bei der Verwendung eines Standard-Layers nicht zu Überschneidungen kommt, sollte jeweils nur eines der beteiligten Projekte das Schreibrecht auf der Datenbank besitzen (siehe Kapitel 3.2 Seite 119).
- DBA-Layer: Wird ein DBA-Layer ausgewählt, wird beim Anlegen ein eigenes Schema bzw. eine eigene Datenbank für das jeweilige Projekt erstellt (beim ersten Sync). Eine Überschneidung beim Schreiben der Inhalte ist hier nicht möglich (siehe Kapitel 3.3 Seite 120).

Bei Fragen zur Datenbank wenden sie sich bitte an den Projekt- oder Systemadministrator.

Ist für das Projekt noch *kein Datenbank-Layer* vorhanden, kann ein neuer Layer direkt über den Dialog "Schema anlegen" erzeugt werden. In der Klappliste "Datenbank-Layer" wird statt eines Datenbank-Layers (vgl. Abbildung 2-72) dann der Eintrag "Neuer DBA-Layer" angezeigt. Beim Bestätigen des Dialogs wird der neue Layer (siehe Abbildung 2-71) zusätzlich zu Schema bzw. Datenbank erzeugt.

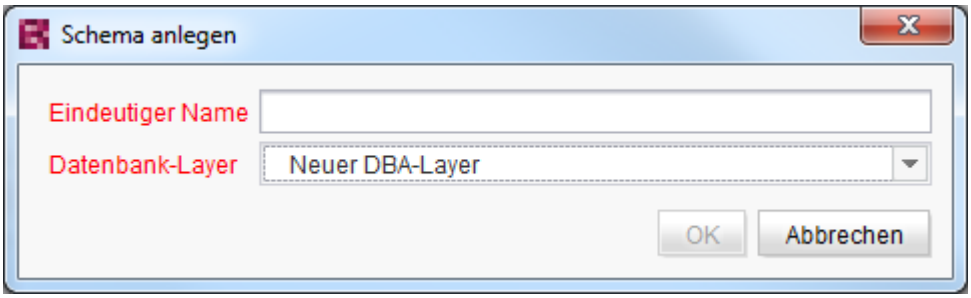


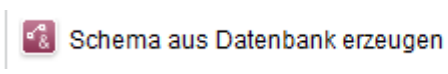
Abbildung 2-73: Schema anlegen – wenn kein Layer vorhanden ist

OK Mit einem Klick auf den Button wird das neue, leere Schema in die Baumansicht eingehängt und ein Schema bzw. eine Datenbank im konfigurierten DBMS erzeugt. Mithilfe des grafischen Editors kann das Schema weiter bearbeitet werden.

Abbrechen Mit einem Klick auf den Button wird der Vorgang abgebrochen. Ein neues Schema wird nicht angelegt.

Ein neues Schema kann auch über das Importieren einer Export-Datei aus einem anderen FirstSpirit-Projekt angelegt werden (siehe Kapitel 2.3.3.2 Seite 39).

2.12.2 Neu: Schema aus Datenbank erzeugen



Mithilfe dieser Funktion kann ein bereits vorhandenes Schema aus einer (externen) Datenbank in den neuen Schemaknoten übernommen werden.

Statt einen leeren Schemaknoten zu erzeugen (vgl. Kapitel 2.12.1), wird also auf Basis der bereits vorhandenen Tabellen und Beziehungen einer Datenbank ein

neuer Schemaknoten im FirstSpirit-Projekt angelegt. Das Anlegen des neuen Schemas aus einer Datenbank erfolgt über den Kontextmenüeintrag "Schema aus Datenbank erzeugen".

Die Struktur und die Inhalte einer externen Datenbank dürfen nicht verändert werden. Im Gegensatz zu internen Datenbanken, ist für externe Datenbanken nur ein lesender aber kein schreibender Zugriff möglich. Dazu müssen die Einschränkung "kein Schema Sync" und "Schreibgeschützt" vom Projektadministrator aktiviert werden. In diesem Fall können die Inhalte aus einem externen Schema ausgelesen und im FirstSpirit JavaClient als neuer Schemaknoten erzeugt werden. Über Tabellenvorlagen können die Inhalte anschließend in der Datenquellen-Verwaltung angezeigt (aber nicht verändert) werden.

Weiterführende Informationen siehe "FirstSpirit Handbuch für Administratoren".

Das Erzeugen eines neuen Schemas aus einer bestehenden Datenbank im JavaClient fügt einen neuen Schema-Knoten unterhalb des Wurzelknotens "Datenbank-Schemata" ein. Dabei wird versucht, die vorhandenen Tabellen und Inhalte aus einer bestehenden Datenbank bzw. aus einem bestehenden Datenbank-Schema, automatisch in den grafischen Schema-Editor zu übernehmen *(zu "Einschränkungen" siehe "FirstSpirit Handbuch für Administratoren")*. Ist beispielsweise vom Projektadministrator eine externe Oracle-Datenbank für das Projekt konfiguriert, wird über das Kontextmenü "Neu – Schema aus Datenbank erzeugen" ein Schema basierend auf der externen Datenbank erzeugt. Die zugehörigen Tabellen werden ebenfalls übernommen. Abhängig von der Datenbankkonfiguration erhält der Redakteur über die Datenquellen-Verwaltung lesenden Zugriff auf die Datenbank und kann Inhalte aus den betreffenden Tabellen auslesen und generieren.

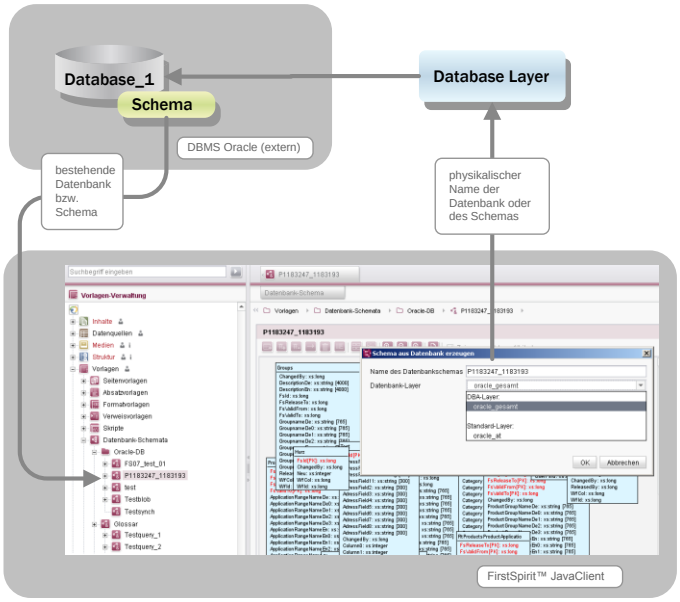


Abbildung 2-74: Erzeugen eines Schemas aus einer externen Datenbank

Um ein neues Schema zu erstellen, werden folgende Eingaben benötigt:

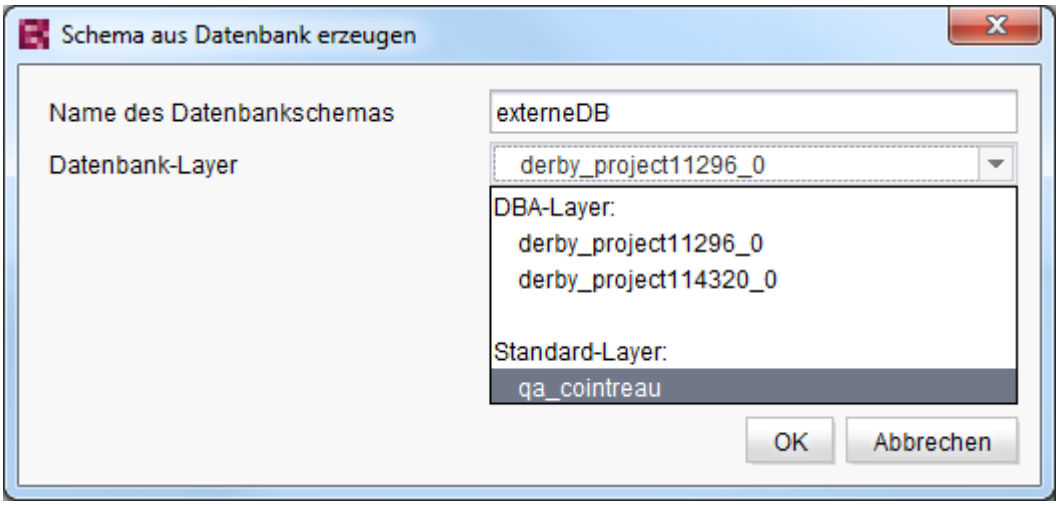


Abbildung 2-75: Schema aus Datenbank erzeugen

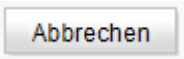
Name des Datenbankschemas: In diesem Feld muss der Name für das Datenbank-Schema angegeben werden. Im Gegensatz zum Anlegen eines neuen, leeren Schemas kann der Name *nicht* frei gewählt werden, er muss stattdessen genau dem physikalischen Namen des Datenbankschemas (bzw. der Datenbank) entsprechen.

Wird ein Name gewählt, der in der betreffenden Datenbank nicht vorhanden ist, dann wird ein leerer Schemaknoten angelegt. Es können in diesem Fall keine Inhalte (z. B. Tabellen) aus der gewählten Datenbank übernommen werden.

Datenbank-Layer: Im Feld "Datenbank-Layer" muss ein bestehender Datenbank-Layer ausgewählt werden (siehe Kapitel 2.12.1 Seite 95).

Die Auswahl einer externen Datenbank steht nur zur Verfügung, wenn der Projektadministrator in den Projekteigenschaften den Zugriff auf eine externe Datenbank für das Projekt konfiguriert hat.
Bei Fragen zur Datenbank, wenden sie sich bitte an den Projekt- oder Systemadministrator.

Mit einem Klick auf den Button werden das neue Schema und die zugehörigen Tabellen in die Baumansicht eingehängt und können mithilfe des grafischen Editors weiter bearbeitet werden.



Mit einem Klick auf den Button wird der Vorgang abgebrochen. Ein Schema wird nicht angelegt.

2.12.3 Der FirstSpirit-Schema-Editor

Für die Bearbeitung eines Schemas im FirstSpirit JavaClient steht auf der rechten Fensterseite ein grafischer Editor zur Verfügung, mit dessen Hilfe das gewünschte Datenbank-Schema erstellt werden kann. Abhängig von der Konfiguration kann ein Schema dabei auf bestehende Datenbankstrukturen zurückgreifen oder neue Tabellenstrukturen in einer bestehenden Datenbank anlegen.

Die Bedienung des Editors erfolgt entweder über die Symbolleiste des Editors oder über ein Kontextmenü, das an einer beliebigen Position des Editors über die rechte Maustaste aufgerufen werden kann.

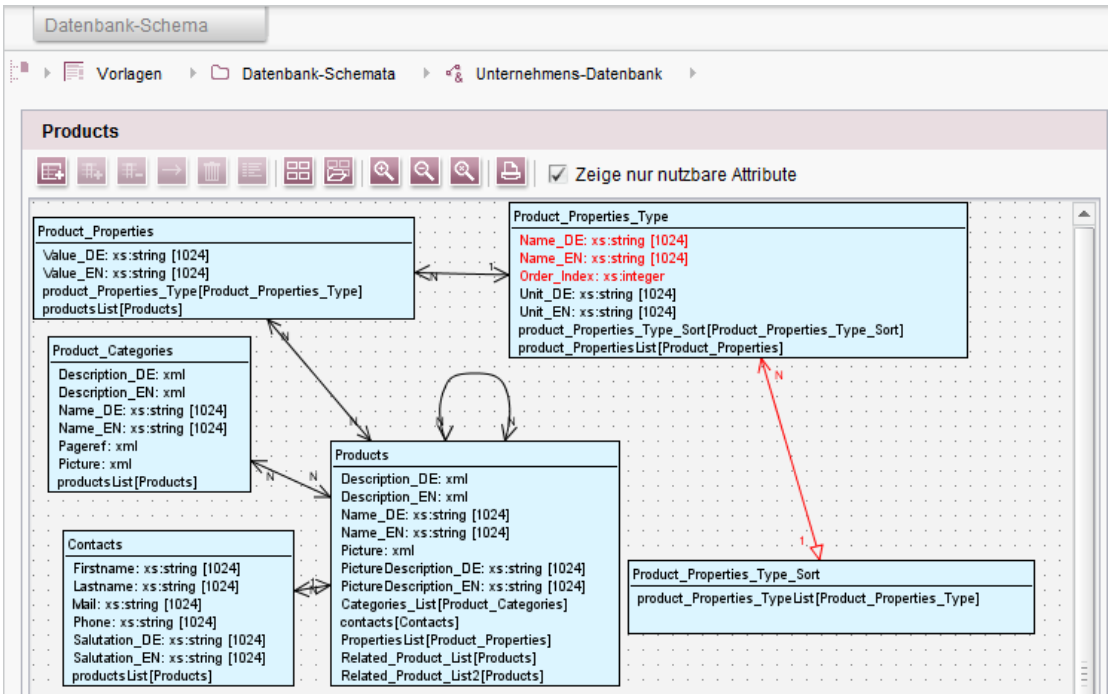


Abbildung 2-76: Datenbank-Schema-Editor

Abbildung 2-76 zeigt beispielhaft das Datenbank-Schema der Produktdatenbank des Testprojekts "Mithras Energy". Zu sehen sind unter anderem die Tabellen Produkte (*Products*), Produktkategorien (*Product_Categories*), Produkteigenschaften (*Product_Properties*) und Kontakte (*Contacts*). Dabei sind z. B. Produkte und Kontakte über eine 1:N-Beziehung, Produkte und Produktkategorien über eine M:N-Beziehung miteinander verknüpft.

 Tabelle anlegen, mithilfe dieser Schaltfläche kann eine neue Tabelle in das Datenbank-Schema eingefügt werden. Es öffnet sich folgendes Fenster:

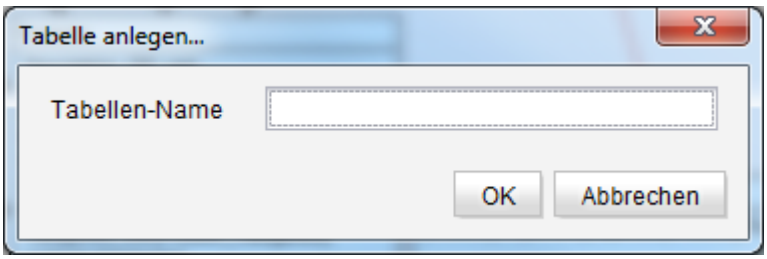


Abbildung 2-77: Tabelle anlegen

Tabellen-Name: In diesem Feld muss ein eindeutiger Name für die Datenbanktabelle angegeben werden.

 Spalte anlegen, mithilfe dieser Schaltfläche kann der aktivierten Tabelle eine neue Spalte hinzugefügt werden. Es öffnet sich folgendes Fenster:

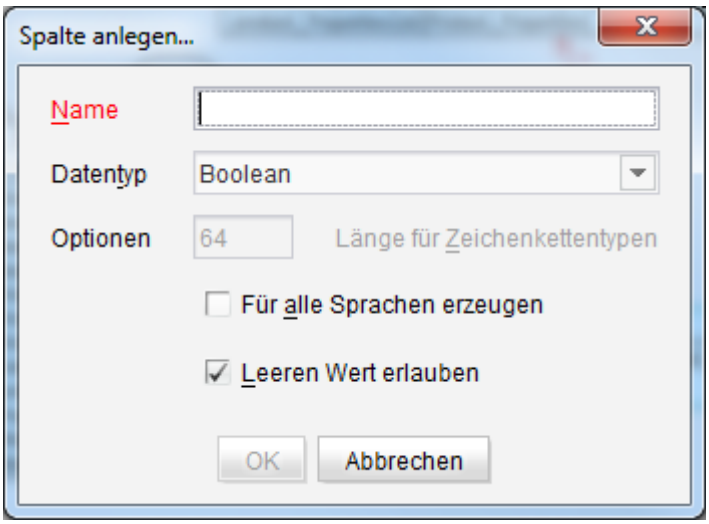


Abbildung 2-78: Spalte anlegen

Name: In diesem Feld muss der Spaltenname angegeben werden. Solange das Feld leer ist, wird *Name* in roter Schrift angezeigt und die neue Spalte kann nicht gespeichert werden.

Datentyp: Über diese Combobox kann der gewünschte Datentyp der neuen Tabellenspalte ausgewählt werden.

Boolean: Dieser Datentyp ermöglicht zwei Werte: `true` für "wahr" oder `false` für "falsch". Im Schema-Editor erhält dieser Datentyp ein `xs: boolean`.

Date: Dieser Datentyp wird für Datumswerte verwendet. Im Schema-Editor erhält dieser Datentyp ein `xs: date`.

Double: Dieser Datentyp ermöglicht die Eingabe von Gleitkommazahlen. Im Schema-Editor erhält dieser Datentyp ein `xs: decimal`.

FirstSpirit-Editor: Dieser Datentyp ermöglicht das Verwenden von DOM-Editoren. Die maximale Zeichenlänge ist dabei 65535. Im Schema-Editor erhält dieser Datentyp ein `xml`.

Integer: Dieser Datentyp wird für ganze Zahlen verwendet. Im Schema-Editor erhält dieser Datentyp ein `xs: integer`.

Long: Dieser Datentyp wird ebenfalls für ganze Zahlen verwendet, der Wertebereich ist aber größer als der des Datentyps Integer. Im Schema-Editor erhält dieser

Datentyp ein `xs:long`.

String: Dieser Datentyp wird für Zeichenketten verwendet. Im Schema-Editor erhält dieser Datentyp ein `xs: string`. Für diesen Datentyp kann zusätzlich die Zahl der maximal erlaubten Zeichen vorgegeben werden.

Optionen: In dem Feld muss für den Spaltentyp String die maximale Zeichenlänge angegeben werden. Der jeweilige Wert wird in eckigen Klammern hinter dem `xs: string` angezeigt.

Für alle Sprachen erzeugen: Über diese Option wird eine sprachabhängige Eingabe der Werte durch den Redakteur ermöglicht. Wird die Checkbox aktiviert, wird für jedes Attribut in jeder Sprache eine einzelne Spalte erzeugt. Dies ist dann sinnvoll, wenn die Attributbegriffe sich sprachlich unterscheiden. Die Spalten erhalten dabei für jede Sprache ein entsprechendes Sprachkürzel.

Leeren Wert erlauben: Durch Aktivierung dieser Option wird es dem Redakteur gestattet, einen neuen Datensatz anzulegen, ohne diese Spalte mit einem Wert zu belegen. Ist kein leerer Wert erlaubt (das bedeutet, es handelt sich um eine Pflichteingabe), wird der Spaltenname im Datenbank-Schema-Modell in roter Schrift angezeigt.

 Spalte entfernen, mithilfe dieser Funktion lassen sich die einzelnen Spalten einer Tabelle des Datenbank-Schemas entfernen. Die gewünschte Spalte kann aus der Combobox im folgenden Dialog ausgewählt werden:

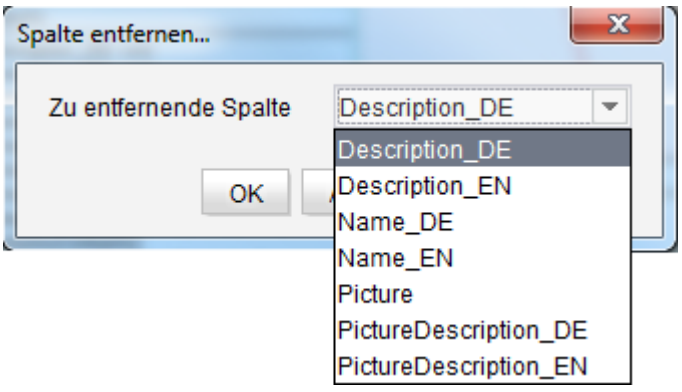


Abbildung 2-79: Spalte entfernen

 Fremdschlüssel-Beziehung anlegen, mithilfe dieser Schaltfläche kann eine Beziehung zwischen der aktivierten Tabelle und einer anderen Tabelle hergestellt werden.

Das Anlegen einer Beziehung soll am Beispiel der in Abbildung 2-76 gezeigten Produktdatenbank erläutert werden. Wir wollen die Beziehung zwischen den Tabellen Produkten und Kontakten herstellen: eine Kontaktperson kann dabei mehreren Produkten zugeordnet sein, sie stehen also in einer 1:N-Beziehung.

Zunächst müssen also die Tabelle Kontakte und zusätzlich (durch gedrückte Shift-Taste) die Tabelle Produkte aktiviert werden (aktivierte Tabellen ändern ihre Rahmenfarbe). Wird nun die Schaltfläche *Beziehung anlegen* ausgewählt, erscheint der erste Schritt des Vorgangs, bei dem die Art der Beziehung festgelegt werden muss. Die beiden Tabellen sollen in einer 1:N-Beziehung stehen.

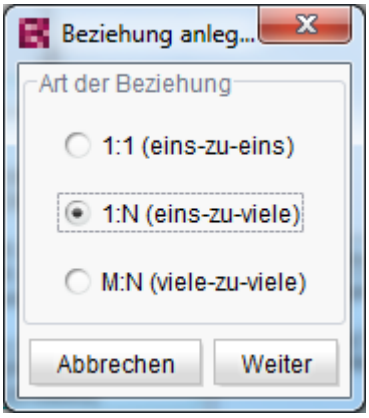


Abbildung 2-80: Beziehung anlegen – Schritt 1

Das zweite Fenster der Beziehung sieht folgendermaßen aus (je nach Auswahl im ersten Fenster, kann das Aussehen allerdings abweichen):

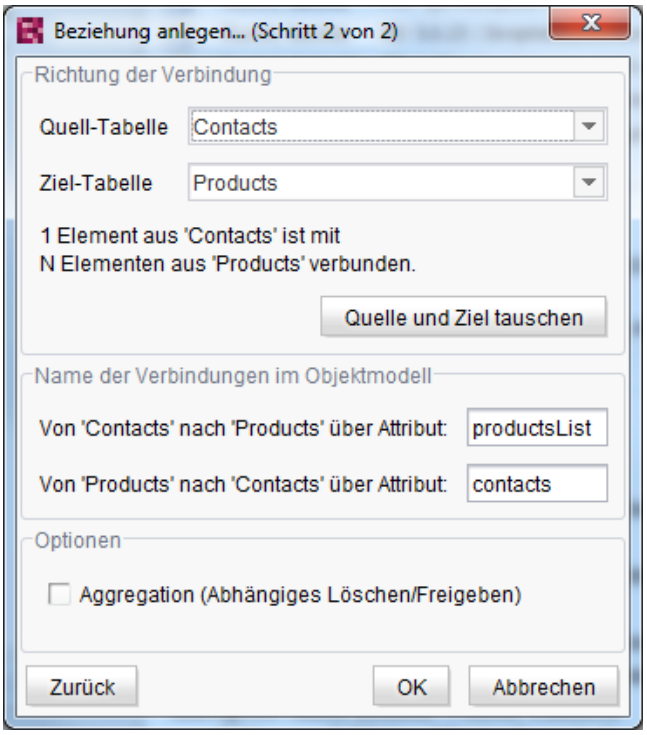


Abbildung 2-81: Beziehung anlegen – Schritt 2

Durch die Reihenfolge der Tabellen-Aktivierung ist bei der Richtung der Verbindung bereits vorbelegt, so dass 1 Element aus der Tabelle Kontakte mit N Elementen aus der Tabelle Produkte verbunden ist. Wurden die Tabellen versehentlich in der falschen Reihenfolge aktiviert, kann mithilfe des Buttons *Quelle und Ziel tauschen* die Reihenfolge umgedreht werden. Die weiteren Angaben in diesem Fenster können in der Regel so übernommen werden, wie vom System vorgeschlagen. Die Namen, die im Bereich "Namen der Verbindungen im Objektmodel" angegeben sind, werden bei der späteren Nutzung dazu verwendet, den Datenbeständen entlang ihrer Beziehungen zu folgen.

Elemente löschen, mithilfe dieser Schaltfläche kann die aktivierte Tabelle aus dem Datenbank-Schema gelöscht werden. Sie wird direkt, ohne Sicherheitsabfrage gelöscht, nicht gespeicherte Daten gehen dabei verloren und können nicht wiederhergestellt werden.

Eigenschaften, über diese Schaltfläche kann der Name der aktivierten Tabelle angezeigt werden.

Automatisch anordnen, durch Aktivieren dieser Schaltfläche werden die angezeigten Tabellen im Editor automatisch angeordnet.

 Gesicherte Anordnung laden, mithilfe dieser Schaltfläche können Änderungen bei der Anordnung der Tabellen des Schemas rückgängig gemacht werden. Es wird wieder die letzte gespeicherte Anordnung angezeigt.

 Ansicht vergrößern, mithilfe dieser Schaltfläche können die Elemente des Datenbank-Schemas vergrößert dargestellt werden.

 Ansicht verkleinern, mithilfe dieser Schaltfläche können die Elemente des Datenbank-Schemas verkleinert dargestellt werden.

 Normalansicht, mithilfe dieser Schaltfläche können die Elemente des Datenbank-Schemas wieder in Originalgröße dargestellt werden.

 Drucken, mithilfe dieser Schaltfläche kann das Datenbank-Schema ausgedruckt werden. Es öffnet sich zunächst folgendes Druckvorschau-Fenster:

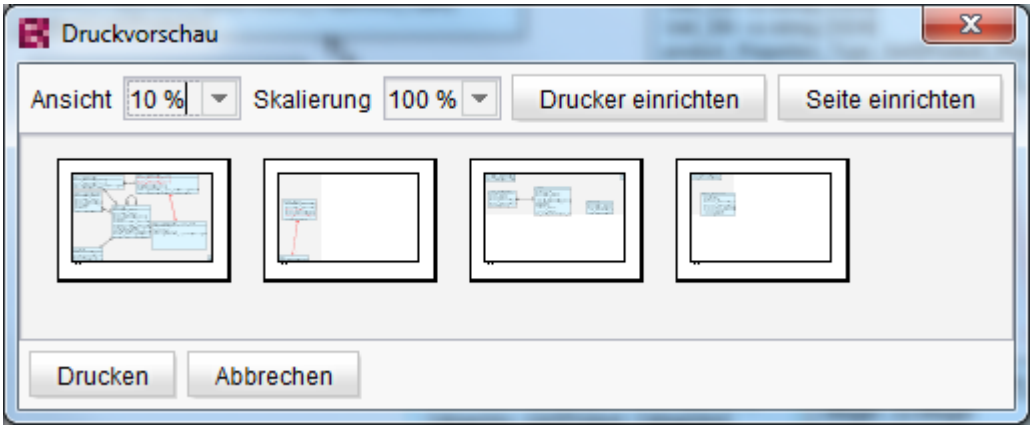


Abbildung 2-82: Druckvorschau

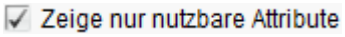
Ansicht: Über diese Combobox kann die Vorschau-Größe des Datenbank-Schemas beeinflusst werden. Mögliche Zoomstufen sind 10%, 25%, 50% und 100%.

Skalierung: Das Datenbank-Schema wird bei Bedarf verkleinert ausgedruckt. Möglich sind die Skalierungsstufen 10%, 25%, 50% und 100%.

Drucker einrichten: Öffnet den Dialog für die Drucker-Einstellungen.

Seite einrichten: Öffnet den Dialog für die Seiten-Einstellungen.

Über die Schaltfläche Drucken wird der Druckauftrag mit den aktuellen Einstellungen gestartet.



Zeige nur nutzbare Attribute, wird dieses Häkchen gesetzt, dann werden alle Attribute einer Tabelle ausgeblendet, die nicht vom Redakteur mit Inhalten gefüllt werden können.

Über das Kontextmenü können zusätzlich zu den Funktionen der Icons auch noch zwei weitere Funktionen aufgerufen werden:

Tabelle / Spalte umbenennen: Mithilfe dieser Funktion kann für eine bestehende Tabelle bzw. für eine bestehende Spalte einer Tabelle ein neuer Name vergeben werden. Es erscheint ein Fenster, in dem die gewünschte Tabelle bzw. Spalte ausgewählt und ein neuer Name vergeben werden kann. Bei der Umbenennung ist zu beachten, dass Tabellenvorlagen und Abfragen, die auf dieser Tabelle bzw. Spalte basieren, angepasst werden müssen.

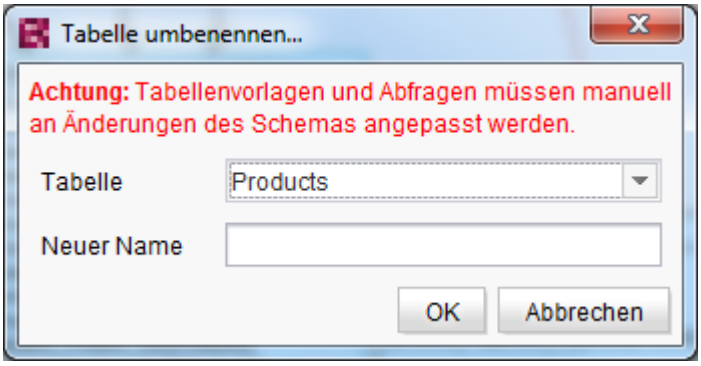


Abbildung 2-83: Tabelle umbenennen

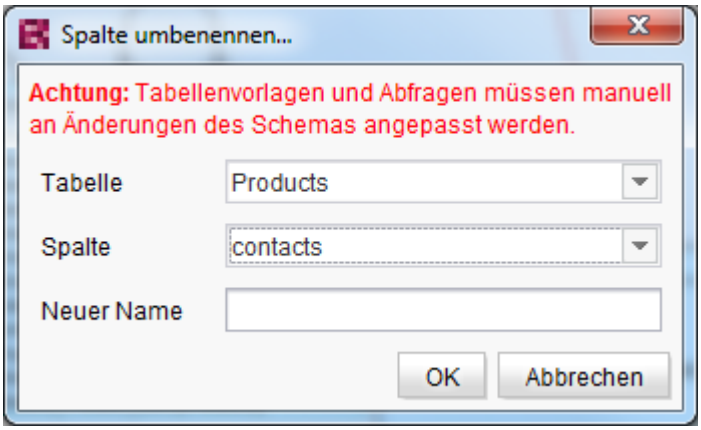


Abbildung 2-84: Spalte umbenennen

2.12.4 Tabellenvorlagen

Für jede eingepflegte Tabelle im Datenbankmodell muss unterhalb des Schemas eine Tabellenvorlage angelegt werden. In diesen Tabellenvorlagen wird festgelegt,

über welche Eingabekomponenten der Redakteur später die Daten in die entsprechenden Tabellen einpflegen kann.

2.12.4.1 Tabellenvorlagen – Register Vorschau, Eigenschaften und Formular

Die Register Vorschau, Eigenschaften und Formular für Tabellenvorlagen sind identisch zu den gleichnamigen Registern für Seitenvorlagen und können ebenso bearbeitet werden.

Informationen zu den Registern können Sie den Kapiteln 2.5.1 (Seite 54) bis Kapitel 2.5.3 (Seite 57) entnehmen.

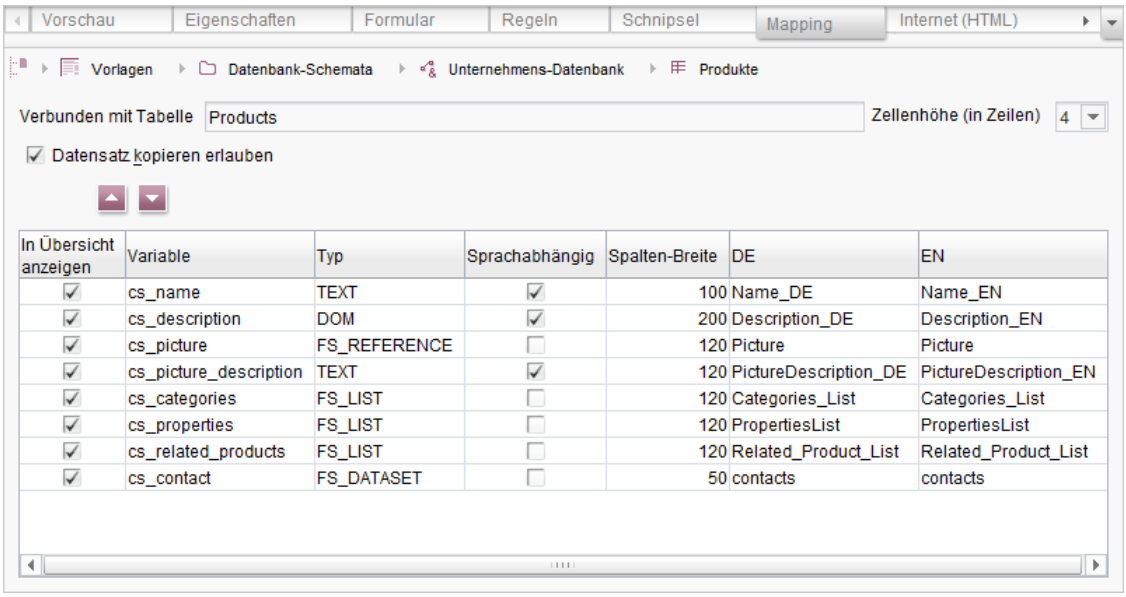
Für Tabellenvorlagen steht die Option "Vorlage in Auswahlliste verstecken" (vgl. Kapitel 2.5.2) nicht zur Verfügung.

Wichtig für die Verwendung im WebClient

Soll eine Tabellenvorlage im WebClient verwendet werden können, muss die Checkbox "in WebEdit verwendbar“ auf dem Register "Eigenschaften“ aktiviert werden. Darüber hinaus muss für eine korrekte Darstellung der Datensätze im WebClient eine geeignete Vorschauseite (vgl. Kapitel 2.5.2) auf dem Register "Eigenschaften" eingestellt werden.

2.12.4.2 Tabellenvorlagen – Register Mapping

In diesem Bereich wird festgelegt, über welche Eingabekomponenten die Datensätze in die Datenbanktabellen eingefügt werden. Jeder (im Register Formular) definierten Eingabekomponente wird dabei eine Tabellenspalte zugeordnet.



In Übersicht anzeigen	Variable	Typ	Sprachabhängig	Spalten-Breite	DE	EN
☑	cs_name	TEXT	☑	100	Name_DE	Name_EN
☑	cs_description	DOM	☑	200	Description_DE	Description_EN
☑	cs_picture	FS_REFERENCE	☐	120	Picture	Picture
☑	cs_picture_description	TEXT	☑	120	PictureDescription_DE	PictureDescription_EN
☑	cs_categories	FS_LIST	☐	120	Categories_List	Categories_List
☑	cs_properties	FS_LIST	☐	120	PropertiesList	PropertiesList
☑	cs_related_products	FS_LIST	☐	120	Related_Product_List	Related_Product_List
☑	cs_contact	FS_DATASET	☐	50	contacts	contacts

Abbildung 2-85: Tabellenvorlage – Register "Mapping"

Verbunden mit Tabelle: In diesem Feld wird die Tabelle angezeigt, für die die Mapping-Einstellungen gelten.

Zellenhöhe (in Zeilen): Datensätze können später in der Datenquelle (siehe *FirstSpirit Handbuch für Redakteure*, Kapitel 5) mehrzeilig dargestellt werden. Über diese Combobox kann eingestellt werden, wie viele Zeilen eine Zelle bzw. ein Datensatz in der Datenübersicht haben soll (maximal 10). Auf diese Weise können beispielsweise auch Thumbnails von Bildern in der Übersicht angezeigt werden.

Datensatz kopieren erlauben: Ist diese Checkbox aktiviert (Standardeinstellung), können bestehende Datensätze in der zugehörigen Datenquelle durch den Redakteur kopiert werden. Wird die Checkbox deaktiviert, können nur "leere" neue Datensätze angelegt werden, das Icon ist deaktiv.

Jede Zeile der Liste entspricht einer Spalte in der Datenübersicht der zugehörigen Datenquelle. Mit einem Klick auf die Icons wird eine markierte Zeile um eine Position nach oben bzw. unten verschoben, die zugehörige Spalte in der Datenübersicht nach links bzw. rechts. Auf diese Weise können wichtigere Spalten weiter nach links verschoben werden. Die Reihenfolge kann durch den Redakteur manuell verändert werden, bei einer Aktualisierung der Ansicht wird die Reihenfolge jedoch wieder auf die auf diesem Register vorgenommene Einstellung zurückgesetzt. In der Datenerfassung wirkt sich die hier gewählte Reihenfolge dagegen nicht aus.

In Übersicht anzeigen: Über diese Spalte können durch Deaktivieren der Checkboxen Tabellenspalten aus der Datenübersicht in der jeweiligen Datenquelle ausgeblendet werden, um z. B. bei vielen Spalten die Übersicht zu verbessern. In der Datenerfassung wirkt sich das Ausblenden dagegen nicht aus.

Variable: In dieser Spalte steht der Name der Variablen, wie er im Formular der Tabellenvorlage (Kapitel 2.12.4.1 Seite 109) definiert wurde.

Typ: In dieser Spalte ist der Typ der Eingabekomponente für die jeweilige Variable angegeben.

Sprachabhängig: Ist die Eingabekomponente im Register Formular mehrsprachig definiert, dann wird dies durch ein Häkchen in dieser Spalte angezeigt.

Spalten-Breite: In diesem Feld wird die Breite der Spalte in Pixels angegeben, wie sie später in der Datenquellen-Verwaltung dargestellt wird.

Sprache (DE/EN): In diesem Feld wird die Tabellenspalte ausgewählt, an die der Inhalt des Eingabeelementes übergeben werden soll. Für jede Projektsprache gibt es eine eigene Spalte. Handelt es sich um eine sprachunabhängige Eingabekomponente, dann ist hier für jede Sprache dieselbe Tabellenspalte auszuwählen. Bei sprachabhängigen Eingabekomponenten muss für jede Sprache eine eigene Tabellenspalte existieren, in welche der Wert übernommen wird.

2.12.4.3 Tabellenvorlagen – Register Vorlagensätze

Über die Register wird festgelegt, welches Aussehen die einzelnen Datensätze später auf der Webseite bzw. in anderen Ausgabekanälen annehmen sollen, die mithilfe dieser Tabellenvorlage eingepflegt werden.

Das Register für Tabellenvorlagen ist identisch zum gleichnamigen Register für Seitenvorlagen und kann ebenso bearbeitet werden.

Informationen zum Register "Vorlagensätze" siehe Kapitel 2.5.4 Seite 58.

2.12.5 Abfragen

Um die Anzahl der Datensätze für die spätere Ausgabe einzuschränken, können für jedes Datenbankschema mehrere Abfragen erstellt werden. In diesen Abfragen wird festgelegt, welche Bedingungen ein Datensatz erfüllen muss, um in die Ergebnisliste aufgenommen zu werden.

2.12.5.1 Abfrage – Register Bedingungen

Im Register "Bedingungen" können über einen grafischen Editor, im sogenannten "Wizard-Mode", die gewünschten Filterkriterien für eine Abfrage definiert werden. Dabei können mehrere Regeln festgelegt werden, die sich anschließend auf die Anzeige der passenden Datensätze im Register "Ergebnis" auswirken.

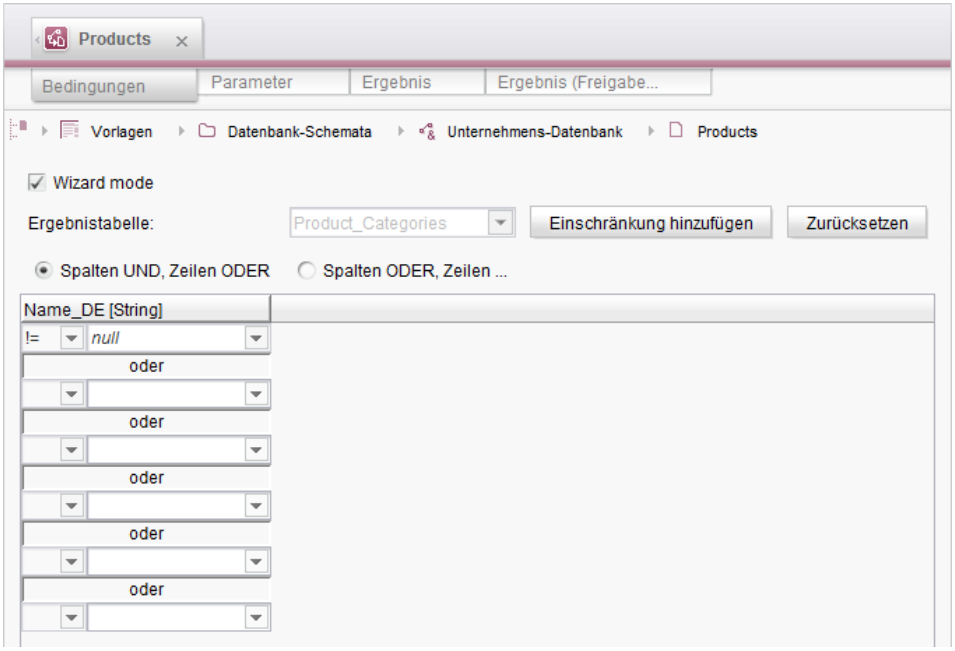


Abbildung 2-86: Abfrage – Register "Bedingungen" (Wizard-Mode) (neues Look&Feel)

Wizard mode: Wird diese Option deaktiviert, dann wird der Quellcode der ausgewählten Abfrage in einem Editor angezeigt und kann bei Bedarf noch modifiziert werden. Eine Abfrage kann auch direkt über diesen Editor programmiert werden.

Tags und Parameter, die zur direkten Programmierung von Abfragen verwendet werden können, können in der FirstSpirit Online Dokumentation nachgeschlagen werden: Absatz "Abfrageteil (QUERY)" im Kapitel "Funktion contentSelect" ("Vorlagenentwicklung" / "Vorlagensyntax" / "Funktionen" / "im Header" / "contentSelect").

Abbildung 2-87: Register "Bedingungen" (kein Wizard-Mode)

Werden Änderungen an der Abfrage vorgenommen, die im Wizard mode nicht abgebildet werden können, dann wird die Abfrage (im Editor) automatisch angepasst, sobald der Wizard mode wieder aktiviert wird.

Ergebnistabelle: Hier kann eine Tabelle aus dem FirstSpirit-Schema ausgewählt werden, für die Einschränkungen bei der Ausgabe vorgenommen werden sollen. Ist diese Auswahl einmal getroffen, wird dieses Feld deaktiviert. Die Auswahl kann nur über das "Zurücksetzen" der gesamten Abfrage entfernt werden (s.u.).

Einschränkung hinzufügen: Durch Aktivierung dieses Buttons wird eine neue Bedingung hinzugefügt. Es öffnet sich ein Fenster, in dem eine bestimmte Spalte der ausgewählten Ergebnistabelle als neue Referenz ausgewählt werden kann. Nur für diese Referenz kann anschließend die einschränkende Bedingung festgelegt werden. In der Bedingungsspalte im unteren Fensterbereich können die konkreten Werte angegeben werden, die erfüllt sein müssen. Dafür wird im linken Feld der gewünschte Vergleichsoperator für die Bedingung ausgewählt. Im rechten Feld kann entweder ein konkreter Vergleichswert eingetragen oder ein Parameter-Bezeichner für den Vergleichswert angegeben werden. Dieser Parameter wird dann bei jeder Ausführung abgefragt, so dass der konkrete Vergleichswert erst bei der Ausführung bestimmt werden muss.

Zurücksetzen: Alle bisher definierten Bedingungen und Abfrageergebnisse werden entfernt. Es kann wieder eine Ergebnistabelle ausgewählt werden. Damit nicht versehentlich Daten gelöscht werden, erfolgt vor dem Zurücksetzen eine Sicherheitsabfrage.

Spalten UND, Zeilen ODER: Wird diese Option ausgewählt, dann wird immer die Schnittmenge aller Spaltenergebnisse ausgegeben. Die einzelnen Zeilen einer Spaltenbedingung werden hierbei durch eine ODER-Verknüpfung verbunden.

Spalten ODER, Zeilen UND: Wird diese Option ausgewählt, dann werden die zusammengefassten Ergebnisse aller Spalten ausgegeben, doppelte Datensätze werden dabei übergangen. Die einzelnen Zeilen einer Spaltenbedingung werden

hierbei durch eine UND-Verknüpfung verbunden.

2.12.5.2 Abfrage – Register Parameter

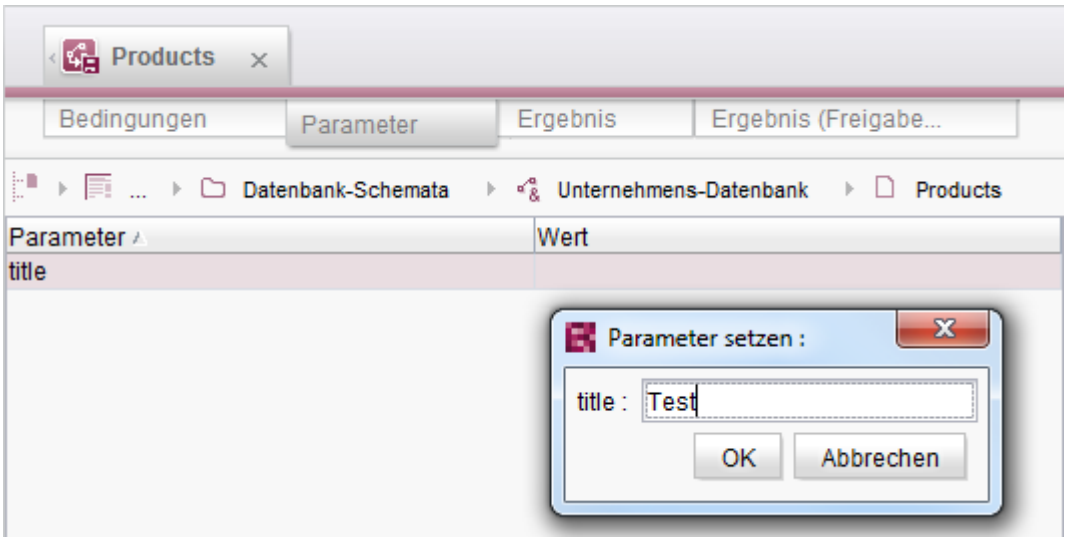
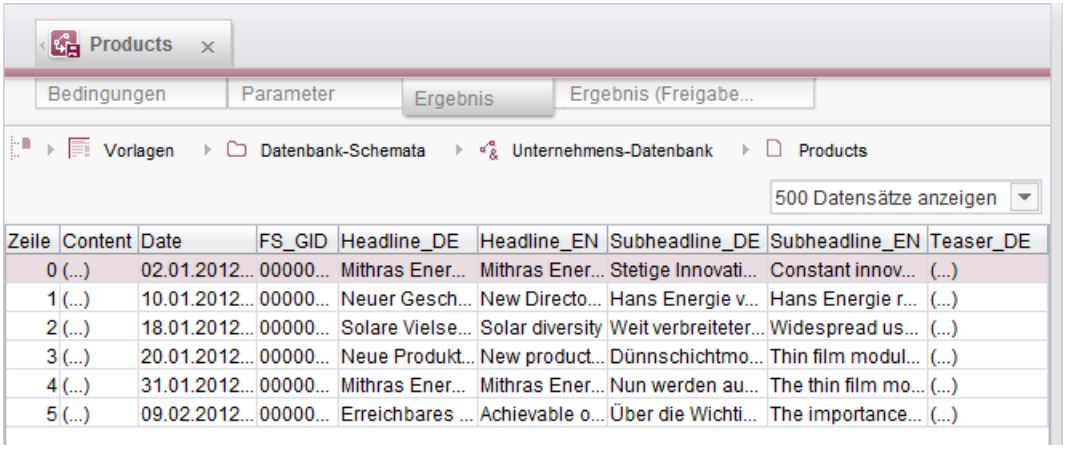


Abbildung 2-88: Abfrage – Register "Parameter" (neues Look&Feel)

In diesem Bereich sind alle Parameter aufgelistet, die in dieser Abfrage verwendet werden. Bei Bedarf können in der Spalte **Wert** die Parameter bereits gesetzt werden. D.h. die entsprechenden Abfrageparameter werden mit Werten belegt. Bei jeder Ausführung werden dann diese Werte für die Abfrage verwendet.

2.12.5.3 Abfrage – Register Ergebnis

In diesem Bereich werden die Ergebnis-Datensätze ausgegeben, die sich durch die Bedingungen in der Abfrage und der Wertebelegung der Abfrageparameter ergeben.



Zeile	Content	Date	FS_GID	Headline_DE	Headline_EN	Subheadline_DE	Subheadline_EN	Teaser_DE
0	(...)	02.01.2012...	00000...	Mithras Ener...	Mithras Ener...	Stetige Innovati...	Constant innov...	(...)
1	(...)	10.01.2012...	00000...	Neuer Gesch...	New Directo...	Hans Energie v...	Hans Energie r...	(...)
2	(...)	18.01.2012...	00000...	Solare Vielse...	Solar diversity	Weit verbreiteter...	Widespread us...	(...)
3	(...)	20.01.2012...	00000...	Neue Produkt...	New product...	Dünnschichtmo...	Thin film modul...	(...)
4	(...)	31.01.2012...	00000...	Mithras Ener...	Mithras Ener...	Nun werden au...	The thin film mo...	(...)
5	(...)	09.02.2012...	00000...	Erreichbares ...	Achievable o...	Über die Wichti...	The importance...	(...)

Abbildung 2-89: Abfrage – Register Ergebnis

2.12.5.4 Abfrage – Register Ergebnis (Freigabe)

In diesem Bereich werden die Ergebnis-Datensätze ausgegeben, die sich durch die Bedingungen und die Abfrageparameter in der Abfrage ergeben und die sich **im Freigabestand** befinden. Die Ergebnismenge kann sich somit von der Liste aus dem Ergebnis-Register unterscheiden.

2.13 Arbeitsabläufe

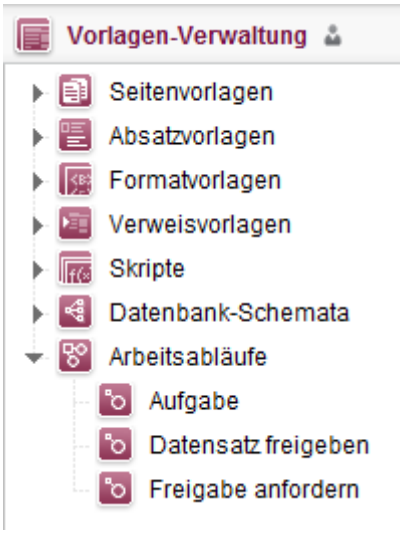


Abbildung 2-90: Baumansicht Vorlagen-Verwaltung – Arbeitsabläufe

Ein Arbeitsablauf ist eine Abfolge von Aufgaben, die nach einer fest vorgegeben Struktur abgearbeitet werden. Für die jeweiligen Aufgaben können sowohl Fälligkeitszeitpunkte als auch berechtigte Personengruppen festgelegt werden. In FirstSpirit integrierte Arbeitsabläufe sind die Aufgabenerteilung und die Freigabeanforderung.

Zur Modellierung, Konfiguration und Ausführung von Arbeitsabläufen siehe Kapitel 4 Seite 123 ff.

3 Datenquellen in FirstSpirit

FirstSpirit verfügt über leistungsfähige Mechanismen für die Anbindung von Datenbanken. Innerhalb der Redaktionsumgebung werden die angebundenen Datenbanken als Datenquellen bezeichnet. Die in den Datenquellen verwalteten Datensätze können in die Webseiten eingebunden und nahtlos in FirstSpirit bearbeitet werden, ohne die Redaktionsumgebung zu verlassen (siehe Kapitel 3.4 Seite 121).

Die Datenbankanbindung ist für eine große Zahl von Datenbanken verfügbar und wird über die von den Datenbankherstellern bereitgestellten JDBC-Treiber ausgeführt. Jeder Datenbankhersteller implementiert eine eigene interne Struktur um die im Datenbank-Server (DBMS) gespeicherten Daten zu verwalten. Diese internen Strukturen in Verbindung mit den Sicherheits- und Wartungsvorgaben des firmeninternen Betriebs haben Implikationen auf die Form und Konfiguration der Anbindung der Datenbanken an FirstSpirit.

Weiterführende Informationen zur Anbindung und Konfiguration von Datenbanken an FirstSpirit siehe "FirstSpirit Handbuch für Administratoren", Kapitel 4.8 "Datenbankanbindung".

Die folgenden Kapitel sollen den Vorlagenentwickler bei der Wahl der korrekten Anbindung unterstützen und die Konzepte zum Arbeiten mit Datenquellen im FirstSpirit JavaClient erläutern:

Kapitel 3.1 definiert die verwendeten Begriffe, da diese im Bereich von Datenbanken je nach verwendetem Kontext mit unterschiedlichen Bedeutungen belegt sein können.

Die Kapitel 3.2 und 3.3 behandelt die in FirstSpirit verwendbaren Layer-Typen für die Datenbankanbindung. Die Wahl des Layer-Typs hat diverse Auswirkungen auf den späteren Betrieb, ist nicht ohne weiteres änderbar und sollte daher wohl überlegt sein.

In Kapitel 3.4 wird das Konzept der Datenquellen im FirstSpirit JavaClient beschrieben.

3.1 Begriffe

Im Umgang mit der Anbindung von Datenquellen an FirstSpirit entstehen viele Missverständnisse aufgrund einer Vielzahl von, zum Teil auch mehrdeutig verwendeten, Begriffen. Die Hersteller der anzubindenden Datenbanken pflegen meist eigene Begriffswelten, die sich mit der von FirstSpirit überschneiden. Daher soll hier zunächst eine Klärung der in diesem Dokument verwendeten Begriffe vorgenommen werden:

Layer: Dieser Begriff bezeichnet die Verbindungskonfiguration in FirstSpirit zu einem Datenbank Management Systemen (DBMS). Ein Layer kann in FirstSpirit mehreren FirstSpirit-Schema*ta* (s.u.) zugeordnet sein.

- Standard-Layer: Dieser Layer-Typ enthält eine explizite Definition des genutzten DB-Schemas in der Layer-Definition. In diesem Fall werden alle Tabellen der FirstSpirit-Schemata, die diesen Layer nutzen, in das angegebene DB-Schema gespeichert. In einem FirstSpirit-Projekt, dem ausschließlich Standard-Layer zugeordnet sind, kann ein FirstSpirit-Benutzer keine neuen zusätzlichen Schemata anlegen. Nur der FirstSpirit-Administrator kann dem Projekt weitere Standard-Layer hinzufügen. Ein Standard-Layer sollte immer nur genau einem FirstSpirit-Schema zugewiesen werden. (siehe: Kapitel 3.2 Seite 119)
- DBA-Layer: Dieser Layer-Typ enthält keine explizite Definition des zu nutzenden DB-Schemas. FirstSpirit legt automatisch für jedes FirstSpirit-Schema ein eigenes DB-Schema an. Damit wird das Anlegen weiterer Schemata auch FirstSpirit-Benutzern ermöglicht. Bei den meisten DBMS werden dazu jedoch umfassende DBA-Rechte (DBA = Database Administrator) benötigt. (siehe: Kapitel 3.3 Seite 120)

FirstSpirit-Schema: Dieser Begriff beschreibt die in FirstSpirit beschriebenen Strukturen und Vorlagen von Datenquellen. FirstSpirit-Schemata enthalten somit sowohl Tabellen und deren Fremdschlüsselbeziehungen als auch die Vorlagen für die Generierung. Die Tabellenstruktur und die Datensätze der FirstSpirit-Schemata werden in einem DBMS innerhalb eines *DB-Schemas* (s.u.) hinterlegt. Jedes FirstSpirit-Schema ist immer genau einem Layer zugeordnet (siehe Kapitel 3.2 Seite 119 und Kapitel 3.3 Seite 120).

DB-Schema: Dieser Begriff beschreibt den logischen Bereich innerhalb der Datenbank in dem die Tabellen abgelegt werden ("Tablespace"). Jede Tabelle innerhalb dieses Bereichs muss einen eindeutigen Namen besitzen. Als technischer Begriff wird in DBMS häufig auch der Begriff "Database" verwendet. In der Layer-Konfiguration nennt FirstSpirit diese einfach "Schema".

3.2 Standard-Layer

Der Standard-Layer (siehe auch: Kapitel 3.1 Seite 118) fand unter der Bezeichnung „MultiProjektLayer“ bereits in früheren FirstSpirit-Versionen Verwendung. Der Mechanismus zur Vermeidung von Konflikten bei gleichen Tabellennamen in unterschiedlichen FirstSpirit-Schemata ist im Standard-Layer jedoch nicht mehr vorhanden. Als Ersatz für die flexible Zuordnung von FirstSpirit-Schemata zu DB-Schemata wurde der DBA-Layer eingeführt (vgl. Kapitel 3.3).

Wird der Standard-Layer mehreren FirstSpirit-Schemata zugeordnet, so tritt bei gleichen Tabellennamen innerhalb der FirstSpirit-Schemata ein Konflikt auf, da diese der gleichen Tabelle im DB-Schema zugeordnet werden (vgl. Abbildung 3-1).

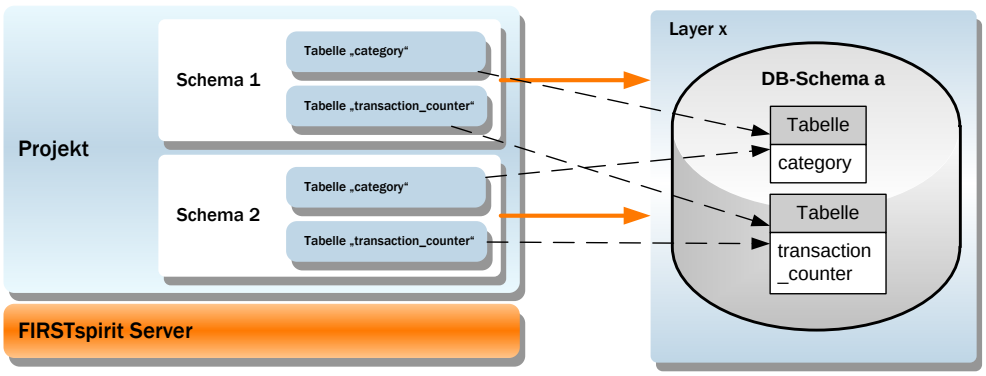


Abbildung 3-1: Problematische Nutzung eines Standard-Layers

Ein Sonderfall ist dabei die Systemtabelle "transaction_counter", die für jedes FirstSpirit-Schema versteckt angelegt wird. FirstSpirit versucht hier den o.g. Konflikt aufzulösen, indem es die Tabellen in eine Tabelle überführt.

*In jedem Fall wird davon abgeraten, zwei FirstSpirit-Schemata in einem DB-Schema zu mischen. Standard-Layer sollten **immer** nur einem FirstSpirit-Schema zugeordnet sein.*

Die korrekte Verwendung von Standard-Layern zeigt Abbildung 3-2. Für jedes FirstSpirit-Schema wird ein eigener Standard-Layer angelegt und somit ein eigenes DB-Schema, in dem die zugehörigen Tabellen abgelegt werden.

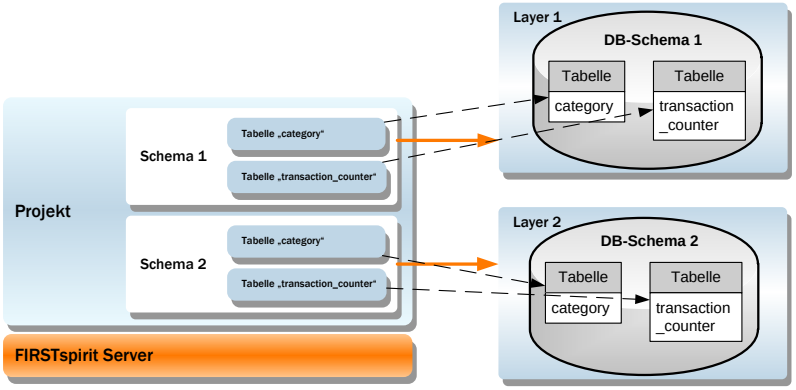


Abbildung 3-2: Korrekte Verwendung von getrennten Standard-Layern

3.3 DBA-Layer

Der DBA-Layer fand unter der Bezeichnung „SingleProjectLayer“ bereits in früheren FirstSpirit-Versionen Verwendung. Er wurde eingeführt, um in einem Projekt auch ohne das Zutun eines Datenbank-Administrators FirstSpirit-Schemata anlegen zu können.

Beim DBA-Layer wird im Gegensatz zum Standard-Layer kein explizites DB-Schema für die Speicherung der Tabellen in der Layer-Definition angegeben. FirstSpirit erstellt die zu den FirstSpirit-Schemata gehörenden DB-Schemata selbstständig im DBMS. Der Name der DB-Schemata setzt sich dabei aus Schema- und Projekt-ID zusammen (vgl. Kapitel 2.12.1 Seite 95).

Für die Nutzung eines DBA-Layers benötigt der im Layer angegebene Benutzer die Rechte DB-Schemata im DBMS anzulegen. Dies ist in vielen DBMS nur mit Rechten ähnlich eines Datenbank-Administrators möglich.

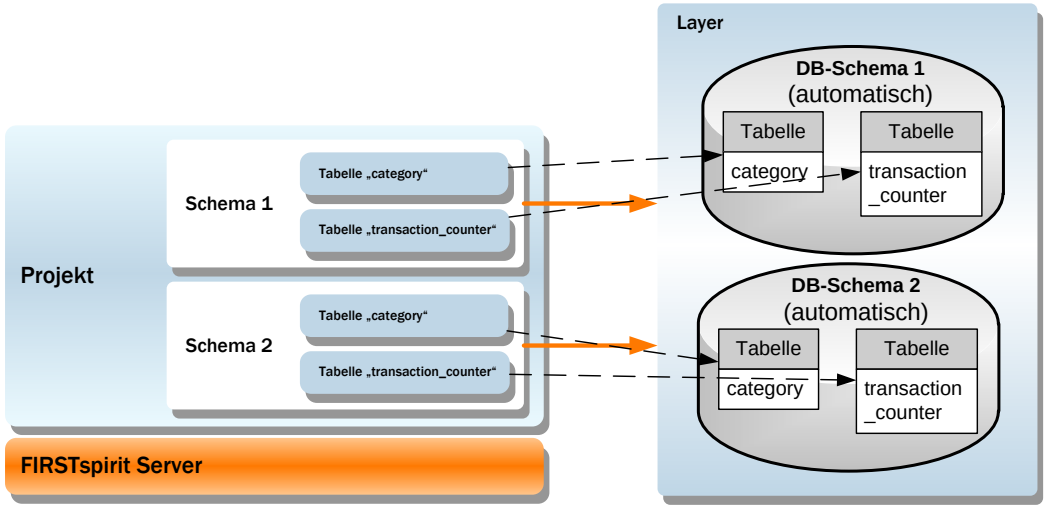


Abbildung 3-3: DBA-Layer

3.4 Datenquellen im FirstSpirit JavaClient

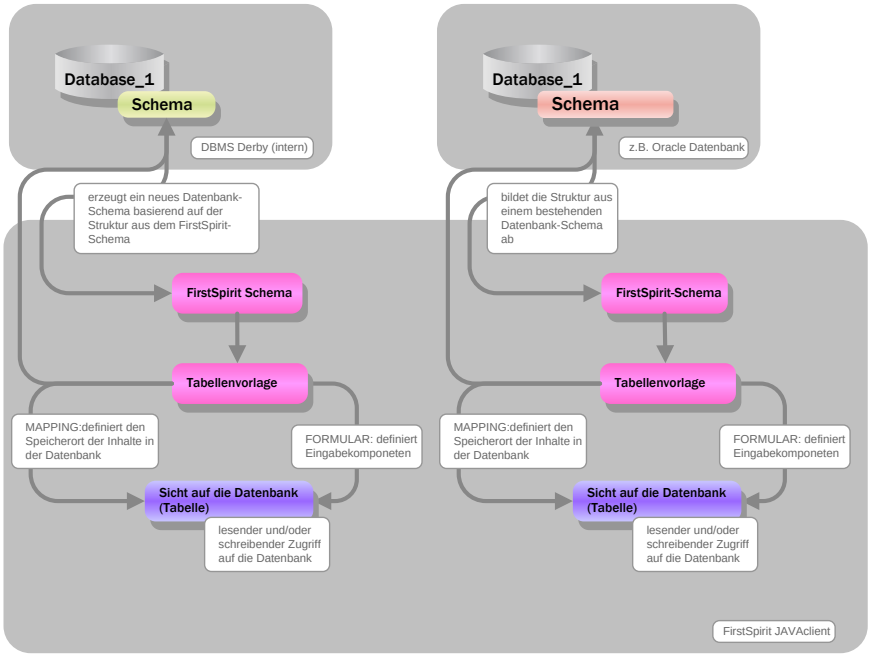


Abbildung 3-4: Konzept – Schemata, Tabellenvorlagen, Sichten auf Datenbanken

FirstSpirit-Schema: Über den FirstSpirit JavaClient kann entweder ein neues, leeres Datenbank-Schema (siehe Kapitel 2.12.1 Seite 95) oder ein Datenbank-Schema aus einer bestehenden Datenbank erzeugt werden (siehe Kapitel 2.12.2 Seite 98).

Nachdem ein neues Schema angelegt wurde, können mithilfe des grafischen Editors im FirstSpirit JavaClient die benötigten Tabellen in der ausgewählten Datenbank angelegt und miteinander in Beziehung gesetzt werden (siehe Kapitel 2.12.1 Seite 95). Für jede Tabelle müssen die Spalten angegeben werden, die später vom Redakteur eingepflegt werden sollen. Eine Spalte mit dem notwendigen Primärschlüssel wird beim Anlegen der Tabelle automatisch erzeugt.

Statt einen leeren Schemaknoten zu erzeugen (vgl. Kapitel 2.12.1 Seite 95), kann ein neuer Schemaknoten auch auf Basis der bereits vorhandenen Tabellen und Beziehungen einer Datenbank im FirstSpirit-Projekt angelegt werden (siehe Kapitel 2.12.2 Seite 98).

Abhängig von den Einstellungen des Projektadministrators für die konfigurierte Datenbank können die Änderungen innerhalb eines Schemas im JavaClient, beispielsweise das Hinzufügen einer Tabelle in die physikalische Datenbank übernommen ("Sync") oder unterbunden werden ("kein Sync").

Tabellenvorlage: Für jede innerhalb des Schemas modellierte Tabelle kann eine Tabellenvorlage (unterhalb des Schema-Knotens) erzeugt werden. In diesen Tabellenvorlagen wird festgelegt, über welche Eingabeelemente der Redakteur später die Daten in die entsprechenden Tabellen einpflegen kann bzw. über welche Eingabeelemente der Redakteur Daten einer Referenztabelle übernehmen kann (siehe Kapitel 2.12.4.1 Seite 109). Über das Register "Mapping" kann außerdem die Zuordnung der über die Eingabekomponente gepflegten Inhalte zu einer Datenbank-Tabelle der physikalischen Datenbank hergestellt werden (siehe Kapitel 2.12.4.2 Seite 109). Das Mapping definiert damit den Speicherort der Inhalte in der Datenbank. Über die Register Vorlagensätze kann das Aussehen der Datensätze für die Generierung in den einzelnen Ausgabekanälen festegelegt werden (siehe Kapitel 2.12.4.3 Seite 111).

Abfragen: Für jedes Datenbank-Schema können außerdem Abfragen angelegt werden (siehe Kapitel 2.12.5.1 Seite 112). In diesen Abfragen werden Einschränkungen vorgenommen, anhand derer die Ergebnistabelle ausgewertet wird. Die vorgenommenen Einschränkungen werden dann bei der Ausgabe der Datensätze einer Tabelle berücksichtigt.

Sicht auf eine Datenbank: Innerhalb der Datenquellen-Verwaltung von FirstSpirit arbeiten die Redakteure auf einer "Sicht" der Datenbank. Dazu wird eine Tabelle mit Verknüpfung zur Datenbanktabelle angelegt. In dieser Tabelle werden die Daten in tabellarischer Form angezeigt. Abhängig von den Einstellungen des Projektadministrators für die konfigurierte Datenbank können die Redakteure entweder nur lesend auf die Datenbank-Inhalte zugreifen und diese beispielsweise als Ergebnis einer Abfrage sortiert auf einer Seite ausgeben ("Content-Projektion"), oder zusätzlich schreibend zugreifen und damit neue Inhalte in die Datenbank einfügen. Sofern ein schreibender Zugriff gestattet ist, können neue Datensätze hinzugefügt oder bestehende Datensätze verändert werden. Dazu stehen dem Redakteur die in der Tabellenvorlage definierten Eingabeelemente zur Verfügung (siehe Kapitel 2.12.4.1 Seite 109).

4 Arbeitsabläufe

Ein Arbeitsablauf ist eine Abfolge von Aufgaben, die nach einer fest vorgegebenen Struktur abgearbeitet werden. Für die jeweiligen Aufgaben können sowohl Fälligkeitszeitpunkte als auch berechtigte Personengruppen festgelegt werden. In FirstSpirit integrierte Arbeitsabläufe sind die Aufgabenerteilung und die Freigabeanforderung.

Mithilfe eines grafischen Arbeitsablauf-Editors können projektspezifische Arbeitsabläufe in der Vorlagen-Verwaltung erstellt werden (siehe Kapitel 4.2 Seite 130).

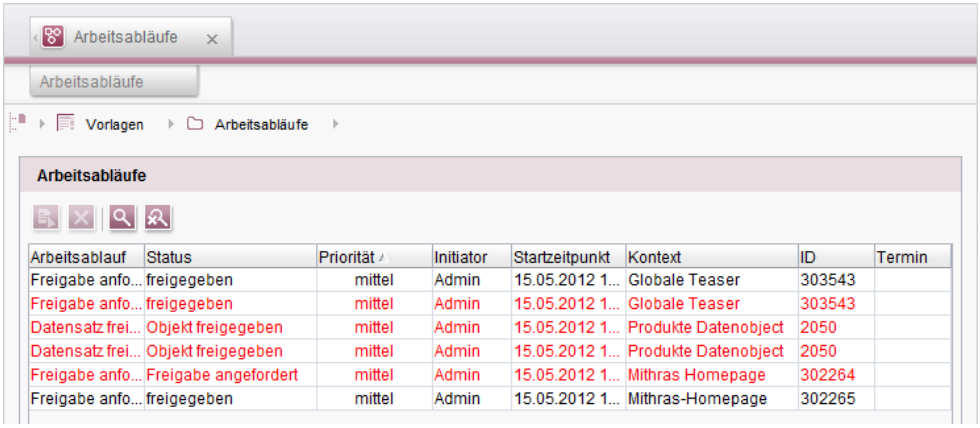
Instanzen dieser Arbeitsabläufe können anschließend kontextgebunden auf jedem Element innerhalb des FirstSpirit-Projekts oder kontextlos über die FirstSpirit-Menüleiste gestartet werden. Jede Instanz eines Arbeitsablaufs muss entsprechend der im Arbeitsablauf festgelegten Regeln durchlaufen werden.

Auf dem Wurzelknoten "Arbeitsabläufe" in der Vorlagen-Verwaltung befindet sich eine Übersicht über alle offenen bzw. bereits geschlossenen Arbeitsabläufe (Instanzen) innerhalb des Projekts (siehe Kapitel 4.1 Seite 123), wobei eine gefilterte Ansicht abhängig von verschiedenen Suchkriterien ebenfalls möglich ist (siehe Kapitel 4.1.1 Seite 125). Innerhalb der Übersicht können Aufgaben bearbeitet (siehe Kapitel 4.1.2 Seite 127) und wieder geschlossen werden (siehe Kapitel 4.1.3 Seite 129).

Weitere Informationen zum Starten und Weiterschalten von Arbeitsabläufen siehe "FirstSpirit Handbuch für Redakteure", Kapitel 12 "Arbeitsabläufe im FirstSpirit JavaClient" und "FirstSpirit Handbuch für Redakteure (WebClient)".

4.1 Übersicht

Auf dem Wurzelknoten "Arbeitsabläufe" in der Vorlagen-Verwaltung wird eine Übersicht über alle offenen bzw. bereits geschlossenen Arbeitsabläufe (Instanzen) innerhalb des Projekts dargestellt.



Arbeitsablauf	Status	Priorität	Initiator	Startzeitpunkt	Kontext	ID	Termin
Freigabe anfo...	freigegeben	mittel	Admin	15.05.2012 1...	Globale Teaser	303543	
Freigabe anfo...	freigegeben	mittel	Admin	15.05.2012 1...	Globale Teaser	303543	
Datensatz frei...	Objekt freigegeben	mittel	Admin	15.05.2012 1...	Produkte Datenobject	2050	
Datensatz frei...	Objekt freigegeben	mittel	Admin	15.05.2012 1...	Produkte Datenobject	2050	
Freigabe anfo...	Freigabe angefordert	mittel	Admin	15.05.2012 1...	Mithras Homepage	302264	
Freigabe anfo...	freigegeben	mittel	Admin	15.05.2012 1...	Mithras-Homepage	302265	

Abbildung 4-1: Übersicht Arbeitsabläufe

ein Klick auf das Icon öffnet die Aufgabenliste zum Bearbeiten der Aufgabe (siehe Kapitel 4.1.2 Seite 127).

ein Klick auf das Icon schließt die markierte Aufgabe (siehe Kapitel 4.1.3 Seite 129).

ein Klick auf das Icon öffnet den Dialog "Aufgabensuche" zur Definition einer Aufgaben-Suchfilters (siehe Kapitel 4.1.1 Seite 125).

ein Klick auf das Icon hebt die gefilterte Darstellung auf und zeigt stattdessen die Gesamtansicht aller noch offenen Arbeitsabläufe an (siehe Kapitel 4.1.1 Seite 125).

Die (gefilterten oder ungefilterten) Arbeitsabläufe werden in der Tabelle aufgelistet. Dabei stehen zu jeder Aufgabe die folgenden Informationen zur Verfügung:

Arbeitsablauf: Name des Arbeitsablaufs, der gestartet wurde.

Status: Status, in dem sich die aktuelle Instanz des Arbeitsablaufs befindet.

Priorität: Die aktuelle Priorität, die für die Bearbeitung der Aufgabe definiert wurde.

Initiator: Login-Name des Bearbeiters, der den Arbeitsablauf gestartet hat.

Startzeitpunkt: Datum und Uhrzeit, zu dem der Arbeitsablauf gestartet wurde.

Kontext: Wurde der Arbeitsablauf auf einem Element, beispielsweise einer Seite oder einem Medium gestartet, wird dieses Element angezeigt. Mit einem Doppelklick auf der Zeile wechselt der Kontext direkt auf das entsprechende Element in der Baumansicht.

ID: Wurde der Arbeitsablauf auf einem Element, beispielsweise einer Seite oder einem Medium gestartet, wird die ID des Elements angezeigt. Mit einem Doppelklick auf die Zeile wechselt der Kontext direkt auf das entsprechende Element in der Baumansicht.

Termin: Wurde die aktuelle Aufgabe terminiert, wird hier der Termin angezeigt.

Bei einem Doppelklick auf einen (kontextabhängigen) Auftrag innerhalb der Tabelle, wechselt der Fokus im JavaClient direkt auf das Element in der Baumansicht, auf dem der Auftrag gestartet wurde.

Eine Mehrfachauswahl innerhalb der Tabelle ist durch gleichzeitiges Drücken der SHIFT- bzw. der STRG-Taste möglich.

Sortierung nach Spalteninhalt: Mit einem Klick auf die jeweilige Spaltenüberschrift kann die Sortierung der Einträge in der Tabelle geändert werden. Beim ersten Klick auf eine beliebige Spaltenüberschrift werden die Einträge aufsteigend, bei einem weiteren Klick absteigend (nach Spalteninhalt) sortiert. Die Sortierung wird mit einem Icon hinter der Spaltenüberschrift angezeigt:

Startzeitpunkt
15.05.2012 11:11
15.05.2012 16:05
15.05.2012 16:05
15.05.2012 16:05
15.05.2012 16:07
15.05.2012 16:07

Abbildung 4-2: Sortierung nach Spalteninhalt (aufsteigend)

Ein dritter Klick hebt die Sortierung wieder auf.

4.1.1 Aufgabensuche (gefilterte Übersicht)

Über den Dialog "Aufgabensuche" können Arbeitsabläufe bzw. Aufgaben nach Filtern wie Arbeitsablauf, Element-ID, Bearbeiter usw. gefiltert werden. Der Dialog wird mit einem Klick auf das Icon  geöffnet:

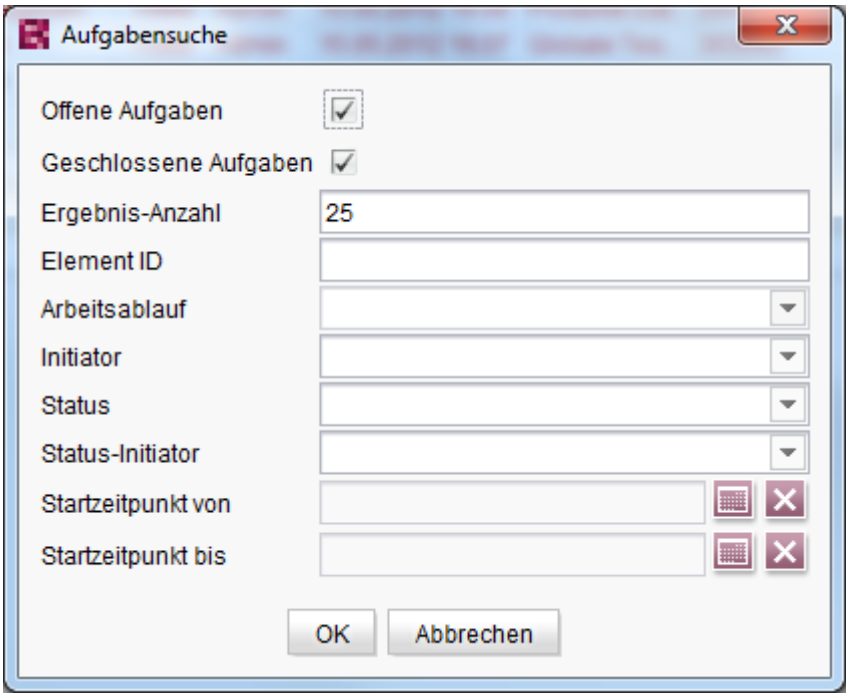


Abbildung 4-3: Dialog "Aufgabensuche" zum Filtern der Ansicht

Offene Aufgaben: Es werden alle "offenen Aufgaben" angezeigt. Offen sind alle Aufgaben, die den Endstatus (des Arbeitsablaufs) bisher nicht erreicht haben.

Geschlossene Aufgaben: Es werden alle "geschlossenen Aufgaben" angezeigt. Geschlossen sind alle Aufgaben, die den Endstatus (des Arbeitsablaufs) erreicht haben.

Ergebnis-Anzahl: Die Anzahl der gefundenen Aufgaben, die den eingegebenen Filterkriterien entsprechen, kann auf eine maximale Anzahl an Ergebnissen eingeschränkt werden. Entsprechen mehr Ergebnisse den Suchkriterien als maximal erlaubt, werden nur die aktuellsten Ergebnisse angezeigt.

Element-ID: Eindeutige ID des Objekts, auf dem der Arbeitsablauf gestartet wurde. Handelt es sich um einen kontextlosen Arbeitsablauf, wird ein leeres Feld angezeigt.

Arbeitsablauf: Name des Arbeitsablaufs, der gestartet wurde. Abhängig von der "Bevorzugten Anzeigesprache" im Menü "Ansicht" wird entweder der eindeutige Referenzname oder der sprachabhängige Anzeigename des Arbeitsablaufs angezeigt.

Initiator: Login-Name des Bearbeiters, der den Arbeitsablauf gestartet hat. Dabei wird die Suche nach Teilstrings unterstützt. Das bedeutet, das Suchergebnis muss dem Suchbegriff nicht genau entsprechen. Stattdessen werden alle Ergebnisse

angezeigt, die den Suchbegriff enthalten.

Status: Status, in dem sich die aktuelle Instanz des Arbeitsablaufs befindet. Dabei wird die Suche nach Teilstrings unterstützt. Das bedeutet, das Suchergebnis muss dem Suchbegriff nicht genau entsprechen. Stattdessen werden alle Ergebnisse angezeigt, die den Suchbegriff enthalten.

Status-Initiator: Login-Name des Bearbeiters, der die aktuelle Instanz des Arbeitsablaufs in den aktuellen Status geschaltet hat. Dabei wird die Suche nach Teilstrings unterstützt. Das bedeutet, das Suchergebnis muss dem Suchbegriff nicht genau entsprechen. Stattdessen werden alle Ergebnisse angezeigt, die den Suchbegriff enthalten.

Startzeitpunkt von / Startzeitpunkt bis: Über das Icon  kann die Datumsauswahlkomponente geöffnet werden. Hier kann ein Datum für den Start- bzw. Endzeitpunkt der Suche angegeben werden. Ausschlaggebend ist immer das Datum, an dem ein Arbeitsablauf gestartet wurde. Wird nur ein Startdatum angegeben, werden alle Arbeitsabläufe des aktuell angegebenen Tages gesucht.

Mit einem Klick auf den Button werden die Aufgaben nach den eingegebenen Kriterien gefiltert. Ein Klick auf das Icon  hebt die gefilterte Darstellung auf und zeigt stattdessen die Gesamtansicht aller noch offenen Arbeitsabläufe an.

4.1.2 Aufgaben bearbeiten

Mit einem Klick auf das Icon  im Dialog "Übersicht Arbeitsabläufe" (siehe Abbildung 4-1) kann die Aufgabenliste geöffnet werden. Die innerhalb der Übersicht selektierte Aufgabe wird auch in der Aufgabenliste markiert. Sofern der Benutzer die erforderlichen Rechte zum Schalten des Arbeitsablaufs besitzt, werden die Transitionen im unteren Bereich der Aufgabenliste direkt angezeigt.

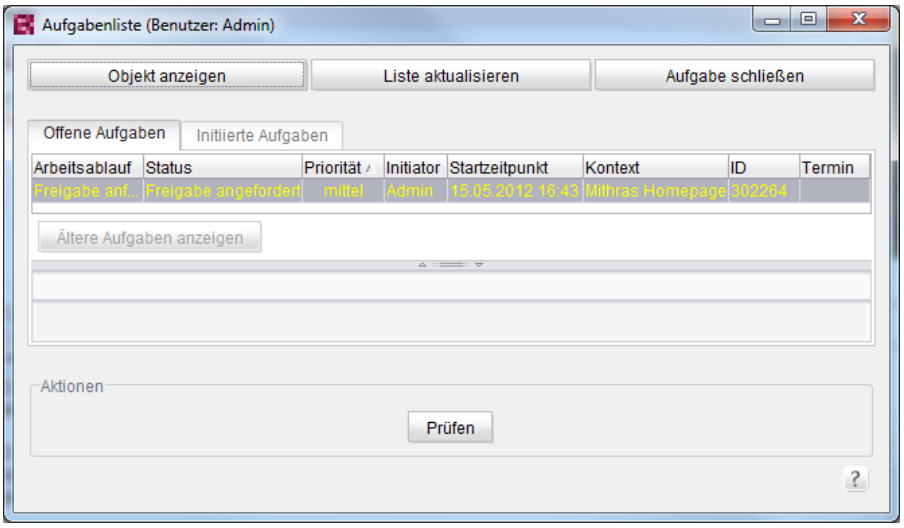


Abbildung 4-4: Aufgabenliste

Aus Performance-Gründen werden in der Aufgabenliste auf den Registern "Offene Aufgaben" und "Initiierte Aufgaben" initial nur 25 Aufgaben angezeigt. Liegen mehr Aufgaben vor, können diese über die Schaltfläche **Ältere Aufgaben anzeigen** eingeblendet werden.

Eine farbige Markierung der Aufgaben gibt z. B. an, ob der eingeloggte Benutzer für die Aufgabe direkt oder durch seine Gruppenzugehörigkeit als Bearbeiter ausgewählt ist (rote Schrift), ob der eingeloggte Benutzer nicht als Bearbeiter ausgewählt ist (schwarze Schrift) oder ob es sich um eine ungültige Aufgabe handelt (roter Hintergrund).

Ungültige Aufgaben können z. B. dadurch entstehen, dass ein Objekt, auf dem eine Arbeitsablauf aktiv ist, gelöscht wird. Diese können nicht weitergeschaltet, sondern nur über die Schaltfläche "Aufgabe schließen" geschlossen werden. Kann die Aufgabe repariert werden, z. B. wenn das gelöschte Objekt, zu dem der Arbeitsablauf noch besteht, wiederhergestellt wird, wird im Bereich Aktionen die Schaltfläche "Aufgabe reparieren" eingeblendet. Durch Ausführen dieser Aktion werden die Aufgabe, die Status-Farbe und der Schreibschutz zurückgesetzt.

Eine Mehrfachauswahl ist durch gleichzeitiges Drücken der SHIFT- bzw. der STRG-Taste möglich (alle Aufgaben können über die Tastenkombination STRG + A ausgewählt werden). Werden mehrere Aufgaben innerhalb der Liste markiert, können diese in einem Bearbeitungsschritt weitergeschaltet werden (siehe Kapitel 4.11.3 Seite 208).

Die Aufgabenliste kann auch über das Menü "Aufgaben" oder mit einem Klick auf das Icon  in der FirstSpirit-Symbolleiste geöffnet werden.)

 *Weitere Informationen zur Aufgabenliste siehe "FirstSpirit Handbuch für Redakteure".*

4.1.3 Aufgaben schließen

Unter bestimmten Voraussetzungen kann es notwendig sein, eine offene Aufgabe zu schließen, obwohl der Endstatus noch nicht erreicht wurde. Das Schließen einer Aufgabe kann mit einem Klick auf das Icon  im Dialog "Übersicht Arbeitsabläufe" erfolgen (siehe Abbildung 4-1).

Die Funktionalität entspricht dem Button "Aufgabe schließen", der innerhalb der Aufgabenliste verfügbar ist.

Eine Mehrfachauswahl ist durch gleichzeitiges Drücken der SHIFT- bzw. der STRG-Taste möglich (alle Aufgaben können über die Tastenkombination STRG + A ausgewählt werden). Werden mehrere Aufgaben innerhalb der Liste markiert, können diese in einem Bearbeitungsschritt gelöscht werden.

Vor dem Löschen der Aufgaben erfolgt eine Sicherheitsabfrage.

 *Geschlossene Aufgaben können nicht wiederhergestellt werden.*

4.2 Modellierung von Arbeitsabläufen

4.2.1 Anlegen eines Arbeitsablaufs

Neue, projektspezifische Arbeitsabläufe werden über das Kontextmenü auf dem Wurzelknoten "Arbeitsabläufe" in der Vorlagen-Verwaltung oder auf einem Ordner innerhalb dieses Knotens angelegt. Ein Klick auf den Eintrag "Arbeitsablauf anlegen" erstellt einen neuen Arbeitsablauf innerhalb der Baumdarstellung.

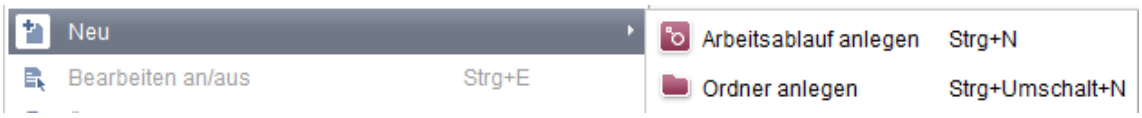


Abbildung 4-5: Anlegen über das Kontextmenü

Im rechten Bearbeitungsfenster öffnet sich ein grafischer Editor zur Modellierung eines neuen Arbeitsablaufs. Standardmäßig werden dort ein Startzustand mit einer Transition zur ersten Aktivität des Arbeitsablaufs und ein Endzustand angezeigt.

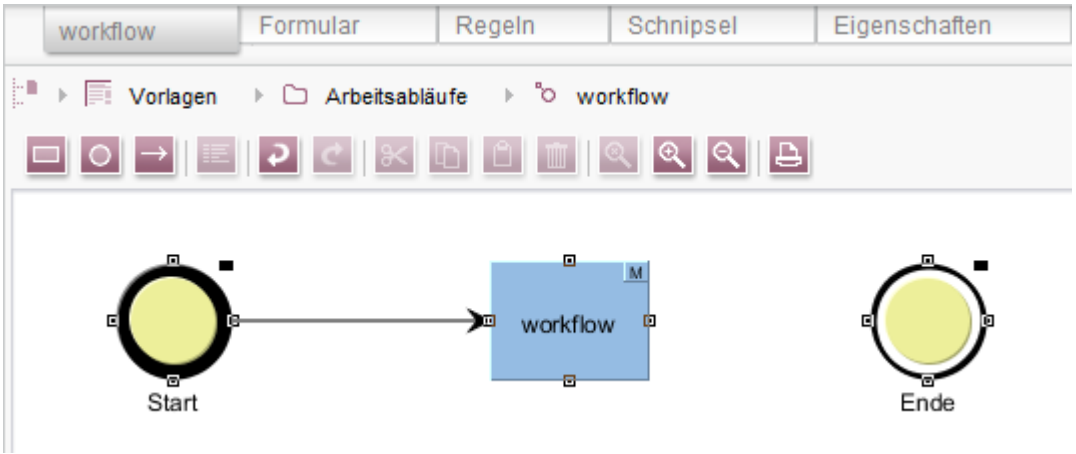


Abbildung 4-6: Initialer Status nach Erstellen eines neuen Arbeitsablaufs

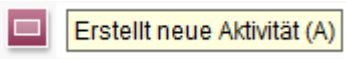
Innerhalb des Editors kann nun durch das Hinzufügen weiterer Status, Aktivitäten und Transitionen der Arbeitsablauf modelliert werden (siehe Kapitel 4.2.3 ff.).

Jeder Arbeitsablauf muss dabei mit einem Startzustand beginnen und mit einem Endzustand enden.

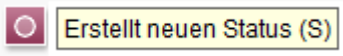
Die Bedienung des Editors erfolgt entweder über die Symbolleiste (siehe Kapitel 4.2.2 Seite 131) oder über ein Kontextmenü, dass an einer beliebigen Position des Editors aktiviert werden kann.

4.2.2 Symbolleiste des Arbeitsablauf-Editors

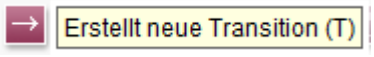
Abbildung 4-7: Symbolleiste des Arbeitsablauf-Editors



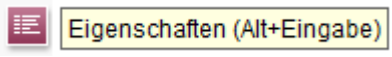
Neue Aktivität erstellen: Eine neue Aktivität wird durch einen Klick auf das Icon (oder das Tastaturkürzel A) im Editor angelegt (siehe Kapitel 4.2.3.2 Seite 133).



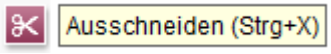
Neuen Status erstellen: Ein neuer Status wird durch einen Klick auf das Icon (oder das Tastaturkürzel S) im Editor angelegt (siehe Kapitel 4.2.3.1 Seite 132).



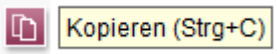
Eine neue Transition wird durch einen Klick auf das Icon (oder das Tastaturkürzel T) im Editor angelegt (siehe Kapitel 4.2.3.3 Seite 133).



Eigenschaften ändern, ein Klick auf dieses Icon öffnet das Eigenschaften Fenster des aktivierten Arbeitsablauf Elements.

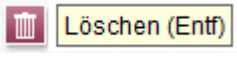


Element ausschneiden, ein Klick auf dieses Icon schneidet alle markierten Elemente des Arbeitsablauf-Editors aus und kopiert sie in den Zwischenspeicher. (Mehrere Elemente können durch Ziehen eines Rahmens mit der Maus markiert werden.)

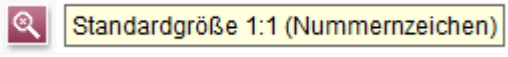


Element kopieren, ein Klick auf dieses Icon kopiert alle markierten Elemente des Arbeitsablauf-Editors in den Zwischenspeicher.

Element einfügen, ein Klick auf dieses Icon fügt die in den Zwischenspeicher kopierten Elemente in den Arbeitsablauf-Editor ein.

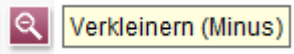


Löschen, mithilfe dieses Icons kann ein Element aus dem Arbeitsablauf Prozess entfernt werden.



Zoom 1:1, mithilfe dieses Icons können die Elemente des Arbeitsablauf-Editors wieder in Originalgröße dargestellt werden.

Zoom in, mithilfe dieses Icons können die Elemente des Arbeitsablauf-Editors vergrößert dargestellt werden.



Zoom out, mithilfe dieses Icons können die Elemente des Arbeitsablauf-Editors verkleinert dargestellt werden.

Drucken, mithilfe dieses Icons (oder über das Tastaturkürzel Strg + P) ist es möglich, die Grafik des Arbeitsablaufs auszudrucken. Es öffnet sich ein Fenster für die Druck-Einstellungen (siehe Kapitel 4.2.8 Seite 137).

4.2.3 Elemente des grafischen Arbeitsablauf-Editors

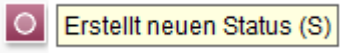
Innerhalb des Editors stehen drei unterschiedliche Objekttypen zur Verfügung, über die neue Arbeitsablaufe modelliert und konfiguriert werden können:

- Zustände oder Status (siehe Kapitel 4.2.3.1Seite 132)
- Aktivitäten (siehe Kapitel 4.2.3.2 Seite 133)
- Übergänge oder Transitionen (siehe Kapitel 4.2.3.3 Seite 133)

4.2.3.1 Zustand / Status

Abbildung 4-8: Status (Zustände) im Arbeitsablauf-Editor

Zustände, auch als Status bezeichnet, werden durch Kreise dargestellt. Ein Zustand ist das Ergebnis einer (automatischen oder manuellen) Aktivität. Zustände geben den Status an, in dem sich ein Arbeitsablauf befinden kann.



Ein neuer Zustand wird durch einen Klick auf das Icon (oder das Tastenkürzel S) im Editor angelegt. Abhängig von der Konfiguration kann der Status:

- ein Startzustand (besitzt nur ausgehende Transitionen),
- ein Endzustand (besitzt nur eingehende Transitionen)
- oder ein normaler Status (besitzt ein- und ausgehende Transitionen) sein.

Die Darstellung der unterschiedlichen Typen wird im Editor durch einen dunklen Rahmen hervorgehoben (bei Start- und Endstatus) (siehe Abbildung 4-8).

4.2.3.2 Aktivität

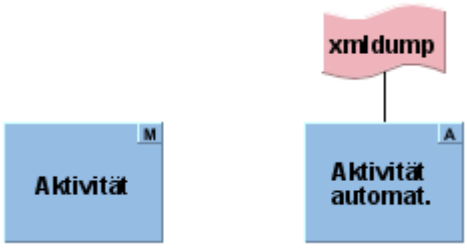
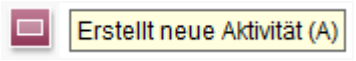


Abbildung 4-9: Aktivitäten im Arbeitsablauf-Editor

Aktivitäten werden durch Rechtecke dargestellt. Eine Aktivität besteht aus der Durchführung einer Aufgabe (z. B. "Prüfen") und dem Auslösen einer Aktion (z. B. Klick auf den Button "Freigabe erteilen").

Die Ausführung einer Aktivität kann entweder manuell durch einen Benutzer ausgeführt werden, oder automatisch durch ein Skript (siehe Kapitel 4.5.4 Seite 154).

Manuelle Aktivitäten werden im Editor mit einem "M" in der rechten, oberen Ecke gekennzeichnet (siehe Abbildung 4-9 – linke Aktivität), automatische Aktivitäten werden im Editor mit einem "A" in der rechten, oberen Ecke gekennzeichnet (siehe Abbildung 4-9 – rechte Aktivität).



Eine neue Aktivität wird durch einen Klick auf das Icon (oder das Tastenkürzel A) im Editor angelegt. Abhängig von der Konfiguration kann die Aktivität:

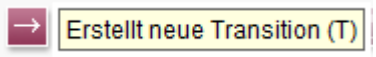
- Manuell (durch einen Bearbeiter) oder
- Automatisch (durch ein Skript) ausgeführt werden.

4.2.3.3 Übergang / Transition

Abbildung 4-10: Transition im Arbeitsablauf-Editor

Übergänge werden durch Pfeile dargestellt. Übergänge bilden die Verbindung zwischen einer Aktivität und einem Status. Hier werden die Rechte für ein

Arbeitsablauf Modell definiert. Das Abbrechen einer Aktion führt dazu, dass der vorherige Status (vor dem Schalten der Transition) erhalten bleibt. Das Abbrechen muss im Arbeitsablauf nicht getrennt modelliert werden.



Eine neue Transition wird durch einen Klick auf das Icon (oder das Tastenkürzel T) im Editor angelegt.

4.2.4 Tastaturkürzel im Arbeitsablauf-Editor

A	Anlegen einer neuen Aktivität.
T	Anlegen einer neuen Transition.
S	Anlegen eines neuen Status.
Strg + P	Anfordern einer Druckvorschau des Arbeitsablauf-Modells.
Alt + Eingabe	Öffnen des Dialogfensters Eigenschaften für ein markiertes Element innerhalb des Arbeitsablauf-Modells.
Strg + Z	Rückgängig
Strg + Umschalt + Z	Wiederherstellen
Strg + X	Ausschneiden
Strg + C	Kopieren
Strg + V	Einfügen
Entf	Löschen

4.2.5 Bedienungshilfen zum Editor

Durch einen einfachen Klick auf ein Element innerhalb des Editors, wird ein Element selektiert. Mit gedrückter linker Maustaste kann dieses Element an die gewünschte Stelle im Editor verschoben werden. Eingehende und ausgehende Übergänge folgen dabei dem verschobenen Element.

Mit der Maus lässt sich ein Rahmen um mehrere Elemente ziehen. So lassen sich mehrere Elemente gleichzeitig verschieben, ausschneiden oder kopieren.

Ist im Arbeitsablauf-Editor ein Status selektiert, dann bewirkt die Funktion **Aktivität einfügen**, dass die neue Aktivität automatisch durch eine Transition mit diesem Status verbunden wird. Analog dazu wird bei einer selektierten Aktivität und der Funktion **Status einfügen** eine Transition zwischen der Aktivität und dem neuen Status angelegt.

Übergänge sind automatisch immer gerade Verbindungen von der Quelle zum Ziel. Um vor allem die Darstellung von Schleifen übersichtlicher gestalten zu können, können auf einem Übergang Stützpunkte eingefügt werden. Die Verbindung zwischen zwei Stützpunkten ist wieder eine Gerade, es können aber beliebig viele Stützpunkte hinzugefügt werden.

Um einen Stützpunkt hinzuzufügen, muss der Übergang an der gewünschten Stelle mit der rechten Maustaste angeklickt werden. Wird ein Stützpunkt mit der rechten Maustaste angeklickt, dann wird dieser Stützpunkt wieder entfernt.

Mit gedrückter linker Maustaste kann ein Stützpunkt an die gewünschte Position im Editor verschoben werden.

4.2.6 Regeln der Modellierung

- Jeder Arbeitsablauf besitzt genau einen *Startzustand*.
- Der Startzustand kann *genau eine ausgehende Transition* verfolgen. Da im Startzustand keine Auswahl stattfinden kann, wird immer die erste ausgehende Transition berücksichtigt.
- Übergänge stellen eine zielgerichtete Verbindung zwischen genau einem Quell- und genau einem Zielelement dar.
- Quell- und Zielelement einer Transition können nur Zustände und Aktivitäten aber keine anderen Übergänge sein.
- Transitionen können immer nur zwischen *einem* Zustand und *einer* Aktivität existieren, niemals zwischen zwei Zuständen oder zwei Aktivitäten.
- Zustände und Aktivitäten können beliebig viele eingehende und ausgehende Transitionen haben (Ausnahmen: Startzustand und Endzustand).
- Zustände und Aktivitäten sollten immer einen (für den Arbeitsablauf) eindeutigen Namen besitzen.
- Transitionen *dürfen* einen Namen haben, dieser *muss* dann eindeutig in Bezug auf sein Ausgangs-Element sein.
- Jeder Arbeitsablauf besitzt eine fest definierte Menge von Endzuständen, es

muss mindestens ein *Endzustand* definiert sein.

- Ein Endzustand darf *keine ausgehenden* Transitionen besitzen.

4.2.7 Beispiele zu Modellierungsregeln

- Nach einem Zustand folgt immer eine Aktivität. Zustand und Aktivität sind mit "Übergängen" verbunden und benötigen einen eindeutigen Namen. Für Übergänge kann ein Name vergeben werden, der dann in Bezug auf sein Ausgangselement eindeutig sein muss.

Abbildung 4-11: Modellierungsregel Status und Aktivitäten

- Auf einen Zustand können mehrere Aktivitäten folgen. Ebenso können mehrere Aktivitäten zu einem Zustand führen.

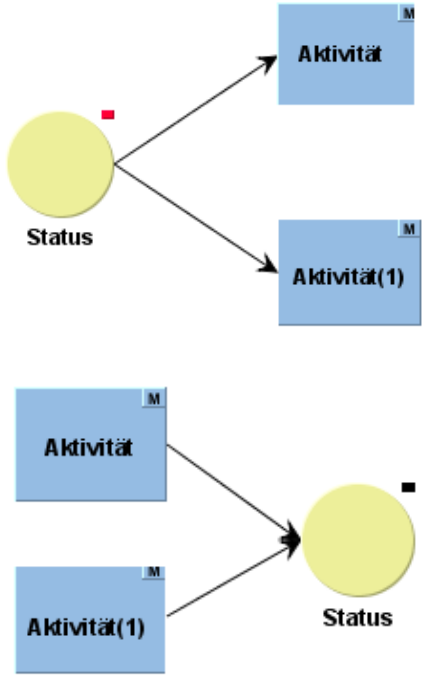


Abbildung 4-12: Modellierungsregel ein Status, mehrere Aktivitäten

- Eine Aktivität kann zu mehreren Zuständen führen. Ebenso können mehrere Zustände eine Aktivität auslösen.

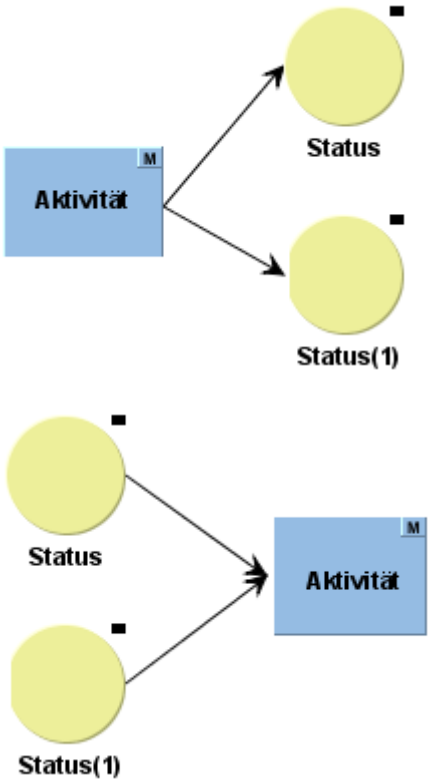


Abbildung 4-13: Modellierungsregel mehrere Status, eine Aktivität

- Ein Skript kann nur an einer (automatischen) Aktivität hängen, wobei die Verbindungslinie keine Richtung hat.

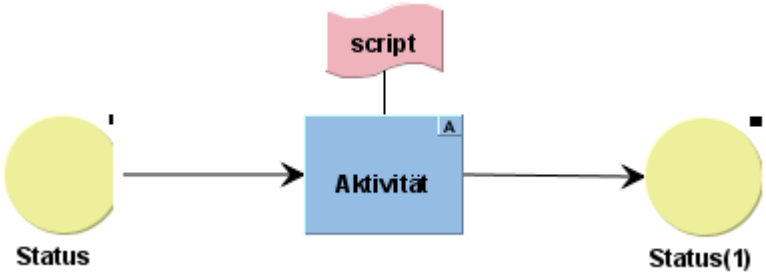


Abbildung 4-14: Modellierungsregel Aktivitäten und Skripte

4.2.8 Druckvorschau für Arbeitsablauf-Modelle

Eine Druckvorschau des modellierten Arbeitsablaufs kann mit einem Klick auf den Button Drucken (oder über das Tastaturkürzel Strg + P) innerhalb des Arbeitsablauf-Editors angefordert werden (siehe Kapitel 4.2.2 Seite 131). Es öffnet sich ein Fenster für die Druck-Einstellungen.

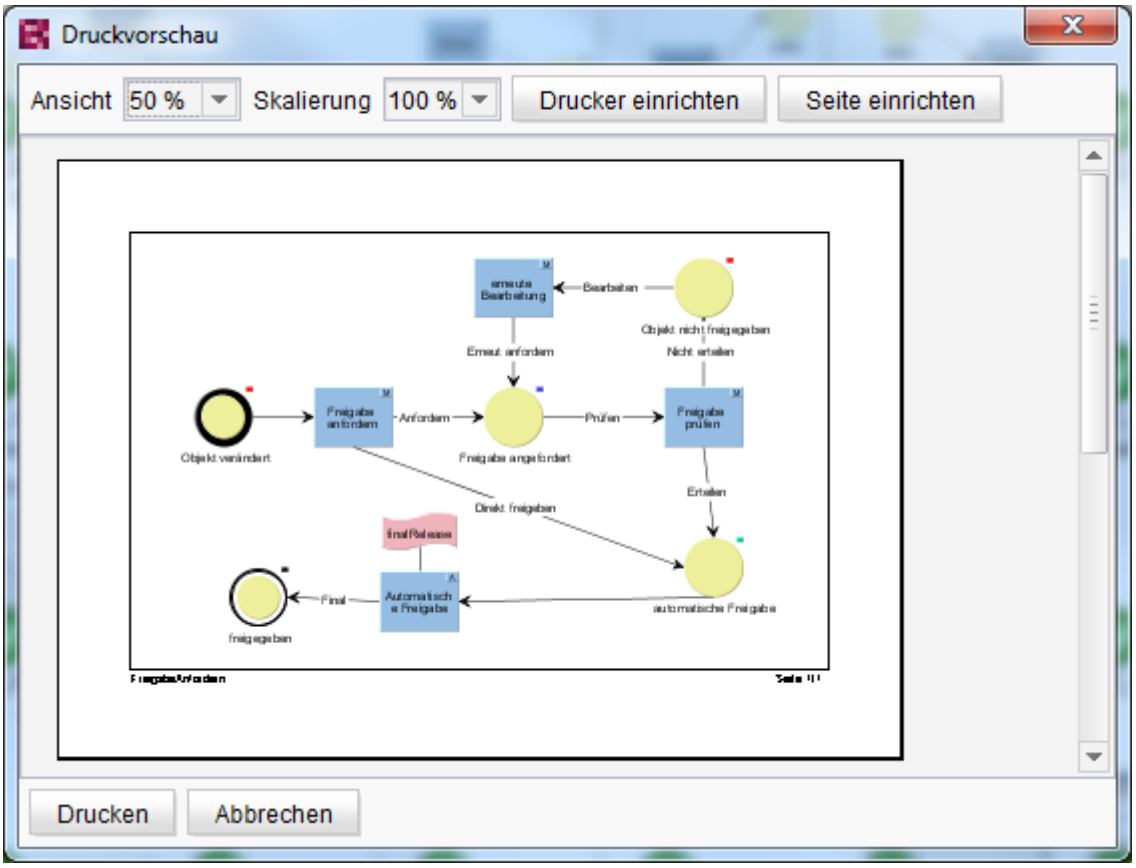


Abbildung 4-15: Druckvorschau

Ansicht: Über die Combobox kann prozentual die Größe der Seiten im Vorschaufenster eingestellt werden.

Skalierung: Über die Combobox kann prozentual die Größe des Arbeitsablauf-Modells auf der Druckvorschau-Seite gewählt werden. Bei entsprechend großer Darstellung werden mehrere Vorschauseiten angezeigt.

Drucker einrichten Mit einem Klick auf den Button öffnet sich ein Fenster in dem Druckeinstellungen vorgenommen werden können.

Seite einrichten Mit einem Klick auf den Button öffnet sich ein Fenster, in dem einige Einstellungen für die gedruckten Seiten vorgenommen werden können.

Drucken Ein Klick auf diesen Button startet den Druckvorgang.

Abbrechen Ein Klick auf diesen Button bricht den Druckvorgang ab.

4.3 Fehlerbehandlung innerhalb von Arbeitsabläufen

4.3.1 Allgemeine Fehlerbehandlung

Beim Starten: Tritt eine Exception beim Starten des Arbeitsablaufs auf, beispielsweise weil der Benutzer kein Recht zum Schalten der Transitionen eines Arbeitsablaufs besitzt, wird der Arbeitsablauf auf dem Objekt nicht gestartet.

Beim Schalten: Anders sieht es aus, wenn der Arbeitsablauf bereits gestartet wurde, und während des Schaltens einer Transition ein Fehler bzw. eine Exception auftritt. In diesem Fall, wird der Status vor dem Schalten der Transition, also der letzte "fehlerfreie" Status beibehalten. Ist innerhalb des Arbeitsablauf-Modells ein Fehler-Status definiert, befindet sich das Element nach dem Auftreten der Exception im Fehler-Status (siehe Kapitel 4.3.2 Seite 139).

4.3.2 Fehler-Status

Es gibt viele Gründe für das Auftreten von Exceptions beim Ausführen eines Arbeitsablaufs, beispielsweise eine Fehlkonfiguration im Modell des Arbeitsablaufs oder ein Skriptfehler in einem angehängten Skript. Um diese Fehler zuverlässig abzufangen und zu verhindern, dass sich eine Instanz des Arbeitsablaufs, nach dem Schalten einer Transition, in einem inkonsistenten Zustand befindet, steht ein optionaler Fehler-Status innerhalb der Modellierung von Arbeitsabläufen zur Verfügung.

Dazu wird einfach ein normaler Status im Modell hinzugefügt. Im Dialog "Eigenschaften" des Status muss nun der Typ "Fehler" aktiviert werden:

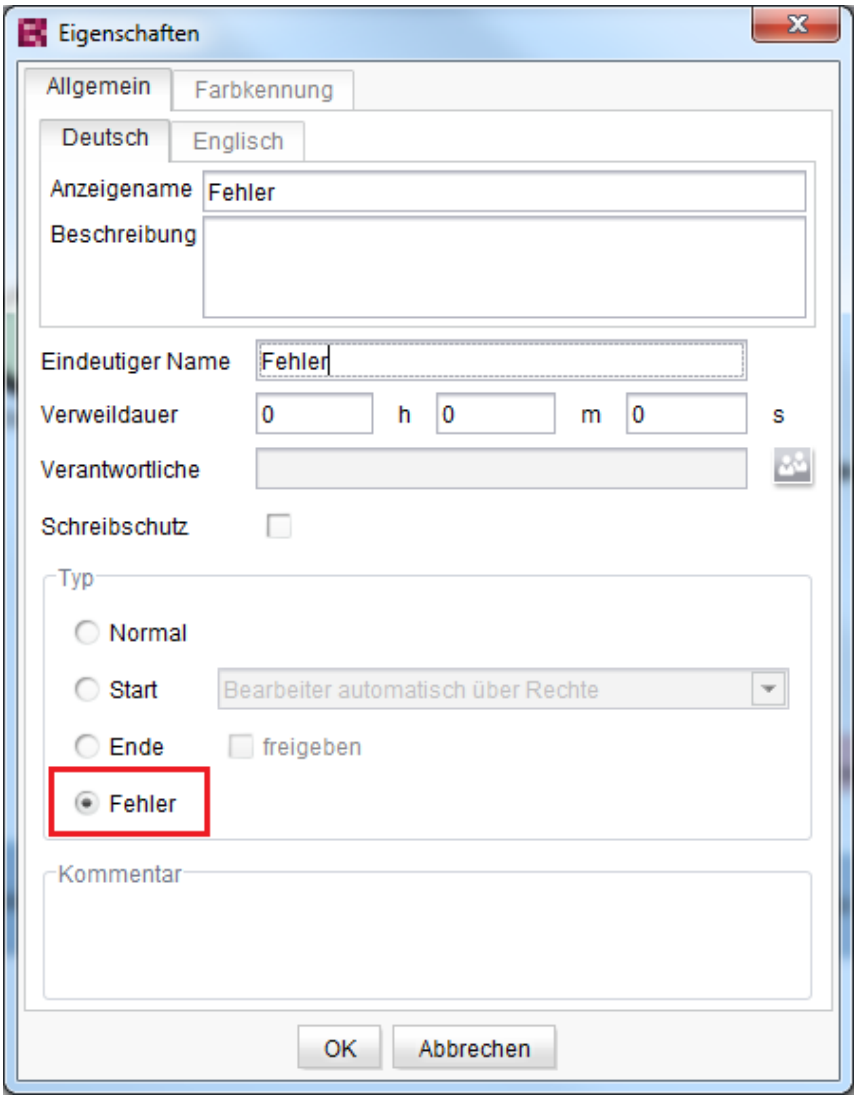


Abbildung 4-16: Fehler-Status konfigurieren

Der Status wird anschließend im Modell mit einem roten Rand angezeigt:

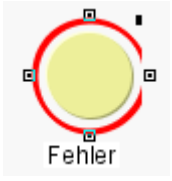


Abbildung 4-17: Fehlerstatus im Modell

Der Fehler-Status darf nicht über eine Transition geschaltet werden, besitzt also analog zum Start-Status nur ausgehende Transitionen. Über diese ausgehenden Transitionen wird die Fehlerbehandlung innerhalb des Arbeitsablaufs modelliert (siehe Abbildung 4-19).

Tritt nun an einer beliebigen Stelle im Arbeitsablauf eine Exception auf, erreicht die Instanz des Arbeitsablaufs direkt den Fehler-Status.

Der Fehler-Status fängt alle Exceptions ab, die innerhalb der Ausführung des Arbeitsablaufs auftreten, auch solche, die nicht innerhalb des Arbeitsablaufs behandelt werden. Beispiele für behandelte bzw. unbehandelte Exceptions werden in Kapitel 4.3.3 beschrieben.

Nach der Fehlerbehebung kann der Arbeitsablauf in den nachfolgenden Status (lt. Arbeitsablauf-Modell) weitergeschaltet werden.

Eine Übersicht, über alle Instanzen des Arbeitsablaufs, die fehlerhaft ausgeführt wurden, bietet die Aufgabenliste:

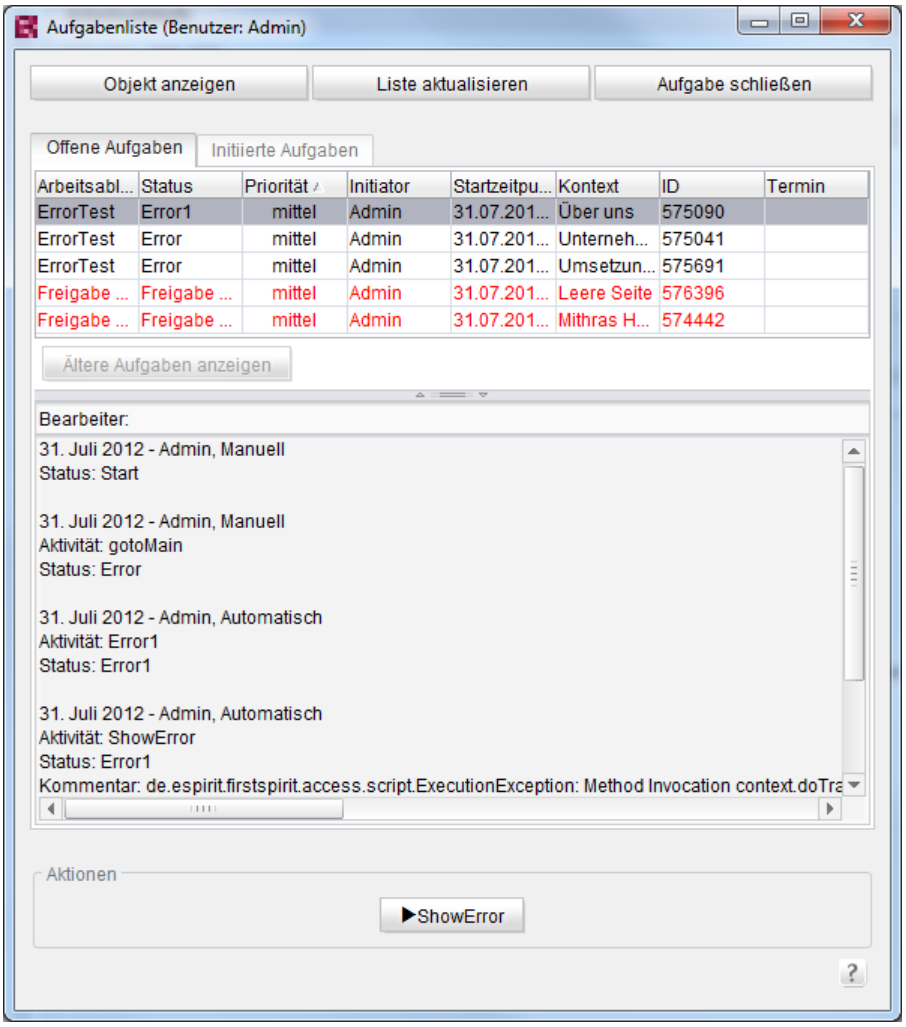


Abbildung 4-18: Aufgabenliste mit Aufgaben im Fehler-Status

Mit einem Klick auf die Tabellenbeschriftung "Status" können die Aufträge nach ihrem aktuellen Status sortiert werden.

Jeder Arbeitsablauf kann nur einen Fehler-Status besitzen. Wird ein Status als Fehler-Status definiert, obwohl im Arbeitsablauf-Modell bereits ein Fehler-Status existiert, wird der erste Status automatisch auf den Typ "normal" zurückgesetzt.

4.3.3 Beispiel: Arbeitsablauf "Error"

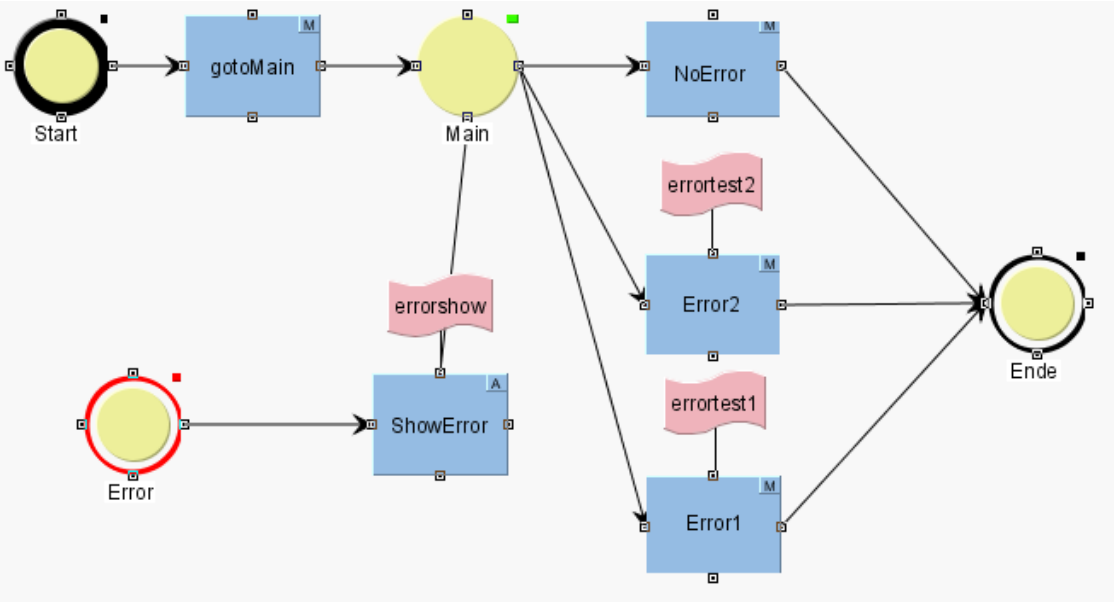


Abbildung 4-19: Beispiel-Arbeitsablauf "Error"

Der Arbeitsablauf besteht aus dem Arbeitsablauf "errortest" und den zugehörigen Skripten "errorshow", "errortest1" und "errortest2". Der Arbeitsablauf wird als komprimierte Zip-Datei zum Import in die Vorlagen-Verwaltung (Knoten "Arbeitsabläufe") zur Verfügung gestellt.

Skript: errortest1:

```
//!Beanshell
throw new IllegalArgumentException("Error test 1");
```

Das erste Skript "errortest1" wirft eine unbehandelte IllegalStateException. Diese Exception wird im Arbeitsablauf nicht behandelt, führt aber trotzdem dazu, dass nicht der Status "ende", sondern der Status "Error" erreicht wird.

Skript errortest2:

```
//!Beanshell
context.gotoErrorState("Error test 2",
new IllegalArgumentException("Error test 2"));
```

Das zweite Skript "errortest2" zeigt die Fehlerbehandlung innerhalb eines Skripts. Über `context.gotoErrorState(...)` wird die Instanz des Arbeitsablaufs beim Auftreten einer Exception direkt in den Status Error geschaltet.

Skript errorshow:

```
import de.espirit.firstspirit.common.gui.*;
import de.espirit.firstspirit.access.*;
import de.espirit.firstspirit.ui.operations.RequestOperation;
import de.espirit.firstspirit.agency.OperationAgent;


errorInfo = context.getTask().getErrorInfo();
if (errorInfo != null) {
text = new StringBuilder("<html>Fehlerinformationen:<br>");
text.append("<ul>");
text.append("<li>Benutzer: " + errorInfo.getUserLogin() + " (" +
errorInfo.getUserName() + ")");
text.append("<li>Kommentar: " + errorInfo.getComment());
text.append("<li>Aktivität: " + errorInfo.getErrorActivity());
text.append("<li>Error: " + errorInfo.getThrowable());
text.append("<li>ErrorInfo: " + errorInfo.getErrorInfo());
text.append("</ul>");
text = text.toString();
requestOperation =
context.requireSpecialist(OperationAgent.TYPE).getOperation(Reques
tOperation.TYPE);
requestOperation.setKind(RequestOperation.Kind.INFO);
requestOperation.addOk();
requestOperation.perform(text);


} else {
requestOperation =
context.requireSpecialist(OperationAgent.TYPE).getOperation(Reques
tOperation.TYPE);
requestOperation.setKind(RequestOperation.Kind.INFO);
requestOperation.perform("Es sind keine Fehlerinformationen
vorhanden.");
```

```
}
context.doTransition("->Main");
```

Das Skript "errorshow" blendet die Fehlerinformationen über einen Fehlerdialog ein. Beim Auftreten eines Fehlers wird der Dialog im Rahmen der Fehlerbehandlung automatisch über den Arbeitsablauf aufgerufen. Der Dialog enthält relevante Informationen zur Fehlerbehebung (z. B. den Benutzer, der den Arbeitsablauf gestartet hat, die Aktivität, die zum Fehler geführt hat, usw.):

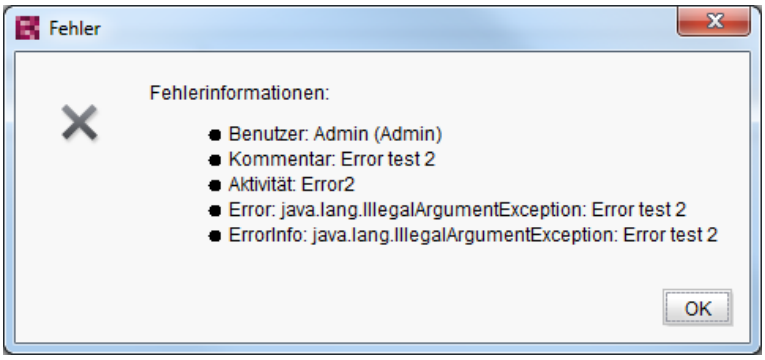


Abbildung 4-20: Dialog mit relevanten Fehler-Informationen

Nach Einblenden des Dialogs wird die Instanz des Arbeitsablaufs automatisch auf den Status "main" zurückgeschaltet:

4.4 Formularunterstützung für Arbeitsabläufe (Formular)

Innerhalb von Arbeitsabläufen können Formulare zur Eingabe von Inhalten verwendet werden. Die Formulare werden innerhalb des Registers "Formular" im Arbeitsablauf definiert:

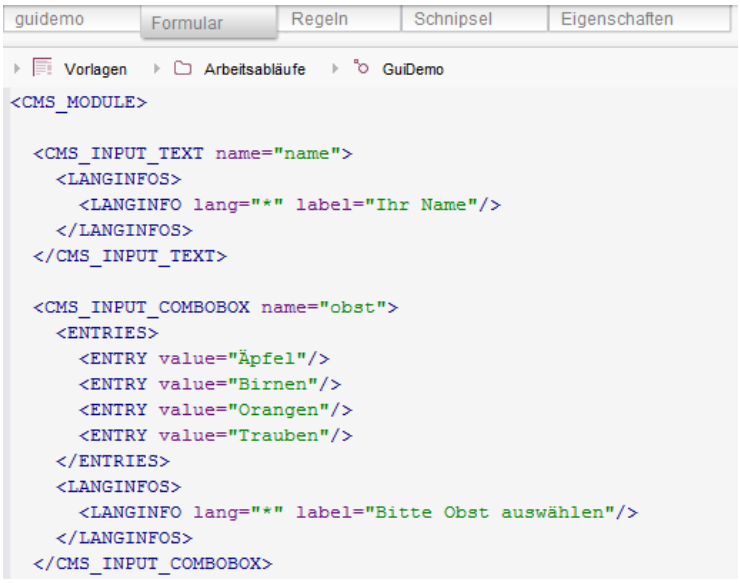


Abbildung 4-21: Register Formular (Arbeitsablauf-Modell)

Während der Ausführung des Arbeitsablaufs kann der Bearbeiter Werte über die Eingabekomponenten einpflegen, die im Formularbereich definiert wurden:

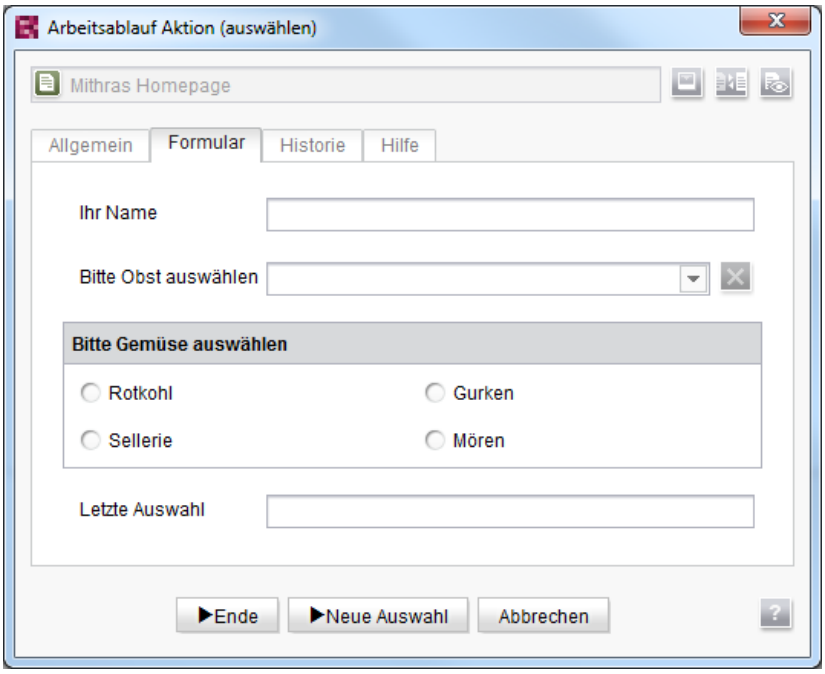


Abbildung 4-22: Formular innerhalb des Ausführung

Die gespeicherten Werte können zu einem späteren Zeitpunkt innerhalb des Arbeitsablaufs wieder ausgegeben werden:

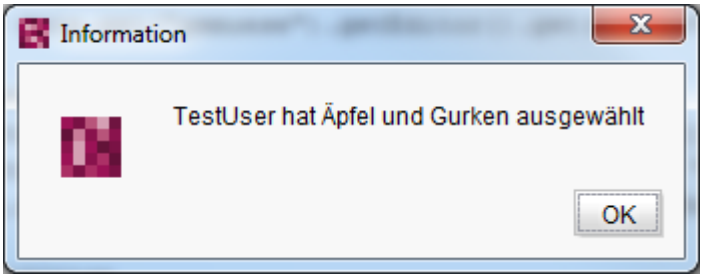


Abbildung 4-23: Informationsdialog mit Formularinhalten

4.4.1 Beispiel: Arbeitsablauf "GUI"

Innerhalb des Beispiel-Arbeitsablaufs wird über die Aktivität ein Skript "guitest", zur Anzeige der Formulare, ausgeführt.

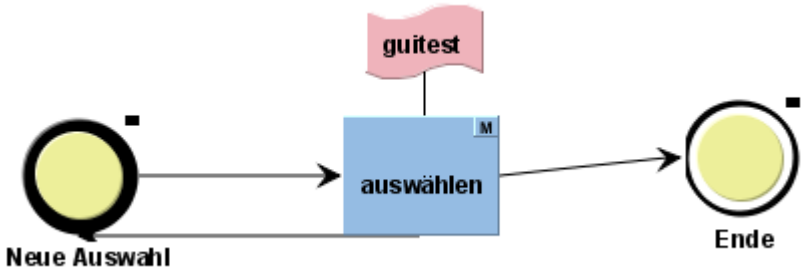


Abbildung 4-24: Beispiel-Arbeitsablauf "GUI"

Skript "guitest":

```
//!Beanshell

import de.espirit.firstspirit.common.gui.*;

import de.espirit.firstspirit.access.editor.*;

import de.espirit.firstspirit.ui.operations.RequestOperation;

import de.espirit.firstspirit.agency.OperationAgent;


se = context.getStoreElement();

transition = context.showActionDialog(); data = context.getData(); if
(transition != null) {

// display selected values

name = data.get("name").getEditor().get(EditorValue.SOLE_LANGUAGE);

obst = data.get("obst").getEditor().get(EditorValue.SOLE_LANGUAGE);

gemuese = data.get("gemuese").getEditor().get(EditorValue.SOLE_LANGUAGE);


// save selected values

lastSelection = data.get("lastSelection").getEditor();

lastSelection.set(EditorValue.SOLE_LANGUAGE, name + ", " + obst + ", " +
gemuese);

text = name + " hat " + obst + " und " + gemuese + " ausgewählt";

requestOperation =
context.requireSpecialist(OperationAgent.TYPE).getOperation(RequestOperat
ion.TYPE);

requestOperation.setKind(RequestOperation.Kind.INFO);

requestOperation.addOk();

requestOperation.perform(text);


// do transition

context.doTransition(transition);

} else {

requestOperation =
context.requireSpecialist(OperationAgent.TYPE).getOperation(RequestOperat
ion.TYPE);

requestOperation.setKind(RequestOperation.Kind.INFO);

requestOperation.perform("Sie haben keine Transition ausgewählt.");

}
```

4.5 Eigenschaften eines Arbeitsablaufs (Konfiguration)

4.5.1 Allgemeine Eigenschaften

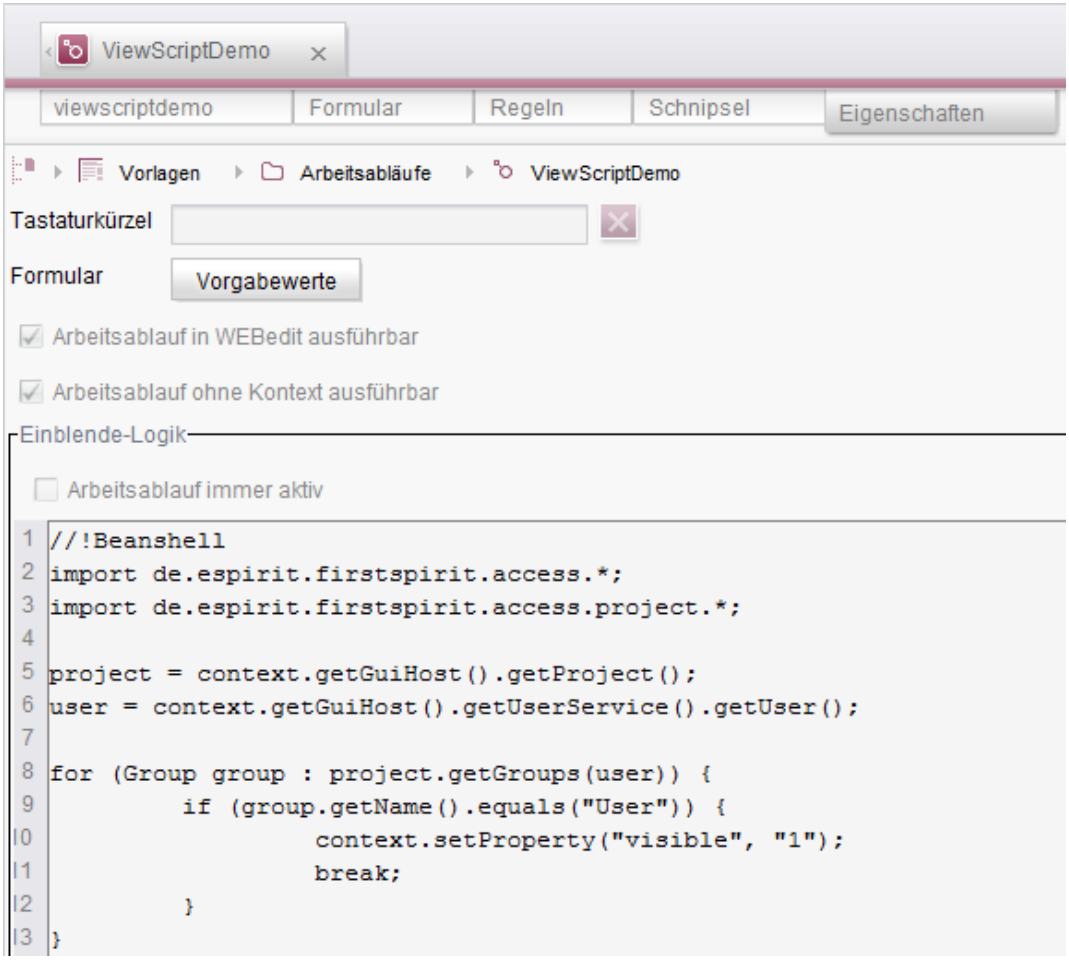


Abbildung 4-25: Register Eigenschaften (Arbeitsablauf-Modell)

Tastaturkürzel: Für jeden Arbeitsablauf kann in diesem Feld ein eindeutiges Tastaturkürzel definiert werden. Der Arbeitsablauf muss in diesem Fall nicht mehr über das Kontextmenü oder das Menü "Aufgaben" gestartet bzw. geschaltet werden, sondern kann direkt über das festgelegte Tastaturkürzel aufgerufen werden. Um ein neues Tastaturkürzel zu definieren, muss sich der Cursor innerhalb des Felds befinden. Anschließend genügt es, die gewünschte Tastenkombination über die Tastatur einzugeben. Die Eingabe wird dann in das Eingabefeld übernommen. Eine Texteingabe ist nicht möglich. Zum Ändern des Tastenkürzels den Cursor erneut im Feld positionieren und anschließend die neue Tastenkombination aufrufen. Um ein definiertes Tastenkürzel für den Arbeitsablauf zu löschen, die Taste "Esc" drücken.

Tastaturkürzel können nur für kontextgebundene Arbeitsabläufe verwendet werden.

Arbeitsablauf in WEBedit ausführbar: Ist die Checkbox aktiviert kann der

Arbeitsablauf nicht nur im JavaClient, sondern auch im WebClient ausgeführt werden.

Arbeitsablauf ohne Kontext ausführbar: Ist die Checkbox aktiviert, kann der Arbeitsablauf ohne einen Kontext zu einem (oder mehreren) Objekten gestartet werden. Der Standard-Arbeitsablauf "Aufgabe" kann beispielsweise kontextlos gestartet werden.

Einblendelogik: Über die Einblende-Logik können Arbeitsabläufe abhängig von bestimmten Eigenschaften eingeblendet oder ausgeblendet werden (siehe Kapitel 4.5.2 Seite 149).

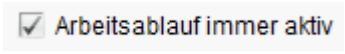
4.5.2 Einblendelogik für Arbeitsabläufe

Im Register "Eigenschaften" eines Arbeitsablaufs kann der Arbeitsablauf in der Vorlagen-Verwaltung mit einer Einblende-Logik versehen werden. Über die Einblende-Logik können Arbeitsabläufe abhängig von bestimmten Eigenschaften eingeblendet oder ausgeblendet werden. Die Einblende-Logik bezieht sich nur auf das Starten des Arbeitsablaufs (nicht auf die Sichtbarkeit innerhalb der Vorlagen-Verwaltung). Unterbindet die Einblende-Logik das Starten eines Arbeitsablaufs, beispielsweise zu einem bestimmten Zeitpunkt oder für eine bestimmte Gruppe, wird dieser Arbeitsablauf über das Kontextmenü (bei kontextgebundenen Arbeitsabläufen) und über die Menüfunktion "Aufgaben – Arbeitsablauf starten" (bei kontextlosen Arbeitsabläufen) nicht mehr angezeigt.

Die Einblende-Logik wird projektspezifisch über ein BeanShell-Skript realisiert. Für jeden Arbeitsablauf können so spezifische Einblendeoptionen hinterlegt werden.

Mögliche Anwendungsfälle:

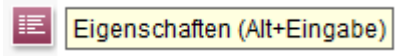
- Arbeitsabläufe dürfen nur in einem bestimmten Zeitraum ausgeführt werden (z. B. nur Montags von 8.00 – 9.00 Uhr)
- Arbeitsabläufe dürfen nur von einem bestimmten Benutzer bzw. einer bestimmten Gruppe ausgeführt werden (vgl. Abbildung 4-25 Gruppe "Redakteure"). Dieser Fall kann zwar auch durch eine Konfiguration der Rechte zur Ausführung eines Arbeitsablaufs realisiert werden, das ist aber nur bei kontextgebundenen Arbeitsabläufen möglich. Für kontextlose Arbeitsabläufe kann das ein- bzw. ausblenden über die Einblende-Logik realisiert werden.
- Arbeitsabläufe dürfen nur für bestimmte Elemente, z. B. nur Bild-Medien eingeblendet werden. Die Konfiguration über die Rechte zur Ausführung eines Arbeitsablaufs auf den einzelnen Elementen wäre abhängig von der Anzahl der Bild-Medien entsprechend umfangreich. Einfacher ist dieser Anwendungsfall daher über die Einblende-Logik des Arbeitsablaufs zu realisieren.



Soll die Einblende-Logik deaktiviert werden, kann die Checkbox "Arbeitsablauf immer aktiv" aktiviert werden. Der Arbeitsablauf wird in diesem Fall, unabhängig von der Einblende-Logik, immer eingeblendet. Die hinterlegte Einblendelogik wird zwar nicht mehr ausgewertet, bleibt aber enthalten und kann durch das Deaktivieren der Checkbox wieder aktiviert werden.

Handelt es sich um einen kontextgebundenen Arbeitsablauf wird zusätzlich zur Einblende-Logik das Recht zum Starten des Arbeitsablaufs auf dem Element ausgewertet. Besitzt der Benutzer nicht das Recht zum Starten des Arbeitsablaufs, wird der Arbeitsablauf unabhängig von der Einblende-Logik nicht angezeigt.

4.5.3 Eigenschaften eines Status



Ist ein Status innerhalb des Arbeitsablauf-Editors markiert, kann mit einem Klick auf das Icon, über das Kontextmenü, mit der Tastenkombination Alt + Eingabe oder mit einem Doppelklick, das Fenster "Eigenschaften" angefordert werden (siehe Kapitel 4.2.2 Seite 131). Anschließend können über die Register "Allgemein" und "Farbkennung" Einstellungen für das ausgewählte Element vorgenommen werden.

4.5.3.1 Register Allgemein

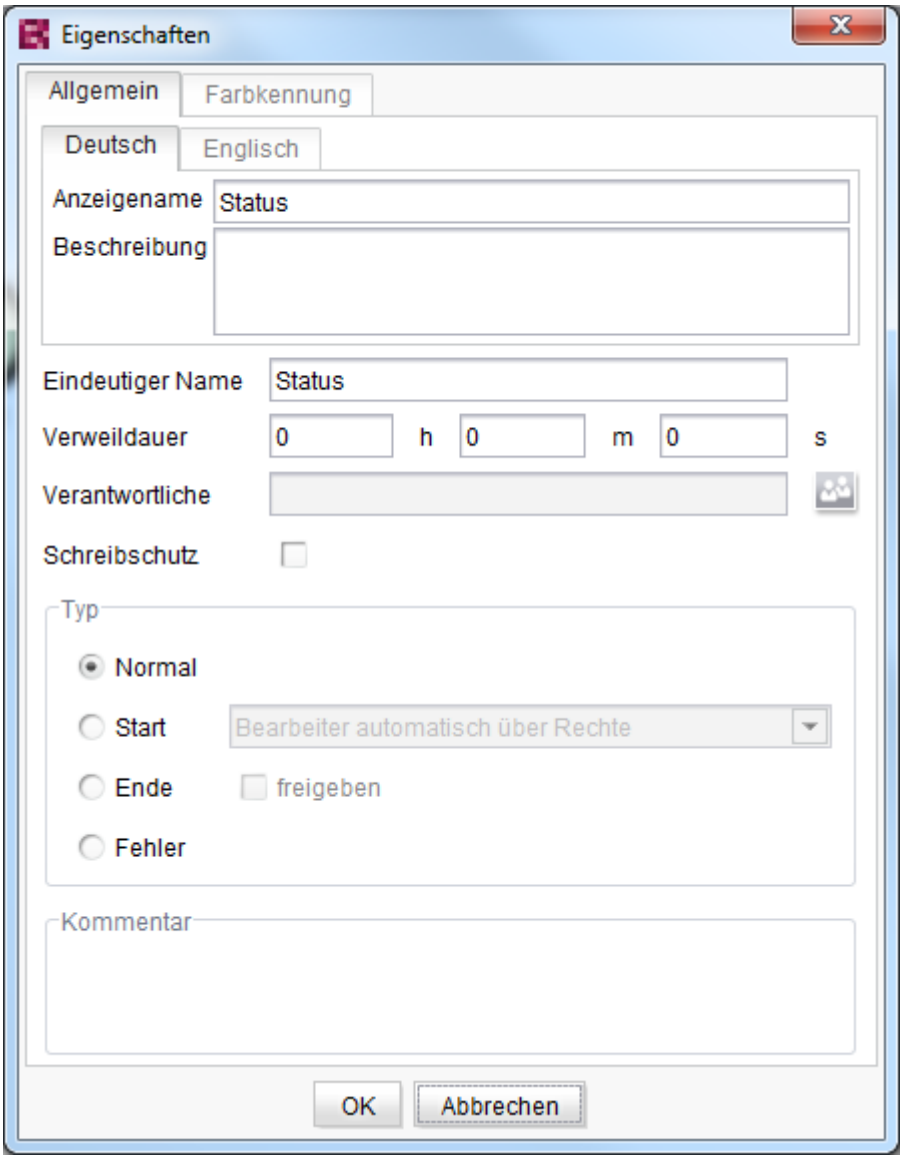


Abbildung 4-26: Eigenschaften eines Status (Allgemein)

Eindeutiger Name: In diesem Feld muss für den ausgewählten Zustand ein eindeutiger Name vergeben werden (Zeichenbeschränkung: <= 40 Zeichen).

Verweildauer: Hier kann eine Zeitspanne angegeben werden, die ein Arbeitsablauf in dem aktuellen Zustand verweilen darf, bevor eine Nachricht an die verantwortlichen Benutzer bzw. Gruppen verschickt wird.

Verantwortliche: In diesem Feld sind die verantwortlichen Benutzer bzw. Gruppen aufgelistet, die bei einer Überschreitung der Verweildauer benachrichtigt werden

sollen. Durch einen Klick auf das Symbol  öffnet sich ein weiteres Fenster, in dem die Verantwortlichen aus einer Liste ausgewählt werden können.

Zur Verwendung der Gruppen- bzw. Benutzerauswahl siehe FirstSpirit Handbuch für Redakteure, Kapitel 13.2.4 "Berechtigte Gruppen / Benutzer ändern".

Schreibschutz: Ist diese Option aktiviert, dann wird für das entsprechende Objekt der Bearbeitungsmodus gesperrt, solange es sich in diesem Status befindet (siehe Kapitel 4.7 Seite 176).

Status-Typ: Hier kann der aktuelle Zustand als Start- oder Endknoten definiert werden. Jeder Arbeitsablauf benötigt genau einen **Startstatus** und mindestens einen **Endstatus** (siehe auch Kapitel 4.2.3.1 Seite 132).

- **Normal:** Standardeinstellung, gilt für alle Zustände, die nicht Start- oder Endzustand sind.
- **Start:** Beschreibt den Zustand eines Objektes, bei dem der Arbeitsablauf gestartet wird. Über den Startzustand wird die Auswahl der berechtigten Benutzer definiert, die den zukünftigen Status einer laufenden Arbeitsablauf-Instanz schalten dürfen (siehe Kapitel 4.6 Seite 163)
    - Bearbeiter manuell (je Aktion)
      (siehe Kapitel 4.6.2.1Seite 166)
    - Bearbeiter automatisch über Rechte
      (siehe Kapitel 4.6.2.2 Seite 167)
- **Ende:** Beschreibt einen möglichen Zustand, in dem sich ein Objekt nach Beendigung des Arbeitsablaufes befinden kann. Es kann zusätzlich festgelegt werden, ob eine Freigabe des Objektes erfolgen soll, sobald dieser Endzustand erreicht wird.

Kommentar: In diesem Feld kann ein erklärender Kommentar zu dem aktuellen Zustand angegeben werden. Dieser Kommentar wird als Tooltip im Arbeitsablauf-Editor angezeigt.

Über die Felder **Anzeigename** und **Beschreibung** können zusätzlich sprachabhängige Anzeigenamen und Beschreibungen hinzugefügt werden. Dabei handelt es sich um die Redaktionssprachen (nicht die Projektsprachen). Redaktionssprachen werden vom Projektadministrator für ein Projekt festgelegt und

können anschließend über das Menü "Ansicht – Bevorzugte Anzeigesprache" vom Redakteur gewechselt werden. Der Anzeigename wird z. B. in Arbeitsablauf-Dialogen (Beschriftung der Buttons im Transitionsdialog, Hilfe- und Historie-Register), als Einträge im Kontextmenü zum Starten/Schalten der Arbeitsabläufe, die Beschreibung als Tooltip und auf dem Hilfe-Register verwendet. Wird kein Anzeigename angegeben, wird der eindeutige Name angezeigt. Ist keine Beschreibung vorhanden, wird der Text aus dem Feld **Kommentar** angezeigt.

4.5.3.2 Register Farbkennung

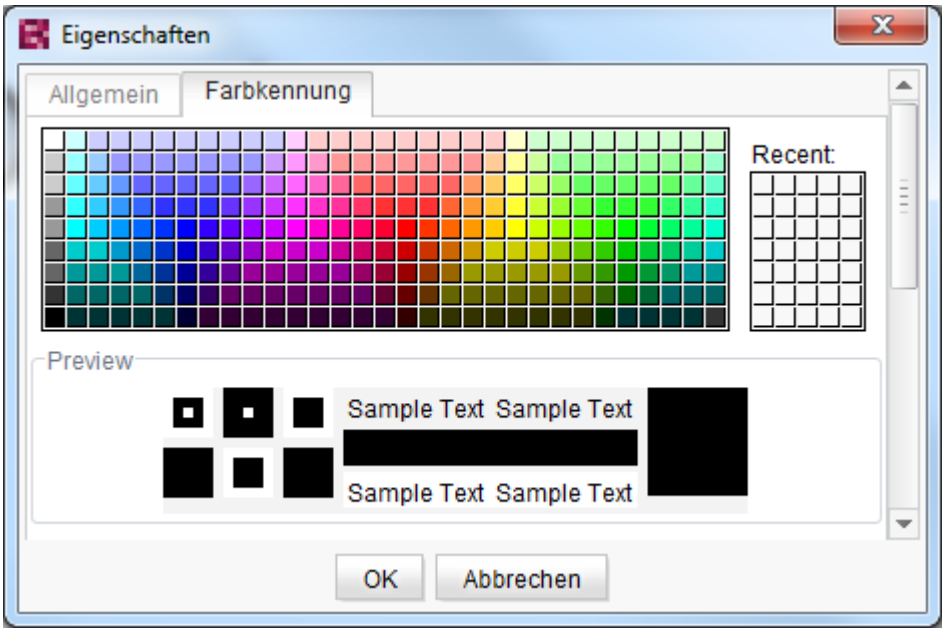
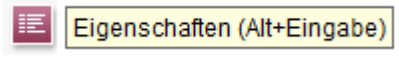


Abbildung 4-27: Eigenschaften eines Status (Farbkennung)

In diesem Register kann über das Farbschema die gewünschte Farbkennung für den aktuellen Zustand ausgewählt werden. Das Objekt in der Baumstruktur des FirstSpirit-Clients (auf dem der Arbeitsablauf gestartet wurde) wird durch diese Farbe hervorgehoben, sobald die Instanz des Arbeitsablaufs den entsprechenden Zustand erreicht hat.

Um ein späteres Finden einer bereits vorher ausgewählten Farbe zu erleichtern, sind im rechten Fensterbereich alle Farben aufgelistet, die innerhalb des Arbeitsablaufs bereits einmal ausgewählt wurden.

4.5.4 Eigenschaften einer Aktivität



Ist eine Aktivität innerhalb des Arbeitsablauf-Editors markiert, kann mit einem Klick auf das Icon, über das Kontextmenü, mit der Tastenkombination Alt + Eingabe oder mit einem Doppelklick, das Fenster "Eigenschaften" angefordert werden (siehe Kapitel 4.2.2 Seite 131). Anschließend können über die Register ""Allgemein" und "Email" Einstellungen für das ausgewählte Element vorgenommen werden.

4.5.4.1 Register Allgemein

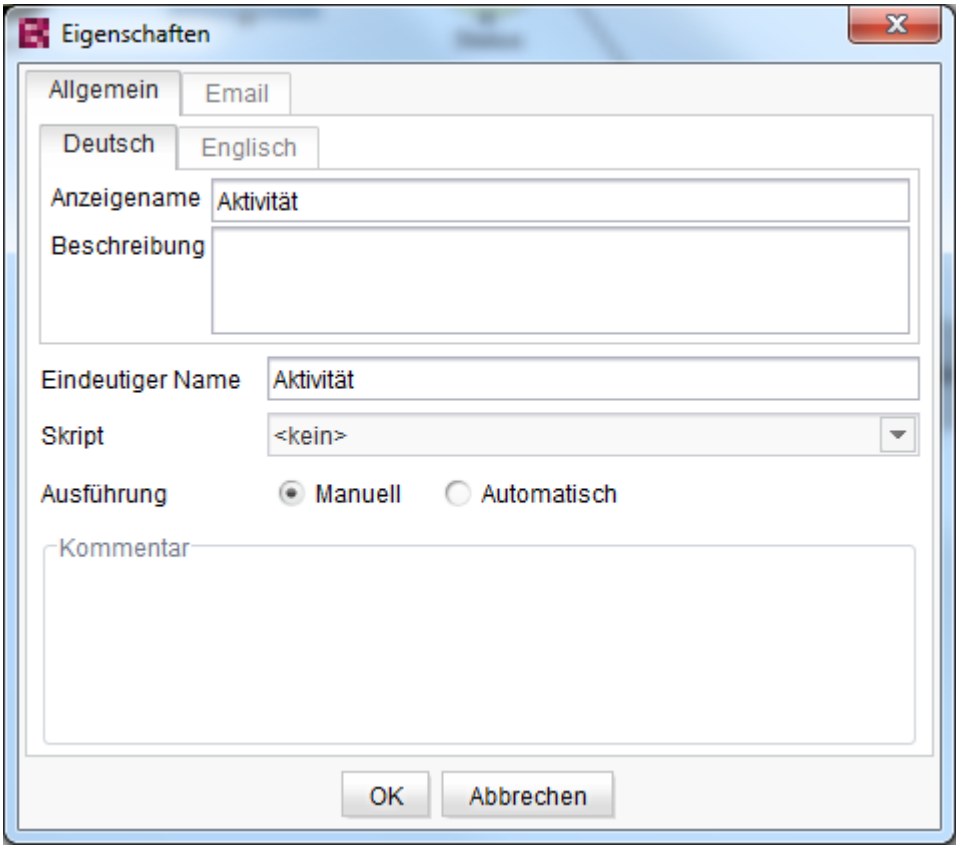


Abbildung 4-28: Eigenschaften einer Aktivität (Allgemein)

Eindeutiger Name: In diesem Feld muss für die ausgewählte Aktivität ein eindeutiger Name vergeben werden (Zeichenbeschränkung: <= 40 Zeichen).

Skript: Über die Combobox kann ein Skript (aus dem Projekt) ausgewählt werden, das ausgeführt wird, sobald diese Aktivität aufgerufen wird. Soll die erforderliche Aktivität über ein Skript ausgeführt werden, muss die automatische Ausführung

ausgewählt werden (siehe "Ausführung").

Ausführung: An dieser Stelle wird festgelegt, ob eine Aktivität manuell durch einen Benutzer erfolgen soll oder automatisch vom System (vgl. Kapitel 4.2.3.2 Seite 133):

- **Manuell:** Beim Ausführen einer *manuellen Aktivität* wird dem Bearbeiter ein Dialogfenster angezeigt, über das der Arbeitsablaufs (Instanz) weitergeschaltet werden kann.
- **Automatisch:** *Automatische Aktivitäten* erwartet keine Benutzerinteraktion und werden ausgeführt, sobald einer der im Modell vorgelagerten Zustände erreicht wird (d.h. die Aktion wird vom System und nicht vom Benutzer ausgelöst). Eine automatische Aktion (und damit auch ein ggf. angekoppeltes Skript) werden also direkt nach dem Erreichen eines Zustands ausgeführt. Das Skript kann sowohl die erforderliche Prüfung als auch das Weiterschalten des Arbeitsablaufs (Instanz) automatisch ausführen.

Kommentar: In diesem Feld kann optional ein erklärender Kommentar für diese Aktivität angeben werden.

Über die Felder **Anzeigename** und **Beschreibung** können zusätzlich sprachabhängige Anzeigenamen und Beschreibungen hinzugefügt werden. Dabei handelt es sich um die Redaktionssprachen (nicht die Projektsprachen). Redaktionssprachen werden vom Projektadministrator für ein Projekt festgelegt und können anschließend über das Menü "Ansicht – Bevorzugte Anzeigesprache" vom Redakteur gewechselt werden. Der Anzeigename wird z. B. in Arbeitsablauf-Dialogen (Beschriftung der Buttons im Transitionsdialog, Hilfe- und Historie-Register), als Einträge im Kontextmenü zum Starten/Schalten der Arbeitsabläufe, die Beschreibung als Tooltip und auf dem Hilfe-Register verwendet. Wird kein Anzeigename angegeben, wird der eindeutige Name angezeigt. Ist keine Beschreibung vorhanden, wird der Text aus dem Feld **Kommentar** angezeigt.

4.5.4.2 Register Email

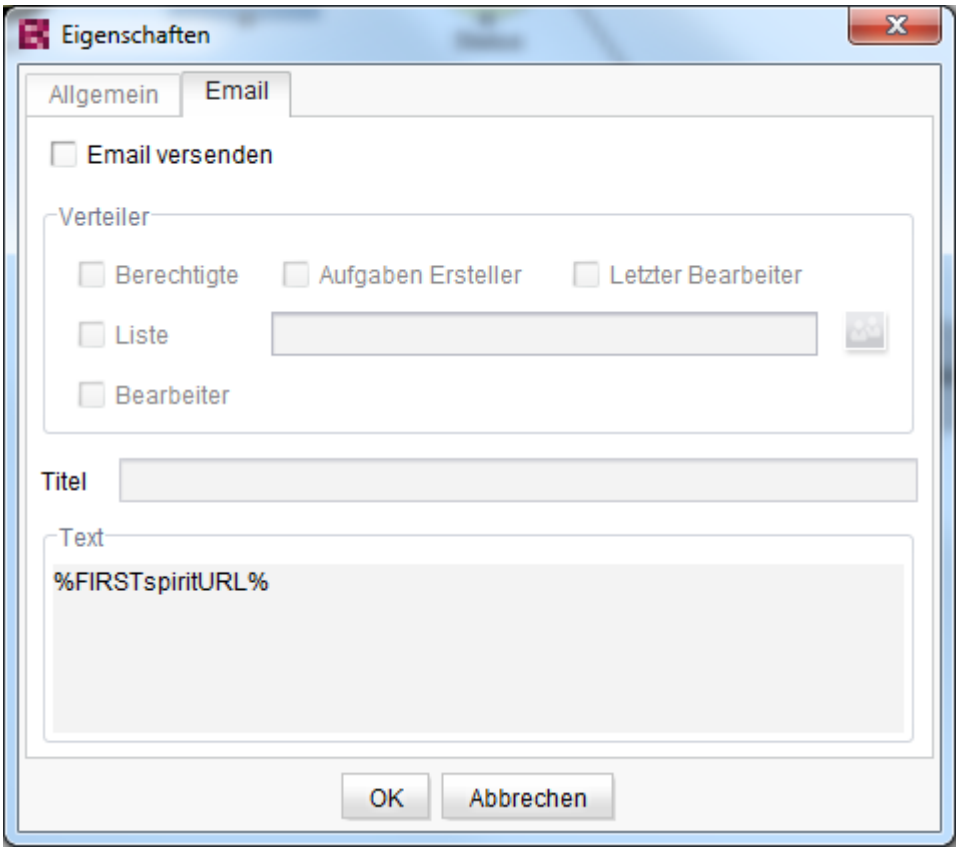


Abbildung 4-29: Eigenschaften einer Aktivität (Email)

Email versenden: Wird die Checkbox aktiviert, wird eine E-Mail an ausgewählte Empfänger verschickt (siehe "Verteiler"), sobald die Aktivität ausgeführt wurde.

Verteiler: Hier kann ausgewählt werden, an welche Personen eine E-Mail verschickt werden soll.

- **Berechtigte:** Personen, die berechtigt sind, den Arbeitsablauf in den nachfolgenden Status weiterzuschalten. Diese Rechte werden entweder über die Rechte zum Schalten der Transition direkt innerhalb des Arbeitsablauf-Modells definiert (siehe Kapitel 4.5.5.2 Seite 160) und/oder über die Rechte zum Schalten einer Transition auf dem Objekt, auf dem die Instanz des Arbeitsablaufs gestartet wurde.
- **Aufgaben-Ersteller:** Der Benutzer, der die Instanz des Arbeitsablaufs gestartet hat.

- **Letzter Bearbeiter:** Der Benutzer, der die Instanz des Arbeitsablaufs in den aktuellen Status geschaltet hat.

- **Liste:** Durch einen Klick auf das Symbol  öffnet sich ein Fenster, in dem die gewünschten Personen bzw. Gruppen aus einer Liste ausgewählt werden können.

Zur Verwendung der Gruppen- bzw. Benutzerauswahl siehe FirstSpirit Handbuch für Redakteure, Kapitel 13.2.4 "Berechtigte Gruppen / Benutzer ändern".

- **Bearbeiter:** Der aktuelle Bearbeiter des Arbeitsablaufs.

Titel: In diesem Feld wird der Text der Betreff-Zeile der E-Mail eingetragen.

Text: In diesem Feld wird die Nachricht eingetragen, die der Empfänger erhalten soll. Hierbei können folgende %-Ausdrücke als Platzhalter verwendet werden, die vom System automatisch ersetzt werden:

Platzhalter für die Erzeugung von kontextspezifischen Informationen:

`%FIRSTspiritURL%` = Verbindungsmodus HTTP (Standardmodus)
`%FIRSTspiritSOCKETURL%` = Verbindungsmodus SOCKET
`%PAGESTORE_PREVIEW_URL%` = Vorschau-URL einer Seite aus der Inhalte-Verwaltung
`%SITESTORE_PREVIEW_URL%` = Vorschau-URL einer Seitenreferenz aus der Struktur-Verwaltung
`%WF_NAME%` = Name des Arbeitsablaufs
`%CREATOR%` = Erzeuger des Arbeitsablaufs (kompletter Name)
`%LAST_USER%` = letzter Bearbeiter
`%LAST_COMMENT%` = letzter Kommentar
`%NEXT_USER%` = nächster Bearbeiter
`%PRIORITY%` = Priorität
`%DATE%` = Fälligkeitsdatum (nur wenn gesetzt)
`%HISTORY%` = Historie der Instanz des Arbeitsablaufs
`%WEBeditURL%` = WebEdit Link auf die Vorschau der Seite

Wenn die Platzhalter `%FIRSTspiritURL%`, `%FIRSTspiritRMIURL%` oder `%FIRSTspiritSOCKETURL%` im Feld "Text" angegeben werden, wird in der versendeten Mail ein Link (der auf den entsprechenden Knoten im Projekt verweist)

erzeugt, z. B. für `%FIRSTspiritURL%`:

http://myServer:9999/start/FIRSTspirit.jnlp?app=client&project=QS_akt&name=vorlage_1&type=Page&id=4394331&host=myServer&port=9999&mode=HTTP

oder für `%PAGESTORE_PREVIEW_URL%`:

http://myServer.espirit.de:9999/fs5preview/preview/4238727/page/DE/current/4238731/4394331

Über die anderen Platzhalter können weitere kontextspezifische Information zur jeweiligen Instanz des Arbeitsablaufs generiert werden, z. B. `%HISTORY%`:

```
16. April 2012 - Admin, Manuell
Aktivität: Freigabe anfordern
Status: Freigabe angefordert
Kommentar: UserB : Freigabe erteilen bitte
```

Neben dem JavaClient-URL (`%FIRSTspiritURL%`) kann im Text auch ein Verweis auf eine Vorschauseite im WebClient übergeben werden (`%WEBeditURL%`), z. B.:

`http://myServer:9999/fs5webedit/?project=476656&store=pagestore&element=477196`

Lässt sich ein Platzhalter nicht auflösen, weil die Information in dem gewählten Kontext nicht verfügbar ist, wird er durch eine entsprechende Information ersetzt:

- *DE: <in aktuellem Kontext nicht verfügbar>*
- *EN: <not available in current context>*

 *Die Platzhalter-Ersetzung funktioniert nur, wenn das JNLP-Servlet auf dem System installiert ist.*

 *Weiterführende Informationen zum JNLP-Servlet siehe FirstSpirit Handbuch für Administratoren, Kapitel 4.3.1.2 "Bereich: Server"*

4.5.5 Eigenschaften einer Transition

4.5.5.1 Register Allgemein

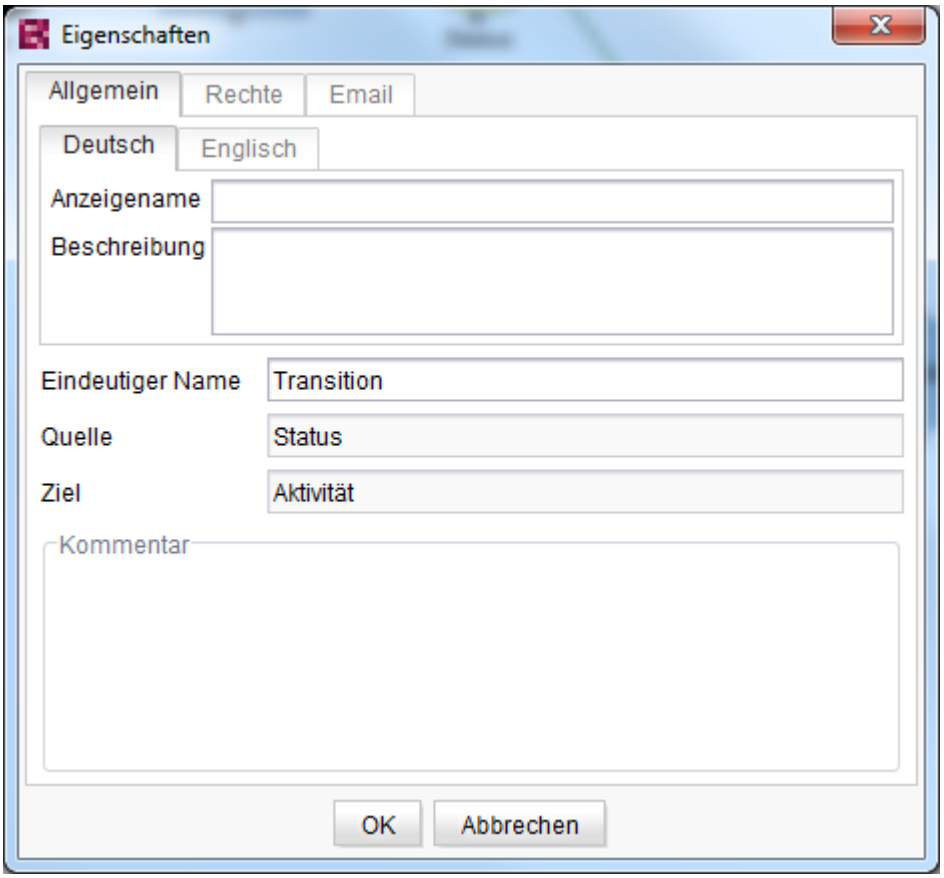


Abbildung 4-30: Eigenschaften einer Transition (Allgemein)

Eindeutiger Name: In diesem Feld kann ein Name für die ausgewählte Transition vergeben werden. Dieser Name muss in Bezug auf seine Quelle eindeutig sein (Zeichenbeschränkung: <= 40 Zeichen).

Quelle: In diesem Feld wird automatisch die Quelle angezeigt, von der die Transition ausgeht.

Ziel: In diesem Feld wird automatisch das Ziel angezeigt, auf das die Transition zeigt.

Kommentar: In diesem Feld kann ein erklärender Kommentar zu dem aktuellen Übergang angegeben werden.

Über die Felder **Anzeigename** und **Beschreibung** können zusätzlich sprachabhängige Anzeigenamen und Beschreibungen hinzugefügt werden. Dabei

handelt es sich um die Redaktionssprachen (nicht die Projektsprachen). Redaktionssprachen werden vom Projektadministrator für ein Projekt festgelegt und können anschließend über das Menü "Ansicht – Bevorzugte Anzeigesprache" vom Redakteur gewechselt werden. Der Anzeigename wird z. B. in Arbeitsablauf-Dialogen (Beschriftung der Buttons im Transitionsdialog, Hilfe- und Historie-Register), als Einträge im Kontextmenü zum Starten/Schalten der Arbeitsabläufe, die Beschreibung als Tooltip und auf dem Hilfe-Register verwendet. Wird kein Anzeigename angegeben, wird der eindeutige Name angezeigt. Ist keine Beschreibung vorhanden, wird der Text aus dem Feld **Kommentar** angezeigt.

4.5.5.2 Register Rechte

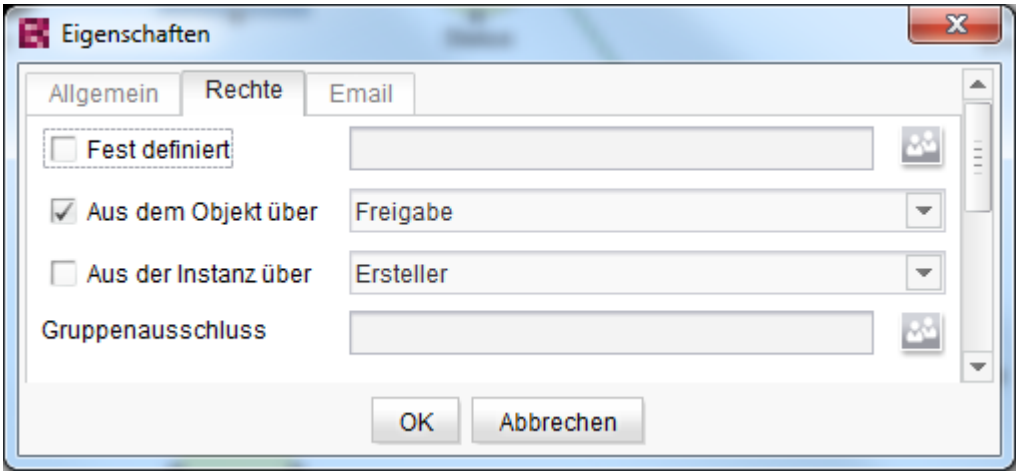


Abbildung 4-31: Eigenschaften einer Transition (Rechte)

Fest definiert: Wird diese Option gewählt, dann sind die berechtigten Benutzer für diese Transition fest definiert. In dem Feld sind die verantwortlichen Benutzer und/oder Gruppen aufgelistet, die diese Transition schalten dürfen. Durch einen Klick auf das Symbol hinter diesem Feld öffnet sich ein weiteres Fenster, in dem die Verantwortlichen aus einer Liste der Projektgruppen bzw. -benutzer ausgewählt werden können.

Aus dem Objekt über: Wird diese Option gewählt, dann ergeben sich die berechtigten Benutzer aus der Rechtedefinition in der Baumstruktur des FirstSpirit-Clients. In dem Feld kann ausgewählt werden, über welches Recht der Benutzer auf dem betrachteten Objekt verfügen muss, um diesen Übergang durchführen zu dürfen.

Aus der Instanz über: Wird diese Option gewählt, dann ergeben sich die berechtigten Benutzer aus der laufenden Instanz des Arbeitsablaufs. In dem Feld kann der Ersteller der Instanz oder der letzte Bearbeiter ausgewählt werden. Die Option "letzter Bearbeiter der Zielaktion" steht dabei nur auf ausgehenden

Transitionen eines Status zur Verfügung und kann nur dann eingesetzt werden, wenn der Arbeitsablauf eine Schleife enthält, so dass eine Aktivität mehrmals durchlaufen werden kann, z. B.:

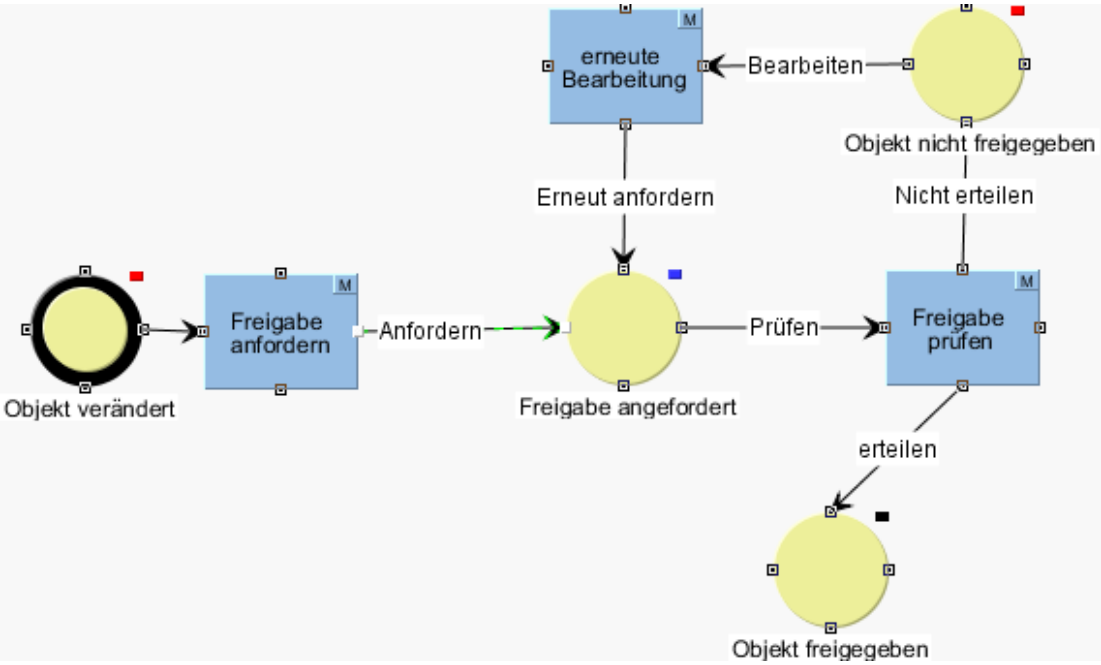


Abbildung 4-32: Standard-Arbeitsablauf Freigabe

Die in Abbildung 4-32 dargestellte Aktivität "Freigabe prüfen" wäre z. B. in diesem Fall eine Zielaktion, d. h. eine Aktivität, auf die ein Status zeigt. Würde die Option "letzter Bearbeiter der Zielaktion" auf der Transition "Prüfen" ausgewählt, kann nur ein Benutzer die betreffende Transition durchführen, der diese Transition bereits zuvor einmal durchgeführt hat.

Gruppenausschluss: Hier können Gruppen ausgewählt werden, die nicht im Feld "Nächster Bearbeiter" eines Arbeitsablaufdialogs "Arbeitsablauf Aktion" erscheinen sollen:

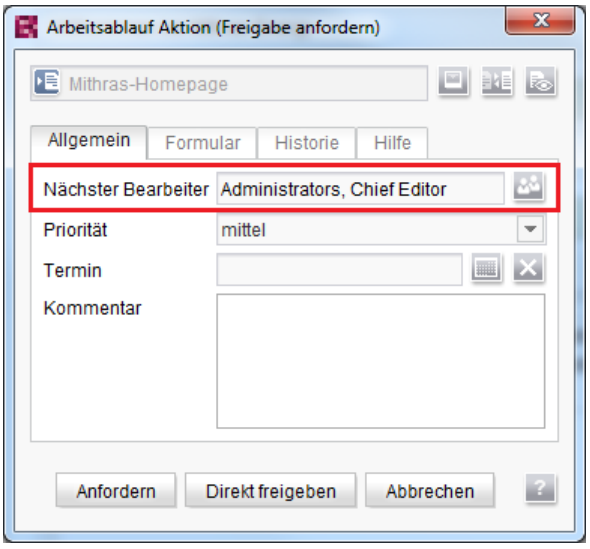


Abbildung 4-33: Vorauswahl "Nächster Bearbeiter"

Die über die Funktion "Gruppenausschluss" ausgewählten Gruppen sind im oben gezeigten Dialog (Abbildung 4-33) aber trotzdem weiterhin über das Icon  auswählbar. Außerdem wirkt sich die Wahl unter "Gruppenausschluss" auch auf den E-Mail-Versand aus.

4.5.5.3 Register Email

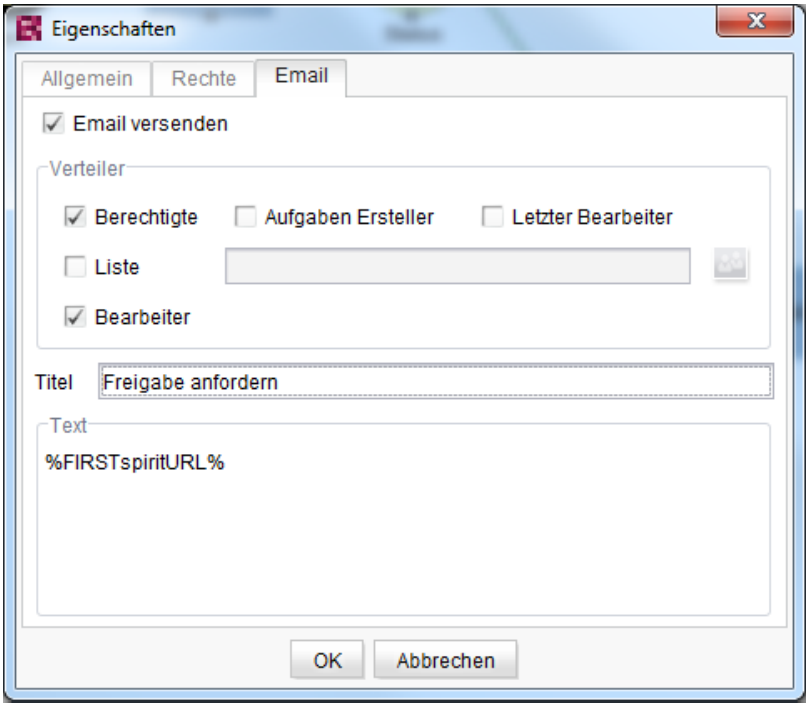


Abbildung 4-34: Eigenschaften einer Transition (Email)

Email versenden: durch Aktivierung dieser Option wird eine E-Mail an ausgewählte Empfänger verschickt, sobald dieser Übergang ausgeführt wurde.

Der Emailversand und die Platzhalter-Ersetzung erfolgt analog zu Beschreibung des Emailversands in Kapitel 4.5.4.2, Seite 156.

4.6 Rechtekonfiguration für Arbeitsabläufe

Rechte zur Ausführung von Arbeitsabläufen sind eine spezielle Art von Redaktionsrechten, die sich nur auf die Arbeitsabläufe innerhalb eines Projekts beziehen.

Die Rechtekonfiguration kann kontextabhängig direkt auf dem Objekt definiert werden, auf dem die Instanz des Arbeitsablaufs gestartet wird oder allgemeingültig innerhalb des Arbeitsablauf-Modells in der Vorlagen-Verwaltung:

- *Allgemeine Rechtekonfiguration* zum Starten und Schalten eines Arbeitsablaufs in der Vorlagen-Verwaltung (für alle Instanzen) (siehe Kapitel 4.6.1 Seite 163)
- *Kontextabhängige Rechtevergabe* für das Starten eines Arbeitsablaufs auf einzelnen Objekten, Teilbäumen und Verwaltungsbereichen (für einzelne Instanzen – abhängig vom Objekt, auf dem der Arbeitsablauf gestartet wird) (siehe Kapitel 4.6.3 Seite 167)
- *Kontextabhängige Rechtevergabe* für das Schalten einzelner Transitionen eines Arbeitsablaufs ("Sonderrechte") auf Objekten, Teilbäumen und Verwaltungsbereichen (für einzelne Instanzen – abhängig vom Objekt, auf dem der Arbeitsablauf gestartet wird) (siehe Kapitel 4.6.4 Seite 171)

Neben der eigentlichen Rechtekonfiguration können die berechtigten Bearbeiter eines Arbeitsablaufs (Instanz) durch den Redakteur (beim Bearbeiten einer Aktivität) eingeschränkt werden, sofern das vom Vorlagen-Entwickler des Arbeitsablaufs konfiguriert wurde (siehe Kapitel 4.6.2 Seite 164).

Die Auswirkungen der Rechtedefinition im JavaClient sind anhand eines Beispiels in Kapitel 4.6.5 (Seite 172 ff.) beschrieben.

4.6.1 Allg. Rechtekonfiguration über die Vorlagen-Verwaltung

Innerhalb der Vorlagen-Verwaltung erfolgt die allgemeine Rechtekonfiguration zum Starten bzw. Schalten eines Arbeitsablaufs über die Rechtevergabe an den einzelnen Übergängen (Transitionen). Dadurch wird gewährleistet, dass jede einzelne Aktivität nur von berechtigten Benutzern durchgeführt werden kann. Der Eigenschaften-Dialog öffnet sich bei einem Doppelklick auf die Transition im Modell

des Arbeitsablaufs. Im Register "Rechte" können die Rechte für das Schalten der Transition vergeben werden (siehe Kapitel 4.5.5.2 Seite 160).

Überschreiben der Transitionsrechte: Die Rechte, die innerhalb des Arbeitsablauf-Modells definiert werden, werden für alle Instanzen des Arbeitsablaufs ausgewertet. Es können auf diese Weise also allgemeingültige Rechtekonfigurationen für den Arbeitsablauf definiert werden. Diese Rechte können jedoch für (kontextabhängige) Arbeitsabläufe überschrieben werden. Das Überschreiben der Transitionsrechte für einzelne Objekte, Teilbäume oder Verwaltungsbereiche ist über den Dialog "Rechtevergabe" auf den jeweiligen Objekten möglich (siehe Kapitel 4.6.3 Seite 167 und Kapitel 4.6.4 Seite 171).

Kontextabhängige Transitionsrechte: Neben den fest definierten Rechten einer Gruppe bzw. eines Benutzers zum Schalten einer Transition können die Transitionsrechte innerhalb der Vorlagen-Verwaltung auch kontextabhängig vergeben werden. In diesem Fall müssen die Transitionsrechte "aus der Instanz über" gewählt werden (siehe Kapitel 4.5.5.2 Seite 160). Wird hier beispielsweise der "letzte Bearbeiter" ausgewählt, so erhält automatisch nur der Bearbeiter das Recht zum Schalten der Transition, der die Instanz des Arbeitsablaufs in den aktuellen Status geschaltet hat.

Verknüpfung mit den Redaktionsrechten: Neben der Möglichkeit, die Rechte kontextabhängig aus der Instanz des Arbeitsablaufs zu ermitteln (siehe oben), können zusätzlich (ebenfalls kontextabhängig) die Redaktionsrechte mit den Transitionsrechten verknüpft werden. In diesem Fall müssen die Transitionsrechte "aus dem Objekt über" gewählt werden (siehe Kapitel 4.5.5.2 Seite 160). Wird hier beispielsweise das Redaktionsrecht "Freigabe" ausgewählt, so erhält automatisch nur der Bearbeiter das Recht zum Schalten der Transition, der auf dem Objekt, auf dem die Instanz des Arbeitsablaufs gestartet wurde, das Recht zur "Freigabe" besitzt.

4.6.2 Ändern bzw. Sperren der Bearbeiter-Vorauswahl

Die Vorauswahl der berechtigten "Bearbeiter" wird dem ausführenden Redakteur des Arbeitsablaufs innerhalb des Aktivitätsdialogs im Feld "Bearbeiter" angezeigt. "Bearbeiter" sind alle Gruppen bzw. Benutzer, die das Recht zum Schalten der zukünftigen Transitionen des Arbeitsablaufs haben. Dabei werden die Rechte berücksichtigt, die auf den ausgehenden Transitionen der zukünftigen Status definiert wurden (siehe Kapitel 4.6.1 Seite 163).

Beispiel (Arbeitsablauf-Modell):

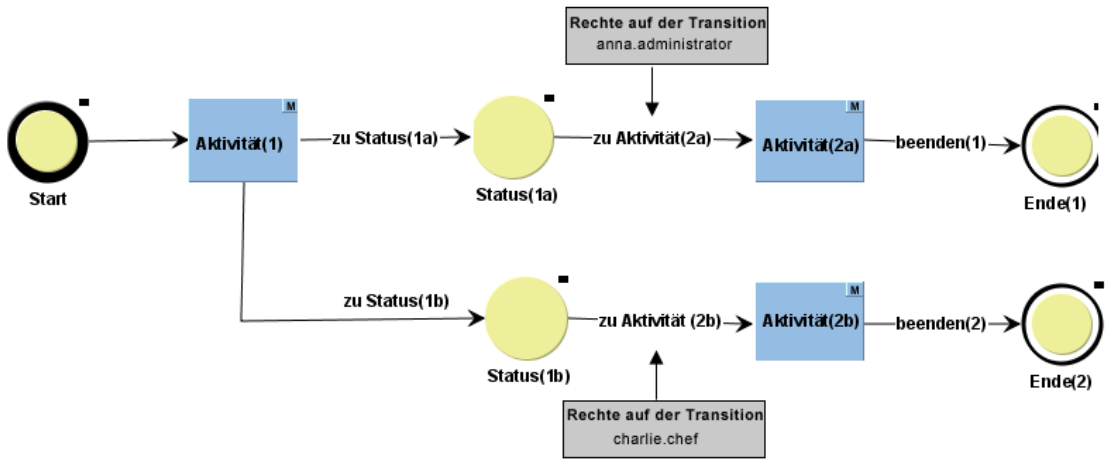


Abbildung 4-35: Arbeitsablauf-Modell mit Rechtekonfiguration (grau hinterlegt)

Beispiel - Beschreibung (siehe Abbildung 4-35):

- Die Rechte auf der Transition "zu Aktivität (2a)" wurden fest definiert für den Benutzer "anna.administrator".
- Die Rechte auf der Transition "zu Aktivität (2b)" wurden fest definiert für den Benutzer "charlie.chef".

Der Arbeitsablauf wird nun vom Redakteur gestartet. Der Aktivitätsdialog "Aktivität(1)" öffnet sich (siehe Abbildung 4-36). Der Redakteur kann dort zwischen den beiden Status "Status(1a)" und "Status(1b)" wählen. Im Feld "Bearbeiter" werden die zukünftigen Bearbeiter des Arbeitsablaufs automatisch aufgelistet. Nach dem Weiterschalten der aktuellen "Aktivität(1)" befindet sich der Arbeitsablauf entweder im "Status(1a)" oder im "Status(1b)". Zukünftige "Bearbeiter" können daher nur Gruppen bzw. Benutzer sein, die Rechte auf den ausgehenden Transitionen dieser beiden Status besitzen. Im Beispiel werden also die "Bearbeiter" angezeigt, die Rechte zum Schalten der Transitionen "zu Aktivität(2a)" und zum Schalten der Transitionen "zu Aktivität(2b)" besitzen.

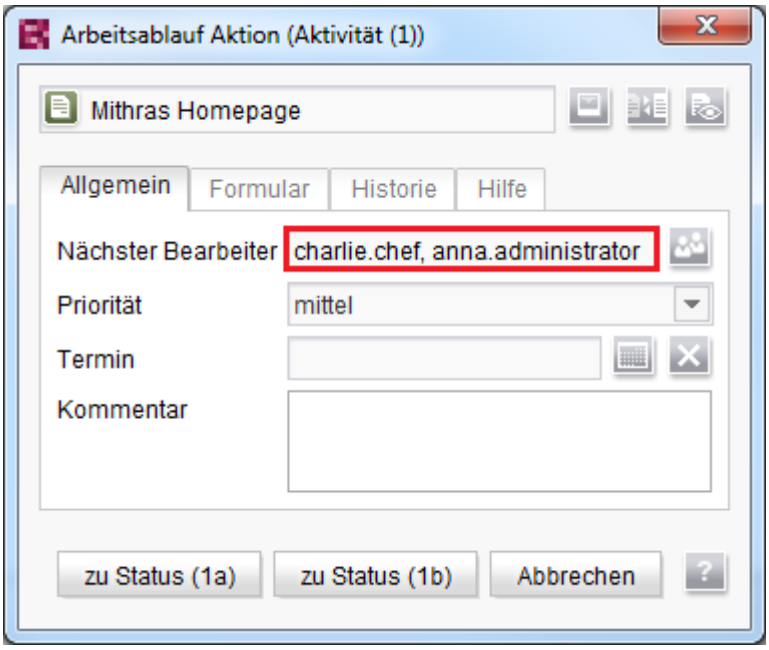


Abbildung 4-36:Beispiel – Arbeitsablauf Aktion "Aktivität(1)"

Diese Vorauswahl der zukünftig möglichen Bearbeiter kann durch den Redakteur geändert werden. Dazu muss vom Vorlagenentwickler auf dem Startzustand des Arbeitsablaufs in der Vorlagen-Verwaltung der Konfigurationsdialog des Startstatus geöffnet werden (siehe Kapitel 4.5.3.1 Seite 151).

Im Register "Allgemein" kann zwischen zwei Optionen gewählt werden:

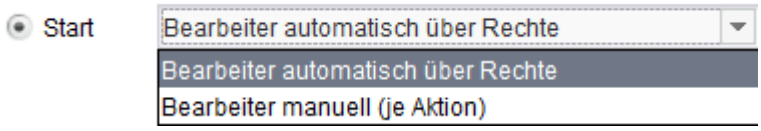


Abbildung 4-37: Rechtekonfiguration für den Startstatus

- Bearbeiter manuell (je Aktion) (siehe Kapitel 4.6.2.1 Seite 166)
- Bearbeiter automatisch über Rechte (siehe Kapitel 4.6.2.2 Seite 167)

4.6.2.1 Bearbeiter manuell (je Aktion)

Im Feld "Bearbeiter" werden die Rechte ausgewertet, die innerhalb des Arbeitsablaufes (auf den ausgehenden Transitionen der zukünftigen Status) definiert wurden. Wird die Option "Bearbeiter manuell (je Aktion)" ausgewählt, können diese Bearbeiter vom Redakteur *geändert* werden. Der Button zur Gruppen- bzw. Benutzerauswahl im Dialog "Arbeitsablauf Aktion", der beim Starten bzw. Schalten des Arbeitsablauf (Instanz) angezeigt wird, ist dann *aktiv*.

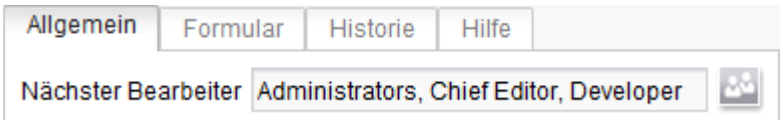


Abbildung 4-38: Arbeitsablauf Aktion – Bearbeiter manuell über Rechte

Mit einem Klick auf den Button  öffnet sich der Dialog zur Gruppen- und Benutzerauswahl mit allen berechtigten Bearbeitern. Diese Auswahl kann nun vom Redakteur auf die gewünschten Benutzer eingeschränkt werden.

Für den Redakteur ist nur eine Einschränkung der Bearbeiter möglich. Soll die Liste der Bearbeiter erweitert werden, so müssen dazu die Rechte auf den Transitionen des Arbeitsablauf-Modells vom Vorlagen-Entwickler angepasst werden.

4.6.2.2 Bearbeiter automatisch über Rechte

Im Feld "Bearbeiter" werden die Rechte ausgewertet, die innerhalb des Arbeitsablaufes (auf den ausgehenden Transitionen der zukünftigen Status) definiert wurden. Wird die Option "Bearbeiter automatisch über Rechte" ausgewählt, können diese Bearbeiter vom Redakteur *nicht geändert* werden. Der Button zur Gruppen- bzw. Benutzerauswahl im Dialog "Arbeitsablauf Aktion", der beim Starten bzw. Schalten des Arbeitsablauf (Instanz) angezeigt wird, ist dann *inaktiv*.

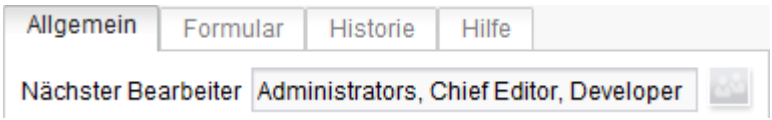


Abbildung 4-39: Arbeitsablauf Aktion – Bearbeiter automatisch über Rechte

4.6.3 Kontextabhängige Rechte zum Starten eines Arbeitsablaufs

Die kontextabhängige Rechtevergabe erfolgt im FirstSpirit JavaClient. Hier können alle Bereiche des Projekts mit Redaktionsrechten für bestimmte Gruppen bzw. Benutzer versehen werden. Dabei ist eine detaillierte Rechtevergabe für jedes Objekt möglich, also beispielsweise für eine einzelne Seite in der Inhalte-Verwaltung. Diese Rechte können innerhalb der einzelnen Verwaltungsbereiche hierarchisch vererbt werden.

Die Rechte zur Ausführung von Arbeitsabläufen werden parallel zu den Redaktionsrechten für Gruppen und Benutzer über den Dialog "Rechtevergabe"

vergeben. Der Dialog "Rechtevergabe" wird über das Kontextmenü "Extras – Rechte ändern" auf dem gewünschten Objekt innerhalb der Baumstruktur des JavaClient geöffnet. Zusätzlich zur allgemeinen "Rechtevergabe" der Redaktionsrechte existiert das Register "Arbeitsablauf Rechte":

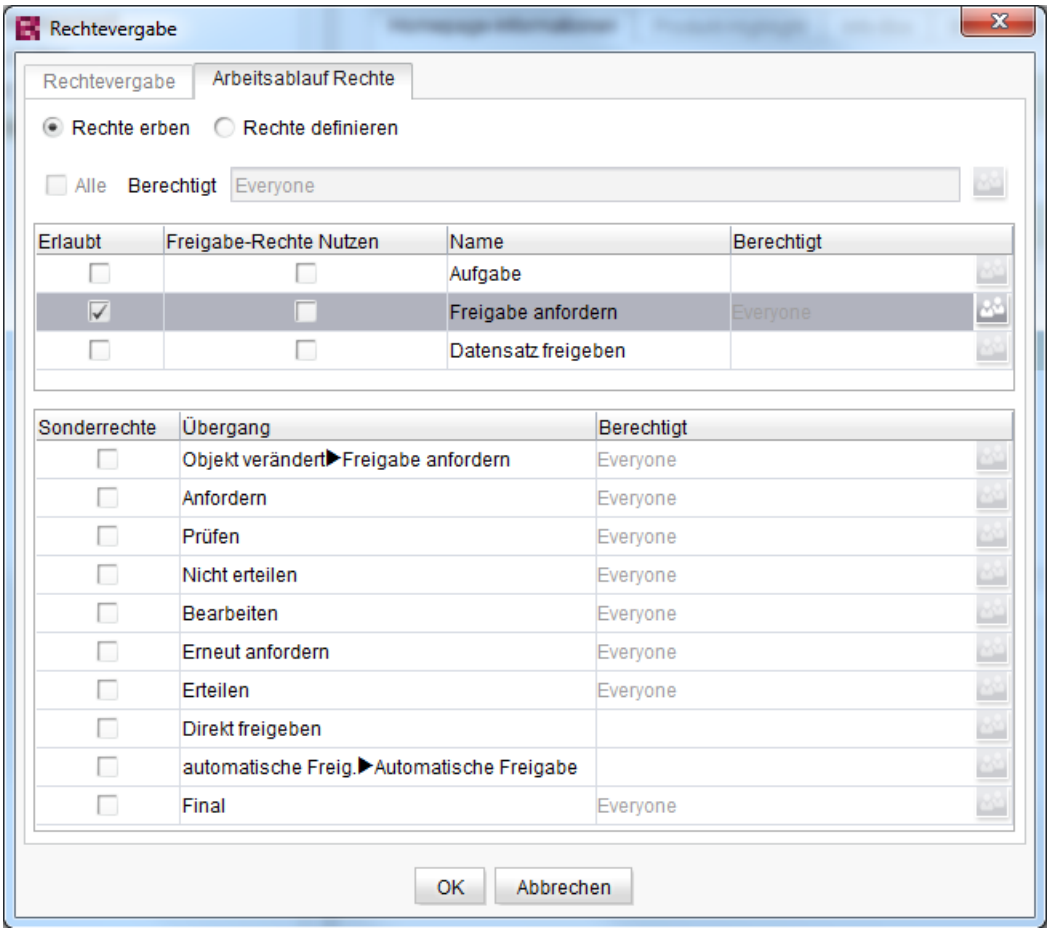


Abbildung 4-40: Kontextabhängige Arbeitsablauf-Rechte

Rechte erben: Der Radiobutton "Rechte erben" ist standardmäßig ausgewählt (Ausnahme Wurzelknoten). Bei dieser Einstellung werden die Rechte "Arbeitsablauf Rechte" aus einem übergeordneten Knoten vererbt.

Rechte definieren: Wird der Radiobutton "Rechte definieren" aktiviert, können auf dem Knoten Rechte für die Arbeitsabläufe definiert werden. Im ersten Schritt öffnet sich das Fenster zur Übernahme der geerbten Rechte.

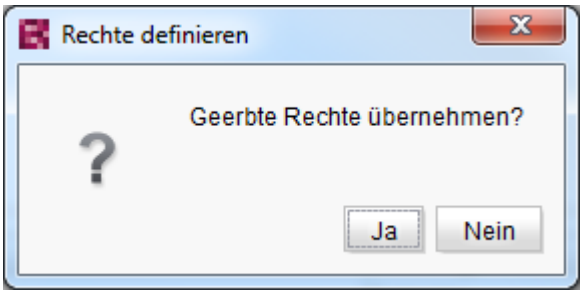


Abbildung 4-41: Geerbte Rechte übernehmen?

Werden die Rechte aus einem übergeordneten Knoten übernommen, dann werden die geerbten Rechte in die Tabelle der Arbeitsabläufe übernommen. Wird der Dialog dagegen mit "Nein" bestätigt, werden die Arbeitsablauf-Rechte zurückgesetzt. Die Tabellenansicht ist unabhängig von der Auswahl jetzt aktiv, das heißt, der Benutzer kann eigene Rechte definieren.

Alle: Wird die Checkbox "Alle" aktiviert, können auf dem aktuellen Knoten und allen hierarchisch untergeordneten Knoten der Baumstruktur alle Arbeitsabläufe vom "berechtigten" Benutzer gestartet werden. Die beiden darunter liegenden Tabellen können in diesem Fall nicht bearbeitet werden, und alle darin vorgenommenen Einstellungen sind ohne Bedeutung. Wird die Checkbox nicht aktiviert, müssen die Einstellungen für jeden Arbeitsablauf einzeln festgelegt werden.

Berechtigt: In diesem Feld sind alle Benutzer und/oder Gruppen aufgelistet, die auf dem aktuellen Knoten einen Arbeitsablauf aufrufen dürfen. Bei einem Klick auf das Icon  öffnet sich das Fenster "Gruppen/Benutzer auswählen". Es werden alle Gruppen und Benutzer des Projekts aufgelistet. Über das Fenster kann eine Auswahl der berechtigten Gruppen und einzelner Benutzer erfolgen.

In der oberen Tabelle (vgl. Abbildung 4-40) werden alle Arbeitsabläufe des Projekts aufgelistet. Sollen für einen Teilbaum nur ausgewählte Arbeitsabläufe zugelassen werden, dann kann bei der Rechtedefinition eine Liste von Arbeitsabläufen erstellt werden, die durch ausgewählte Benutzer gestartet werden können. Dabei kann bei Bedarf für jeden Arbeitsablauf ein anderer Benutzer bestimmt werden.

Die Eingabemöglichkeiten dieser Tabelle sind nur aktiv, wenn die Checkbox "Alle" deaktiviert ist. In diesem Fall kann das Recht zum Starten eines Arbeitsablaufs für einzelne Arbeitsabläufe erteilt oder unterbunden werden:

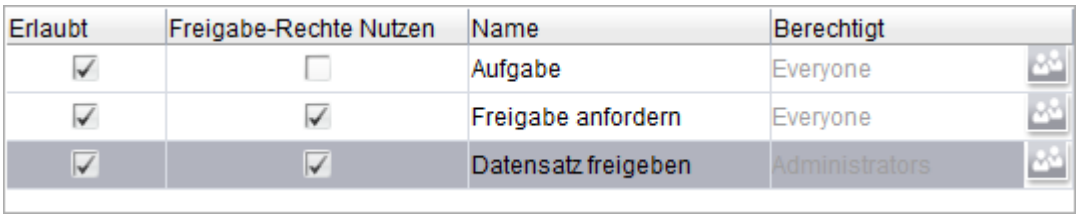


Abbildung 4-42: Kontextabhängige Rechte zum Starten einzelner Arbeitsabläufe

Erlaubt: Wird die Checkbox "Erlaubt" aktiviert, dürfen alle berechtigten Benutzer (siehe Spalte "Berechtigte") den Arbeitsablauf starten. Die Erlaubnis wird für den aktuellen Knoten und alle hierarchisch untergeordneten Knoten der Baumstruktur erteilt.

Freigabe-Rechte-Nutzen: Wird die Checkbox "Freigabe-Rechte-Nutzen" aktiviert, werden die Freigaberechte, die im Register "Rechtevergabe" definiert wurden, für jeden Benutzer ausgewertet. Achtung: Wird die Checkbox nicht aktiviert, können Widersprüche bei der Rechtedefinition auftreten. Hat ein Benutzer beispielsweise *kein* Recht zur Freigabe auf einem bestimmten Objekt, wird aber im Standard-Arbeitsablauf "Freigabe Anfordern" als Berechtigter geführt, kann es zu einer Konfliktsituation kommen. Die Freigabe wird zwar auch in so einem Fall durch das System unterbunden, für den Benutzer ist das Verhalten (keine Freigabe) aber nicht ersichtlich, da der Arbeitsablauf bis zum Status "Freigabe erteilen", wie definiert, durchgeschaltet werden kann. Wird hingegen die Checkbox "Freigabe-Rechte-Nutzen" aktiviert, so werden die Freigaberechte des Benutzers bei jeder Transition des Arbeitsablaufs ausgewertet. Werden dann Widersprüche zwischen den Redaktionsrechten (kein Recht zur Freigabe) und den Rechten im Arbeitsablauf (z. B. Freigabe erteilen) festgestellt, werden diese Transitionen für den "nicht berechtigten" Benutzer ausgeblendet. Der Benutzer kann in diesem Fall zwar die "Freigabe anfordern" also den Arbeitsablauf starten, das Objekt aber nicht mehr in den nachfolgenden Status "Objekt freigegeben" weiterschalten. Die dazu erforderliche Transition wird ausgeblendet.

Name: In dieser Spalte befindet sich der Name des Arbeitsablaufs.

Berechtigt: In diesem Feld sind alle Benutzer und/oder Gruppen aufgelistet, die auf dem aktuellen Knoten einen Arbeitsablauf starten dürfen. Bei einem Klick auf das Icon  öffnet sich das Fenster "Gruppen/Benutzer auswählen". Es werden alle Gruppen und Benutzer des Projekts aufgelistet. Über das Fenster kann eine Auswahl der berechtigten Gruppen bzw. einzelner Benutzer erfolgen.

Weiterführende Informationen zu Redaktionsrechten siehe FirstSpirit Handbuch für Redakteure, Kapitel 13.

4.6.4 Kontextabhängige Rechte zum Schalten eines Arbeitsablaufs

Die bisherigen kontextabhängigen Rechtevergaben aus Kapitel 4.6.3 (Seite 167 ff.) beziehen sich lediglich auf das Recht zum Starten eines Arbeitsablaufs. Es können jedoch auch so genannte kontextabhängigen "Sonderrechte" für die einzelnen Transitionen des Arbeitsablaufs definiert werden.

Soll in einem einzelnen Knoten eine bestimmte Aktivität durch einen anderen Benutzer durchgeführt werden, kann dies über die kontextabhängigen Sonderrechte definiert werden. Dazu muss zunächst der gewünschte Arbeitsablauf in der oberen Tabelle selektiert werden (siehe Abbildung 4-42). Alle Transitionen des markierten Arbeitsablaufs werden dann in der unteren Tabelle zur Definition von Sonderrechten aufgelistet (siehe Abbildung 4-43). Für den gewünschten Übergang des Arbeitsablaufes kann dann berechtigter Benutzer bestimmt werden.

Sind bereits innerhalb der Vorlagen-Verwaltung Rechte für das Schalten der Transition über das Arbeitsablauf-Modell definiert worden (siehe Kapitel 4.6.1 Seite 163), überschreibt diese Definition (kontextabhängige "Sonderrechte") die vorhandenen Berechtigungen (aus dem Arbeitsablauf-Modell).

Sonderrechte	Übergang	Berechtigt
☐	Objekt verändert▶Freigabe anfordern	Everyone
☐	Anfordern	Everyone
☐	Prüfen	Everyone
☑	Nicht erteilen	Chief Editor
☐	Bearbeiten	Everyone
☐	Erneut anfordern	Everyone
☐	Erteilen	Everyone
☑	Direkt freigeben	Administrators

Abbildung 4-43: Kontextabhängigen Sonderrechte zum Schalten einer Transition

Sonderrechte: Wird das Häkchen in dieser Spalte gesetzt, dann werden die im Arbeitsablauf vergebenen Rechte für diesen Übergang auf diesem Knoten ignoriert. Stattdessen gelten für diesen Übergang die Berechtigungen, die an dieser Stelle in

der Spalte Berechtigungen aufgelistet werden.

Übergang: In dieser Spalte sind die Namen der Übergänge aufgelistet. Wurde im Arbeitsablauf für einen Übergang kein Name vergeben, erscheinen hier die Namen von Quelle und Ziel des Übergangs.

Berechtigt: In diesem Feld sind alle Benutzer und/oder Gruppen aufgelistet, die diesen Übergang ausführen dürfen. Die hier aufgelisteten Transitionsrechte werden aus dem Arbeitsablauf-Modell übernommen (siehe Kapitel 4.5.5.2 Seite 160), werden aber beim Aktivieren der Checkbox "Sonderrechte" überschrieben (Standardeinstellung: Gruppe "Everyone").

Bei einem Klick auf das Icon  öffnet sich das Fenster "Gruppen/Benutzer auswählen". Es werden alle Gruppen und Benutzer des Projekts aufgelistet. Über das Fenster kann eine Auswahl der berechtigten Gruppen bzw. Benutzer erfolgen.

4.6.5 Auswirkungen der Rechtekonfiguration

Transitionsrechte werden entweder allgemein über das Arbeitsablauf-Modell definiert (siehe Kapitel 4.6.1 Seite 163) oder kontextabhängig für einzelne Objekte oder Teilbäume (siehe Kapitel 4.6.3 Seite 167 und Kapitel 4.6.4 Seite 171).

Die Auswirkungen sind für beide Rechtedefinitionen identisch:

Übergänge, die zu einer Aktivität führen, berechtigen den Benutzer dazu, diese Aktivität über das Kontextmenü des entsprechenden Objekts aufzurufen und durchzuführen.

Übergänge, die zu einem Zustand führen, berechtigen den Benutzer dazu, diesen Zustand im Aktivitätsdialog zu schalten.

Beispiel (Arbeitsablauf-Modell):

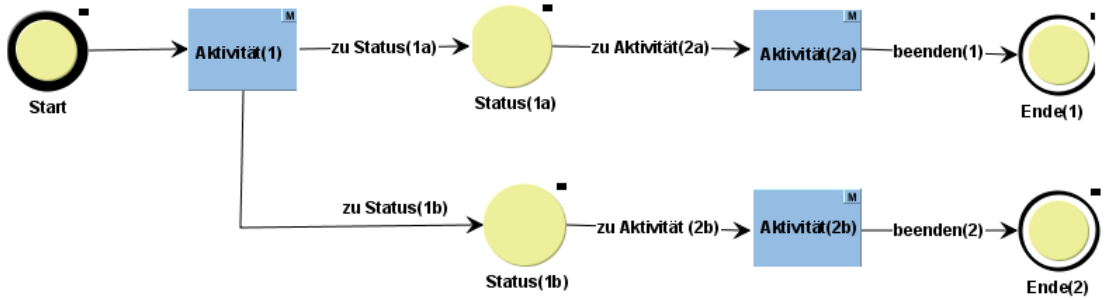


Abbildung 4-44: Beispiel Arbeitsablauf-Modell

Beispiel-Rechtedefinition (definiert über das Arbeitsablauf-Modell):

Sonderrechte	Übergang	Berechtigt
☐	Start▶Aktivität(1)	Editor
☐	zu Status(1a)	Editor
☐	zu Status(1b)	Administrators
☐	zu Aktivität(2a)	Everyone
☐	zu Aktivität(2b)	Chief Editor
☐	beendet(1)	Everyone
☐	beendet(2)	Chief Editor

Abbildung 4-45 Beispiel Rechtedefinition über das Modell

Beispiel: Auswirkungen der Transitionsrechte:

1. Starten des Arbeitsablaufs über das Kontextmenü: Die Gruppe Redakteure kann das Kontextmenü öffnen und den Arbeitsablauf starten:

Abbildung 4-46: Starten des Arbeitsablaufs über das Kontextmenü

2. Der Dialog "Aktivität(1)" bietet die Möglichkeit zum Schalten des Arbeitsablaufs in den "Status(1a)" oder in den "Status(1b)". Der Button zum Schalten von "Status (1a)" wird nur für die Gruppe "Redakteure" eingeblendet (siehe Abbildung 4-47), der Button zum Schalten von "Status (1b)" wird nur für die Gruppe "Administrators" eingeblendet (siehe Abbildung 4-48).

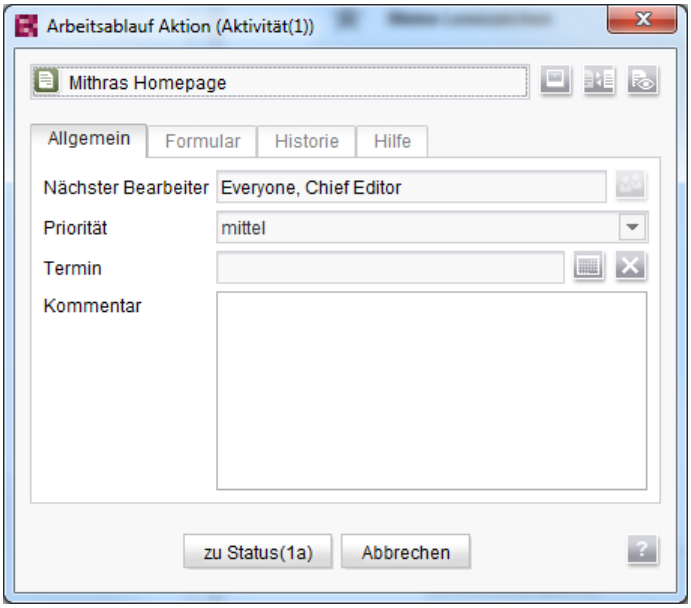


Abbildung 4-47: Aktivitätsdialog zum Schalten der Transition "zu Status(1a)"

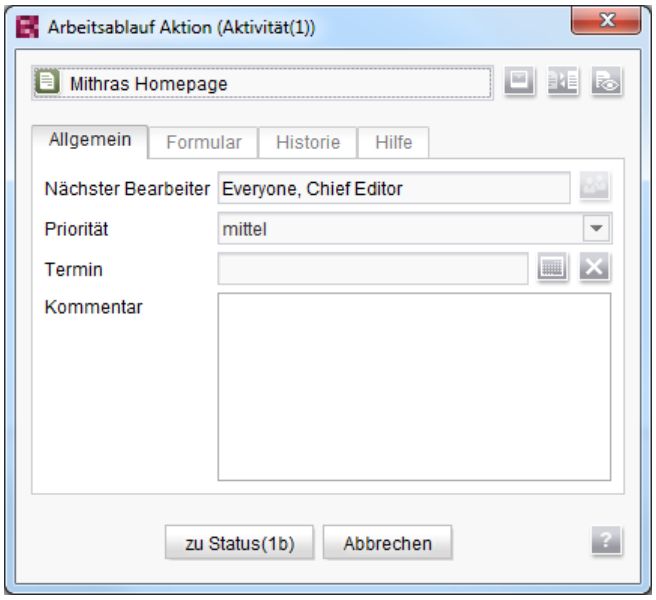


Abbildung 4-48: Aktivitätsdialog zum Schalten der Transition "zu Status(1b)"

3. Befindet sich die Instanz des Arbeitsablaufs im Status(1a), darf jeder Benutzer über das Kontextmenü die nachfolgenden Transition "zu Aktivität (2a)" aufrufen. Der Dialog "Aktivität (2a)" wird angezeigt.

Abbildung 4-49: Schalten des Arbeitsablaufs über das Kontextmenü

4. Befindet sich die Instanz des Arbeitsablaufs im Status(1b), darf der Benutzer „ChiefEditor“ über das Kontextmenü die nachfolgenden Transition "zu Aktivität (2b)" aufrufen. Der Dialog "Aktivität (2b)" wird angezeigt.

Abbildung 4-50: Schalten des Arbeitsablaufs über das Kontextmenü

5. Der Dialog "Aktivität(2a)" bietet die Möglichkeit zum Beenden des Arbeitsablaufs in den Status "Ende(1)". Der Button zum Schalten von "beenden(1)" wird für alle Benutzer eingeblendet.
   (Das Feld mit den zukünftigen "Bearbeitern" ist in diesem Fall leer, da es sich um die letzte Transition handelt (siehe Kapitel 4.6.2 Seite 164)).

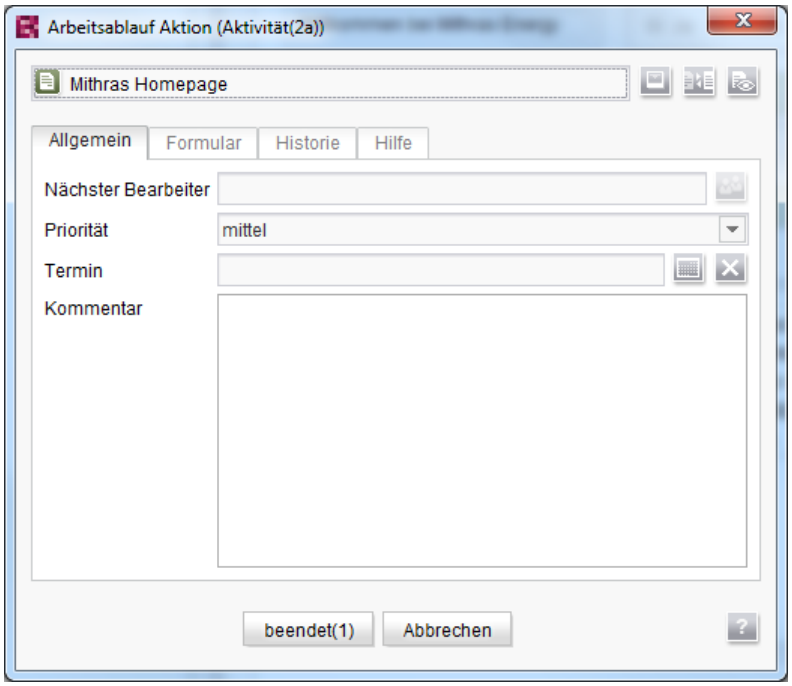


Abbildung 4-51: Aktivitätsdialog zum Schalten der Transition "beenden(1)"

6. Der Dialog "Aktivität(2b)" bietet die Möglichkeit zum Beenden des Arbeitsablaufs in den Status "Ende(2)". Der Button zum Schalten von "beenden(2)" wird nur für den Benutzer „ChiefEditor“ eingeblendet.
(Das Feld mit den zukünftigen "Bearbeitern" ist in diesem Fall leer, da es sich um die letzte Transition handelt (siehe Kapitel 4.6.2 Seite 164)).

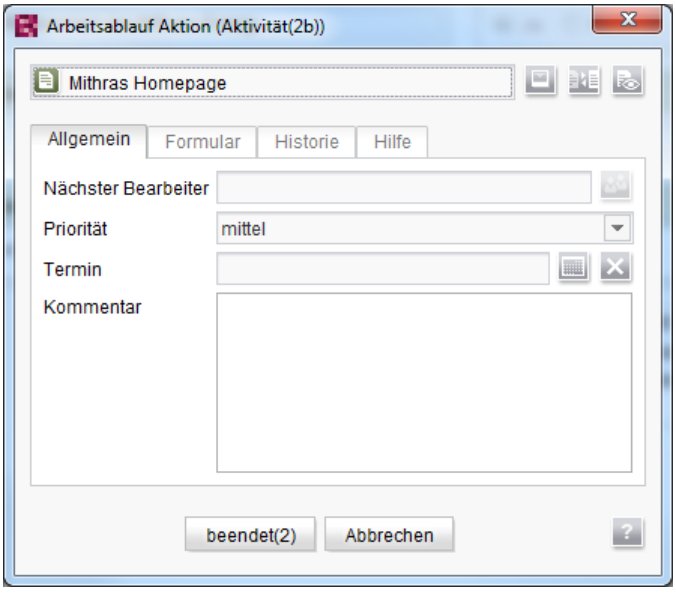


Abbildung 4-52: Aktivitätsdialog zum Schalten der Transition "beenden(2)"

Besitzt ein Benutzer bzw. eine Gruppe das Recht eine Transition zu schalten, die zu einem Aktivitätsdialog führt, so sollte diese Gruppe bzw. dieser Benutzer außerdem das Recht zum Schalten der Transition in mindestens einen nachfolgenden Status besitzen. Andernfalls enthält der Aktivitätsdialog lediglich einen Button zum Abrechen der Aktion. In diesem Fall sollten die Rechte überprüft und ggf. neu definiert werden.

4.7 Schreibschutz innerhalb von Arbeitsabläufen

4.7.1 Allgemein

Beim Starten eines (kontextgebundenen) Arbeitsablaufs, kann das Element, auf dem der Arbeitsablauf gestartet wurde, mit einem Schreibschutz versehen werden (siehe Kapitel 4.5.3.1 Seite 151). Dieser Schreibschutz soll verhindern, dass ein Element von einem anderen Bearbeiter verändert wird, während ein Arbeitsablauf läuft.

Schreibschutz durch laufende Instanzen eines Arbeitsablaufs:

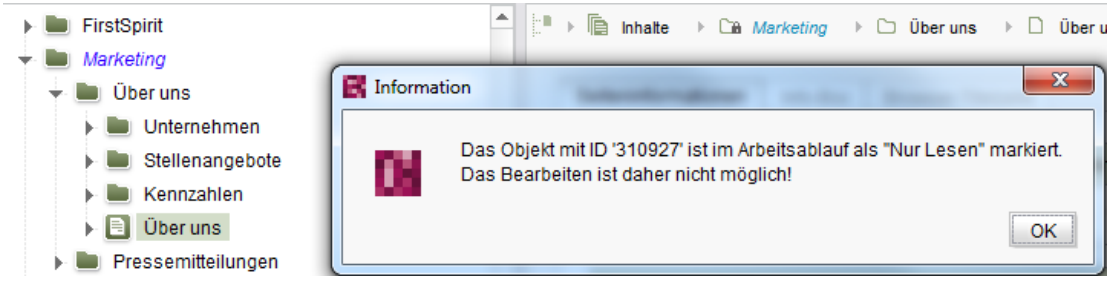


Abbildung 4-53: Schreibschutz auf untergeordneten Objekten

Der Schreibschutz wirkt sich sowohl auf das aktuelle Objekt als auch auf alle untergeordneten Objekte der laufenden Instanz des Arbeitsablaufs aus. Im Beispiel aus Abbildung 4-53 ist auf dem Ordner "Marketing" durch einen laufenden Arbeitsablauf ein Schreibschutz gesetzt. Versucht ein weiterer Benutzer, diesen Ordner zum Bearbeiten zu sperren, erhält er die Information, dass das Element aktuell nicht bearbeitet werden kann. Die gleiche Meldung erscheint auch, wenn der Benutzer versucht den Ordner "Unternehmen" oder ein beliebiges Objekt unterhalb des Ordners "Marketing" zu sperren.

Der Schreibschutz wird unabhängig davon gesetzt, ob der Arbeitsablauf ein Skript verwendet und welche Aktionen auf dem betreffenden Element ausgeführt werden.

4.7.2 Schreibschutz beim Anlegen und Verschieben

Innerhalb des FirstSpirit JavaClients können einige Aktionen ausgeführt werden, ohne dass sich das Objekt im Bearbeitungsmodus befindet. Diese Änderung des Bearbeitungskonzepts soll dafür sorgen, dass auch in großen Projekten ein paralleles Arbeiten (mit vielen Benutzern) möglichst reibungslos funktioniert. So wird beispielsweise beim Bearbeiten eines Elements nicht mehr der gesamte Teilbaum zur Bearbeitung angefordert, sondern nur das Objekt, das aktuell geändert werden soll. Das Anlegen oder Verschieben eines Elements ist daher ebenfalls möglich, ohne zuvor den Schreibschutz auf dem Vaterknoten (Bearbeitungsmodus) anzufordern.

Da es sich bei Arbeitsabläufen jedoch um potentiell kritische Aktionen handelt (z. B. die Freigabe eines Objekts), unterbindet der Schreibschutz eines Arbeitsablaufs auch das Anlegen bzw. Verschieben innerhalb der aktuell laufenden Instanz des Arbeitsablaufs.

Versucht ein Redakteur beispielsweise einen Absatz zu einer Seite hinzuzufügen, auf der aktuell ein Arbeitsablauf gestartet wurde, wird die folgende Fehlermeldung angezeigt:

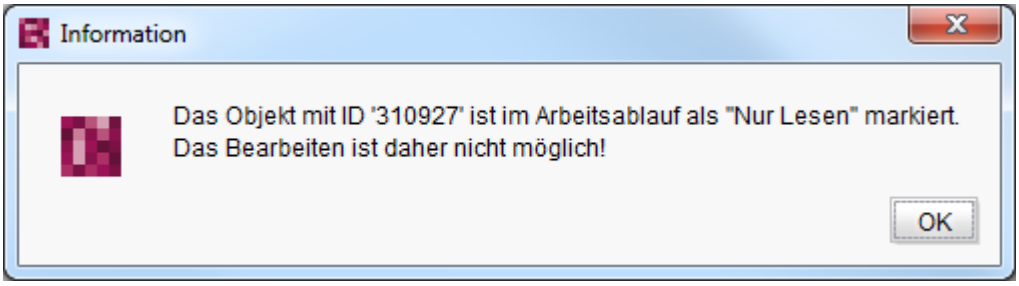


Abbildung 4-54:Schreibschutz auf einer Seite (durch einen Arbeitsablauf)

4.7.3 Schreibschutz innerhalb von Skripten

Für einige Aktionen, die über die Access-API von FirstSpirit ausgeführt werden, ist ein Schreibschutz auf dem betreffenden Element notwendig, z. B.:

- Rekursives Löschen von Elementen im Projekt
- (Rekursive) Freigabe von Elementen im Projekt

Ein Problem, das sich nun in der Praxis stellt, ist das Setzen eines Schreibschutzes in einem Skript (auf einem Element oder Teilbaum – API-Aufruf `setLock(true, false)` bzw. `setLock(true)`), wenn durch das Starten des Arbeitsablaufs bereits ein

Schreibschutz (durch den Arbeitsablauf – siehe Kapitel 4.5.3.1) auf dem Element besteht. Der Schreibschutz des Arbeitsablaufs verhindert in diesem Fall das Setzen des "normalen" Schreibschutzes auf dem Element.

Für einfache Lösch- oder Freigabe-Aktionen innerhalb des Arbeitsablaufs ist das Setzen des Schreibschutzes allerdings nicht notwendig, da das betreffende Element beim Schalten der Transition ja bereits automatisch durch den Arbeitsablauf gesperrt ist.

Anders sieht es aus, wenn das Löschen oder Freigeben rekursiv, das heißt auf einem Teilbaum des Projekts ausgeführt werden soll. In diesem Fall muss ein rekursiver Schreibschutz auf dem kompletten Teilbaum gesetzt werden und das ist nur möglich, wenn der Schreibschutz des Arbeitsablaufs aufgehoben wird. Dazu wird der (automatisch gesetzte) Schreibschutz über den Status des Arbeitsablaufs temporär entfernt und beim Beenden der Lösch- bzw. Freigabe-Option (über das Skript) erneut gesetzt. Das genaue Vorgehen wird anhand eines Beispiels in Kapitel 4.10.1 beschrieben.

4.8 Verwendung von Skripten in Arbeitsabläufen

Skripte stellen ein mächtiges Hilfsmittel für die Umsetzung von kundenspezifischen Wünschen innerhalb der FirstSpirit Arbeitsabläufe dar. Wie in bereits bei der Beschreibung der Elemente des Arbeitsablauf-Editors erwähnt, können Skripte innerhalb von Arbeitsabläufen ausschließlich an Aktivitäten gebunden werden (siehe Kapitel 4.5.4.1 Seite154). Die Ausführung einer Aktivität kann entweder manuell durch einen Benutzer ausgeführt werden oder automatisch durch ein Skript erfolgen (siehe Kapitel 4.2.3.2 Seite 133).

Das Ergebnis einer Aktivität bezieht sich immer auf die Instanz eines Arbeitsablaufs. Es handelt sich entweder um einen Zustandswechsel in einen von der Aktivität aus erreichbaren Nachfolgezustand oder aber um das Beibehalten des aktuellen Zustandes (entspricht der "Abbrechen"-Semantik im Aktivitätsdialog). Dies gilt auch für Skripte, die an diese Aktivität gekoppelt werden. Das Skript muss also "selbst" dafür sorgen, dass ein Transition in den nachfolgenden Zustand durchgeführt wird.

Grundsätzlich können im Arbeitsablauf-Modell die Aktivitäten mit der das Skript verknüpft ist, entweder als "manuell" oder als "automatisch" definiert sein – in beiden Fällen kann es sinnvoll sein, ein Skript einzusetzen.

Werden innerhalb von Arbeitsabläufen Skripte verwendet, findet KEINE automatische Auswertung der Redaktionsrechte (z. B. bei der Freigabe) statt. Diese Rechte müssen innerhalb des Arbeitsablaufs geeignet mit den Transitionsrechten verknüpft werden (siehe Kapitel 4.5.5.2 Seite 160).

4.8.1 automatische Aktivitäten und Skripte

Automatische Aktivitäten erwarten keine Benutzerinteraktion und werden ausgeführt, sobald einer der im Modell vorgelagerten Zustände erreicht wird (d.h. die Aktion wird vom System und nicht vom Benutzer ausgelöst). Eine automatische Aktion (und damit das angekoppelte Skript) werden also direkt nach dem Erreichen eines Zustands ausgeführt.

Durch die Verwendung von automatischen Aktionen können potentiell Endlosschleifen gebaut werden. Diese Situation wird vom FirstSpirit Arbeitsablauf-Interpreter erkannt, die Ausführung der entsprechenden Arbeitsablauf Instanz wird beendet und es erscheint eine Fehlermeldung.

4.8.2 manuelle Aktivitäten und Skripte

In diesem Fall wird die Aktionsausführung von einem Benutzer gestartet. Ist kein Skript vorhanden, so wird dem Benutzer das Standard-Formular für Arbeitsabläufe mit allen ihm erlaubten Übergängen angezeigt ("Aktivitätsdialog"). Sobald der Aktion ein Skript zugeordnet wird, erfolgt diese Dialoganzeige nicht mehr automatisch. Soll dem Benutzer der Aktivitätsdialog angezeigt werden, so muss dies über das Skript ausgeführt werden (siehe Kapitel 4.8.3 Seite 179 und Kapitel 4.8.4 Seite 181).

4.8.3 Arbeitsablauf-Kontext

Der Skriptkontext für Arbeitsabläufe stellt u.a. folgende Methoden zur Verfügung:

```
Transition showActionDialog();
```

Aufgabe: Anzeige des Aktivitätsdialogs (meist nur für "manuelle" Aktionen relevant). Liefert die vom Benutzer ausgewählte Transition als "Transition"-Objekt zurück. Achtung: der eigentliche Übergang wird NICHT durchgeführt (Beispiel siehe Kapitel 4.8.4 Seite 181).

```
void doTransition(firstspirit.workflow.model.Transition transition)
```

Aufgabe: Ausführung der angegebenen Transition. Dies kann z. B. die vom Benutzer ausgewählte sein oder eine andere, in dieser Aktion verfügbare (und erlaubte) Transition. Sollte ein Transition gewählt werden, die nicht erlaubt ist, so kommt es zu einer Fehlermeldung (Beispiel siehe Kapitel 4.8.4 Seite 181).

```
void doTransition(String transitionName)
```

Aufgabe: Ausführung der mit Namen angegeben Transition. Wurde der Transition im Modell kein Name zugeordnet, so wird automatisch ein Name der Form "->"+"Name des Zielzustandes" gebildet, der hier angegeben werden kann (Beispiel siehe Kapitel 4.8.5 Seite184).

```
Transition[] getTransitions()
```

Aufgabe: Ermittelt die Menge aller Transitionen, die im aktuellen Zustand für den aktuellen Benutzer verfügbar sind (Beispiel siehe Kapitel 4.8.4 Seite 181).

```
Data getData();
```

Aufgabe: Einem Arbeitsablauf-Modell kann ein Formular zugeordnet werden. Dieses Formular wird dem Redakteur im Aktivitätsdialog angezeigt und er kann Daten eingeben oder ändern (siehe Kapitel 4.4 Seite 145). Über diese Methode hat das Skript Zugriff auf den Inhalt des Formulars und kann ggf. auch Veränderungen vornehmen (siehe Kapitel 4.4.1 Seite 146).

```
Map getSession()
```

Aufgabe: Jeder Instanz eines Arbeitsablaufs wird (neben dem Formular) eine spezielle Datenstruktur (Java-Map) zugeordnet, die es einem Skript erlaubt eine eigenen Instanzzustand zu speichern und ggf. zu verändern. Da dieser Zustand Teil der Arbeitsablauf-Instanz ist, steht er allen Skripten zur Verfügung, die während des Lebenszyklus der Instanz durchlaufenen werden. Somit ist es über diese Methode möglich (instanzbezogene Daten) zwischen Skripten auszutauschen (Beispiel siehe Kapitel 4.8.5 Seite184).

Beispiele:

Auflistung aller möglichen Übergänge mit Berechtigungen ausgehend von der aktuellen Aktion:

```
//!firstspirit.scripting.BeanshellWrapper

transitions = context.getTransitions();
```

```
print("Anzahl Transitionen:" + transitions.length);

for (i=0; i<transitions.length; i++) {
  print("Transition:" + transitions[i].getTarget());
  allowedUsers = transitions[i].getAllowedUsers();
  for (j=0; j<allowedUsers.size(); j++) {
    print("Allowed User:" + allowedUsers.get(j));
  }
}
```

Status-Verwaltung in Arbeitsablauf-Instanzen (Zähler):

```
//!firstspirit.scripting.BeanshellWrapper

state=context.getSession();
v=state.get("test");
if(v==null) v=0;
state.put("test",++v);
```

Erzeugen einer Instanz zu jedem vorhandenen Arbeitsablauf:

```
//!firstspirit.scripting.BeanshellWrapper

import firstspirit.access.store.templatestore.*;
u=context.getUserService();
ts=u.getTemplateStore();
wfs=ts.getWorkflows().getAllChilds(Workflow.class);

for (i=0; i<wfs.length; i++) {
    print("Workflow:" + wfs[i].getName());
    try {
       u.createTask(null, wfs[i], wfs[i].getName());
    } catch (Exception e) { print("Fehler!");}
}
```

Weitere Methoden können der FirstSpirit Access-API entnommen werden.

4.8.4 Beispiel: Ausgabe von Nachrichten in Arbeitsabläufen

Innerhalb eines Arbeitsablaufs können Nachrichten an den ausführenden Benutzer ausgegeben werden. Die Ausgabe von Nachrichten wird über Skripte innerhalb des Arbeitsablaufs realisiert. Der Bearbeiter, der die entsprechende Aktion innerhalb eines Arbeitsablaufs durchführt, bekommt über das Skript einen Dialog eingeblendet. Dort können bestimmte Informationen aus dem Kontext des Arbeitsablaufs angezeigt werden (siehe Kapitel 4.8.3 Seite 179).

Beispiel: Arbeitsablauf "Message":

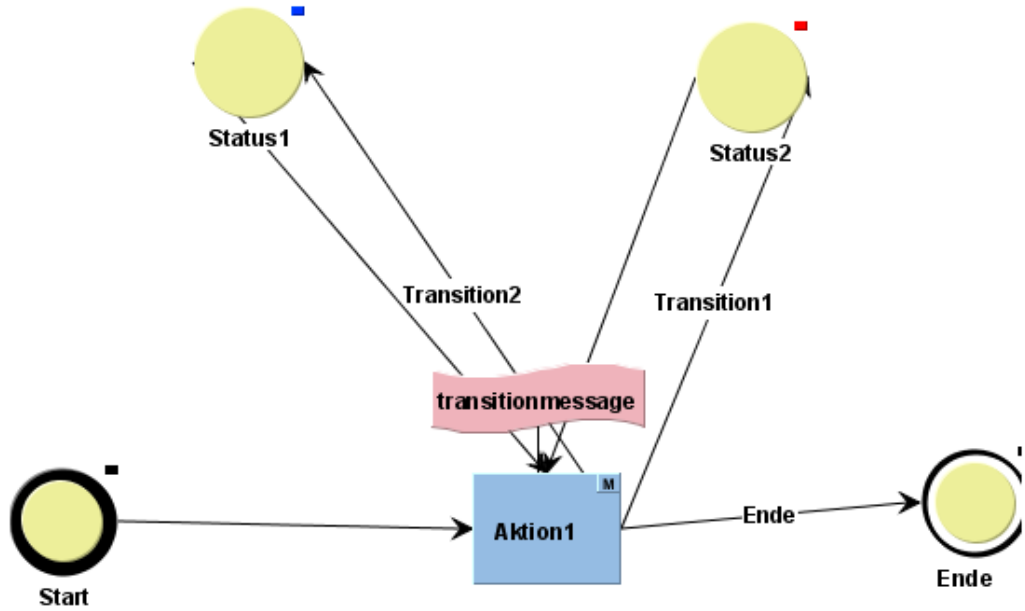


Abbildung 4-55: Beispiel-Arbeitsablauf "Message"

In diesem Beispiel-Arbeitsablauf wird vor und nach dem Schalten einer Transition ein Informationsdialog mit der Ausgabe "Hallo $USER" eingeblendet:

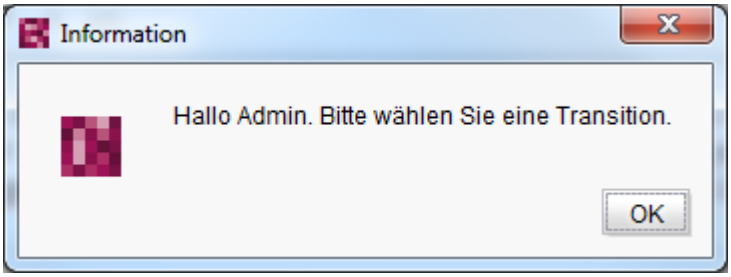
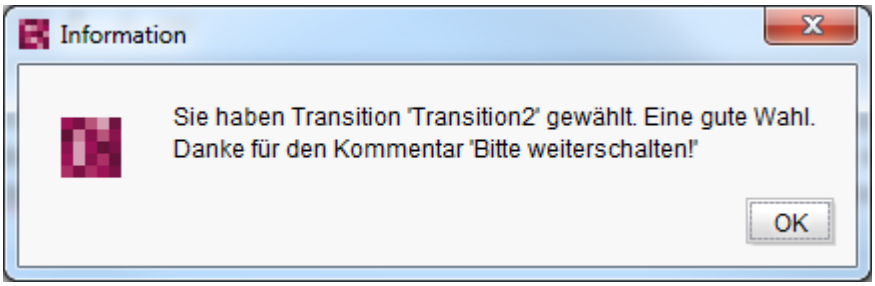


Abbildung 4-56: Erster Informationsdialog

Nach dem Schalten des Transitionsdialogs wird ein weiterer Informationsdialog mit der Ausgabe "Sie haben Transition $TRANSITION gewählt. Danke für den Kommentar $KOMMENTAR" angezeigt:



Skript "transitionMessage":

```
//!Beanshell

import de.espirit.firstspirit.common.gui.*;

import de.espirit.firstspirit.ui.operations.RequestOperation;

import de.espirit.firstspirit.agency.OperationAgent;


userName =
context.getGuiHost().getUserService().getUser().getLoginName();


text = "Hallo " + userName + ". Bitte wählen Sie eine Transition.";


requestOperation =
context.requireSpecialist(OperationAgent.TYPE).getOperation(RequestOperat
ion.TYPE);

requestOperation.setKind(RequestOperation.Kind.INFO);

requestOperation.addOk();

requestOperation.perform(text);


context.showActionDialog();

transition = context.getTransitionParameters();


if (transition.getTransition() != null) {


text="Sie haben Transition '" + transition.getTransition() + "' gewählt.
Eine gute Wahl.\nDanke für den Kommentar '" + transition.getComment() +
"'";

requestOperation =
context.requireSpecialist(OperationAgent.TYPE).getOperation(RequestOperat
ion.TYPE);

requestOperation.setKind(RequestOperation.Kind.INFO);

requestOperation.addOk();

requestOperation.perform(text);

context.doTransition(transition.getTransition());

} else {

requestOperation =
context.requireSpecialist(OperationAgent.TYPE).getOperation(RequestOperat
```

```
ion.TYPE);
requestOperation.setKind(RequestOperation.Kind.INFO);
requestOperation.perform("Sie haben keine Transition ausgewählt.");
}
```

Die Informationen, die dem Bearbeiter innerhalb der Dialoge angezeigt werden, werden im Skript über den Kontext des Arbeitsablaufs (`WorkflowScriptContext`) geholt, z. B. die Transitionsparameter (siehe Beispielskript):

```
context.getTransitionParameters();
```

4.8.5 Beispiel: Persistente Inhalte innerhalb von Arbeitsabläufen

Innerhalb von Arbeitsabläufen können Inhalte über die Session jetzt auch gespeichert und nach dem Schalten einer Transition erneut ausgelesen werden.

Beispiel: Arbeitsablauf "Counter":

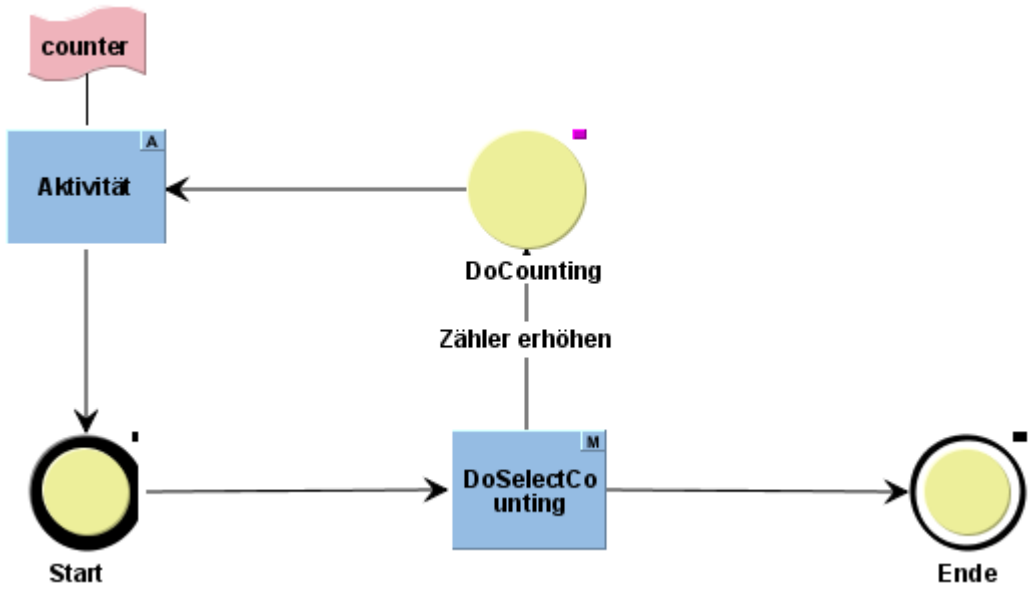


Abbildung 4-57: Beispiel-Arbeitsablauf "Counter"

Innerhalb der Aktivität "DoSelectCounting" kann ein Zähler bei jeder Ausführung des Arbeitsablaufs um den Wert 1 erhöht werden. Der Wert des Zählers wird gespeichert und beim erneuten Start des Arbeitsablaufs erneut um den Wert 1 erhöht. Der Wert wird dem Bearbeiter innerhalb eines Informationsdialogs angezeigt:

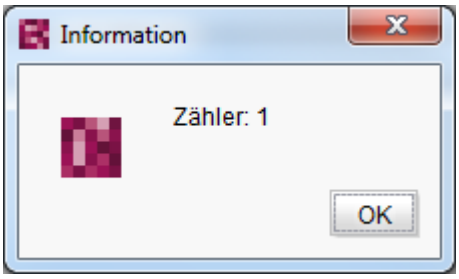


Abbildung 4-58: Wert des Zählers

Skript "counter":

```
//!Beanshell
import de.espirit.firstspirit.common.gui.*;
import de.espirit.firstspirit.ui.operations.RequestOperation;
import de.espirit.firstspirit.agency.OperationAgent;

session = context.getSession();
counter = session.get("counter");
if (counter == null) {
counter = new Integer(1);
}
text = "Zähler: " + counter;
requestOperation =
context.requireSpecialist(OperationAgent.TYPE).getOperation(Reques
tOperation.TYPE);
requestOperation.setKind(RequestOperation.Kind.INFO);
requestOperation.addOk();
requestOperation.perform(text);

session.put("counter", new Integer(counter + 1));
context.doTransition("->Start");
```

4.9 Löschen über einen Arbeitsablauf

Für das Löschen von Elementen im FirstSpirit JavaClient und im FirstSpirit WebClient kann ein projektspezifischer Arbeitsablauf erstellt und direkt an die bisherigen Bedienelemente zum Löschen (Buttons der Menüleiste, Kontextmenüeintrag) von Elementen gebunden werden. Statt dem einfachen Löschen eines Objekts, beispielsweise einer Seite, kann über den Arbeitsablauf eine

komplexere Löschfunktionalität bereitgestellt werden, beispielsweise das zusätzliche Löschen abhängiger Objekte einer Seite (Demo-Arbeitsablauf siehe Kapitel 4.9.2).

Das Löschen über einen Arbeitsablauf steht nur zur Verfügung, wenn das Projekt vom Projektadministrator entsprechend konfiguriert wurde.

Innerhalb der Clients wird dann über die bekannten Bedienelemente der neue Arbeitsablauf gestartet. Die einzelnen Aufgaben des Arbeitsablaufs erscheinen, wie üblich, in der Aufgabenliste (siehe Kapitel 4.9.1 Seite 186).

Wird innerhalb eines Projekts das Löschen über einen Arbeitsablauf konfiguriert, muss die Rechtekonfiguration für den Arbeitsablauf angepasst werden. Die herkömmlichen Redaktionsrechte zum Löschen, die für einen Benutzer oder eine Gruppe definiert werden, greifen nur dann, wenn im Arbeitsablauf die Rechtekonfiguration entsprechend angepasst wird (siehe Kapitel 4.9.3 Seite 189).

4.9.1 Löschen über einen Arbeitsablauf im JavaClient

Wurde das Löschen von Elementen im Projekt an einen Arbeitsablauf gebunden, kann der Arbeitsablauf im JavaClient durch die herkömmlichen Bedienelement zum Löschen gestartet bzw. weitergeschaltet werden. Dazu stehen die folgenden Bedienelemente zur Verfügung:

- Element markieren und Taste <Entf> klicken.
- Element markieren und den Kontextmenüeintrag "Löschen" ausführen

- Element markieren und das Icon  in der Iconleiste klicken

Analog zur Mehrfachselektion von Arbeitsabläufen kann auch das Löschen über einen Arbeitsablauf parallel auf einer Menge von Objekten ausgeführt werden (siehe Abbildung 4-59 und Kapitel 4.11 Seite 206).

Der Arbeitsablauf kann nur gestartet werden, wenn bisher keine Arbeitsabläufe auf einem der markierten Objekte gestartet wurden und der Benutzer die entsprechenden Rechte zum Ausführen des Arbeitablaufs besitzt. Andernfalls sind die entsprechenden Bedienelemente deaktiviert.

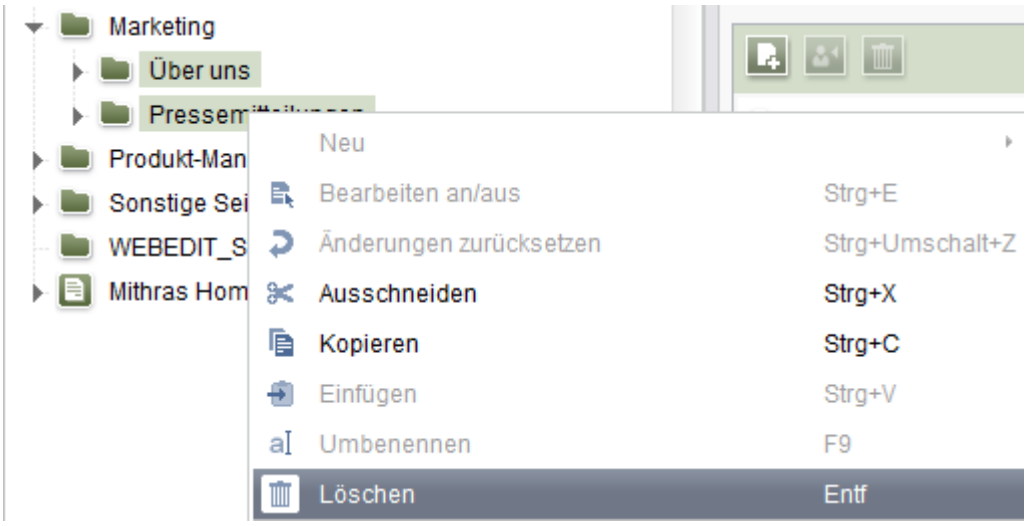


Abbildung 4-59: Mehrfachselektion beim Löschen über einen Arbeitsablauf

Das Recht "Löschen" wird auch ausgewertet, wenn Elemente über einen Arbeitsablauf gelöscht werden. Besitzt ein Benutzer das Recht zum Schalten des Arbeitsablaufs aber NICHT das Recht zum Löschen von Elementen, kann der Arbeitsablauf zwar gestartet werden (Kontextmenü-Eintrag "Löschen" ist aktiviert), das Löschen des Elements ist aber nicht möglich. Die Transition, die das Element löscht, wird diesen Benutzern nicht angezeigt.

4.9.2 Löschen über einen Arbeitsablauf im WebClient

Wurde das Löschen von Elementen im Projekt an einen Arbeitsablauf gebunden, kann der Arbeitsablauf im WebClient über das Menü „Inhalte/Löschen“ oder über das Statusmenü gestartet werden. Ebenfalls im Statusmenü lässt sich ein gestarteter Arbeitsablauf zum Löschen eines Elements weiter schalten.

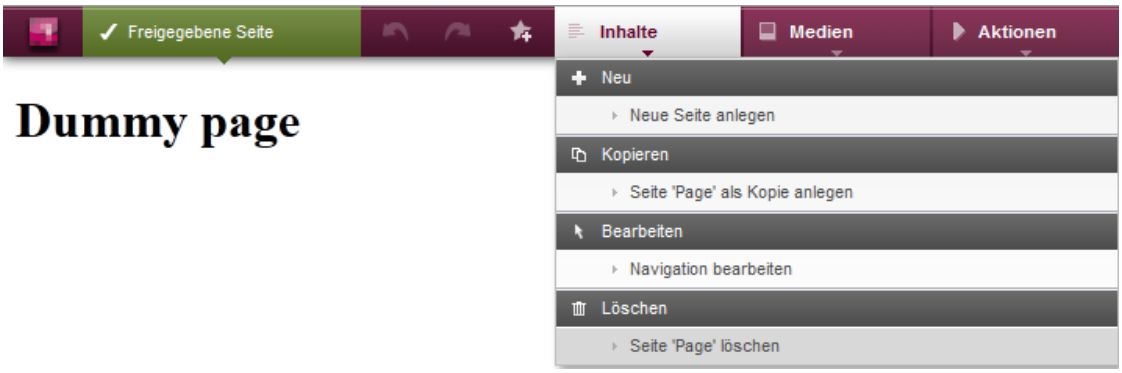


Abbildung 4-60: Arbeitsablauf zum Löschen einer Seite im WebClient – Inhalte-Menü

Abbildung 4-61: Arbeitsablauf zum Löschen einer Seite im WebClient – Status-Menü

Abbildung 4-62: Weiterschalten eines Arbeitsablaufes zum Löschen eines Elements

*Es ist zu beachten, dass ein Workflow im WebClient **immer** auf einer Seitenreferenz ausgeführt wird und diese somit den Kontext für den Arbeitsablauf darstellt. Soll auch die zugehörige Seite gelöscht werden, muss dies über das vom Arbeitsablauf verwendete Skript gesteuert werden. Hierbei ist die Reihenfolge der zu löschenden Elemente zu beachten (siehe auch Kapitel 2.2.8.1 Seite 29).*

Der Arbeitsablauf kann nur gestartet werden, wenn bisher keine Arbeitsabläufe auf einem der markierten Objekte gestartet wurden und der Benutzer die entsprechenden Rechte zum Ausführen des Arbeitablaufs besitzt.

4.9.3 Rechtekonfiguration

Die Rechtevergabe erfolgt im FirstSpirit JavaClient. Hier können alle Bereiche des Projekts mit Rechten für bestimmte Gruppen oder Benutzer versehen werden (siehe Kapitel 4.6 Seite 163).

Die Rechte zum Löschen von Elementen (ohne Arbeitsablauf) werden normalerweise über die sogenannten Redaktionsrechte definiert. Redaktionsrechte werden für einen Benutzer bzw. eine Gruppe auf dem jeweiligen Elements definiert. Für alle redaktionellen Arbeiten können so bestimmte Rechte vergeben werden. Dazu zählt neben "Sichtbar" oder "Ändern" beispielsweise auch das Recht "Objekt löschen" und das Recht "Ordner löschen".

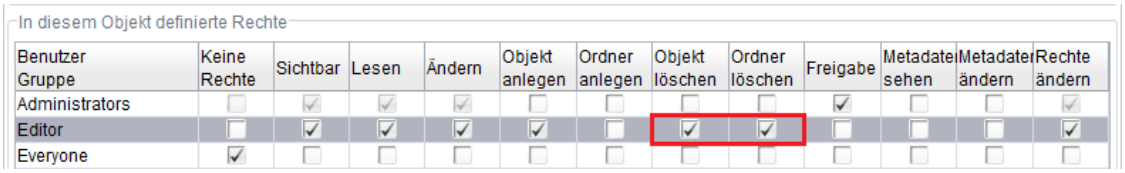


Abbildung 4-63: Redaktionsrechte "Objekt löschen" und "Ordner löschen"

Weitere Informationen zu Redaktionsrechten siehe FirstSpirit Handbuch für Redakteure, Kapitel 13.1.

Diese Redaktionsrechte greifen nicht automatisch, wenn das Löschen an einen Arbeitsablauf gebunden wird. Sollen diese Rechte ausgewertet werden, muss zunächst die Rechtekonfiguration im Arbeitsablauf angepasst werden (siehe Abbildung 4-65).

Wird das Löschen in einem Projekt über einen Arbeitsablauf behandelt, muss die Rechtekonfiguration auf den Arbeitsablauf verlagert werden. Die Rechte zur Ausführung von Arbeitsabläufen werden parallel zu den Redaktionsrechten für Gruppen und Benutzer im Dialog "Rechtevergabe" innerhalb der Verwaltungsbereiche im FirstSpirit JavaClient vergeben:

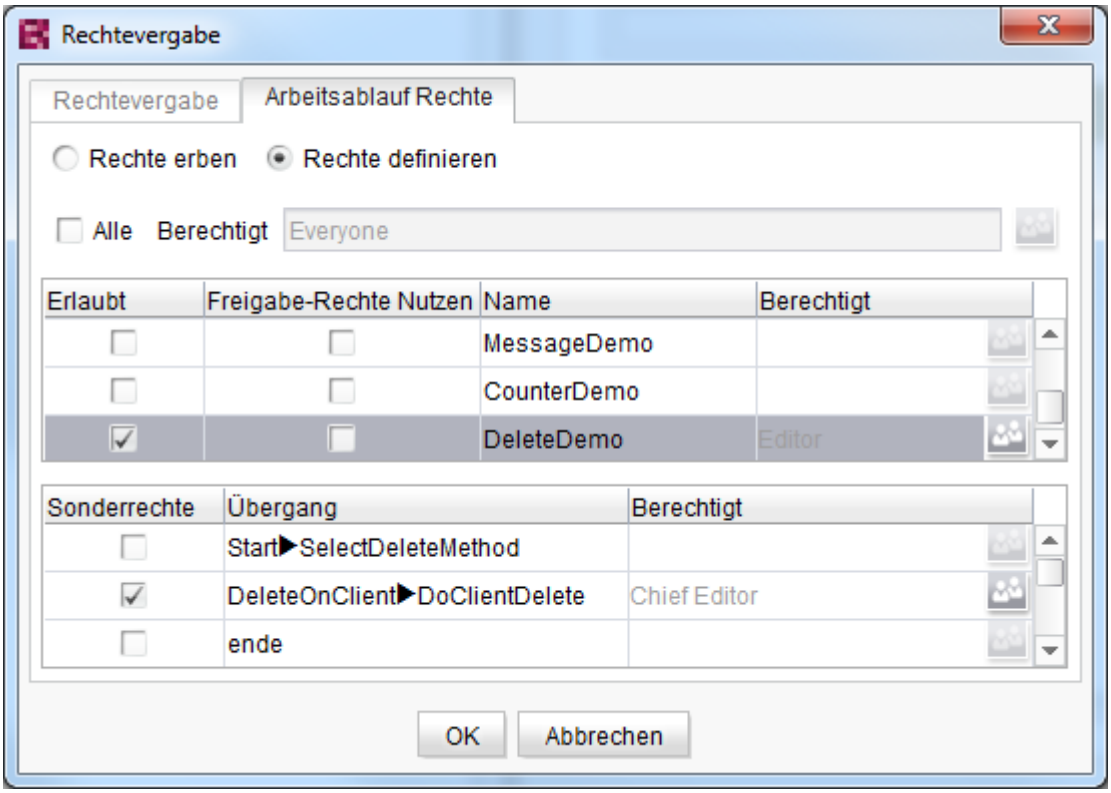


Abbildung 4-64: Rechte zur Ausführung von Arbeitsabläufen

Die Rechte die im oberen Dialogbereich für die Ausführung von Arbeitsabläufen definiert werden ("Erlaubt"), beziehen sich ausschließlich auf das Starten des jeweiligen Arbeitsablaufs (siehe Kapitel 4.6.3 Seite 167).

Die Rechte für die Ausführung eines Übergangs (von einem Schritt des Arbeitsablaufs zum nächsten Schritt) werden entweder:

- über den Vorlagen-Entwickler im Arbeitsablauf festgelegt (siehe Kapitel 4.6.1 Seite 163)
- über die Vergabe von "Sonderrechten" für die einzelnen Schritte eines Arbeitsablaufs (siehe Abbildung 4-64) (siehe Kapitel 4.6.4 Seite 171)

Weitere Informationen zu Rechten zur Ausführung von Arbeitsabläufen siehe FirstSpirit Handbuch für Redakteure, Kapitel 13.2

Sollen die herkömmlichen Redaktionsrechte ("Ordner löschen", "Objekt löschen") und die Rechte zum Ausführen des Arbeitsablaufs geeignet miteinander verknüpft werden, sollte die Rechtekonfiguration innerhalb des Arbeitsablaufs angepasst werden. Innerhalb eines Arbeitsablaufes erfolgt die Rechtevergabe an den einzelnen Übergängen (siehe Kapitel 4.5.5.2 Seite 160). Dadurch wird gewährleistet, dass jede einzelne Aktivität nur von berechtigten Benutzern durchgeführt werden kann. Der

Rechtedialog öffnet sich bei einem Doppelklick auf die Transition im Modell des Arbeitsablaufs. Im Register "Rechte" können die Rechte für das Schalten der Transition vergeben werden:

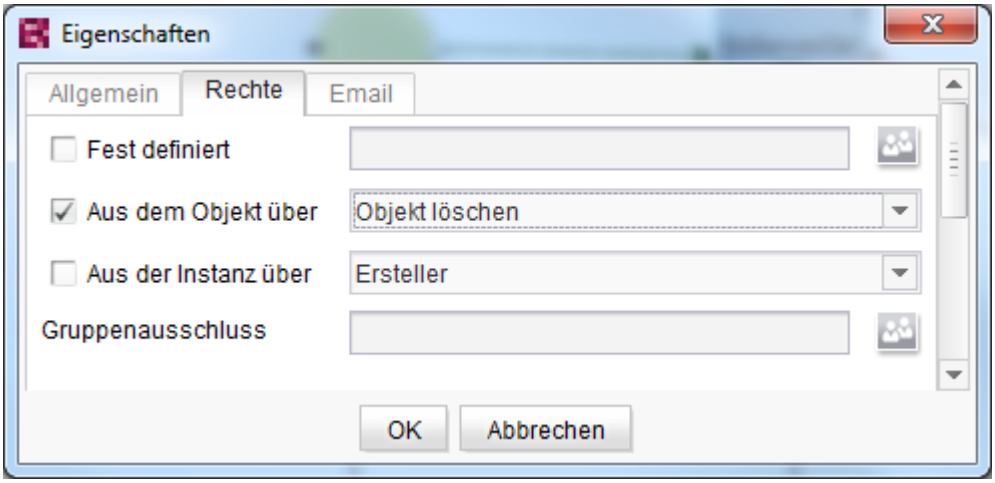


Abbildung 4-65: Redaktionsrechte und Transitionsrechte verknüpfen

Wird die Option "aus dem Objekt über" aktiviert, dann ergeben sich die berechtigten Benutzer aus den Redaktionsrechten, die in der Baumstruktur des JavaClients definiert wurden. In dem Feld kann ausgewählt werden, über welches Recht der Benutzer auf dem betrachteten Objekt verfügen muss, um diesen Übergang durchführen zu dürfen. Wird hier das Recht "Objekt löschen" bzw. "Ordner löschen" ausgewählt, wird beim Starten des Arbeitsablaufs untersucht, ob der Benutzer auf dem Element das Recht zum "Löschen" besitzt. Die Auswertung der Rechte erfolgt dann analog zum herkömmlichen Löschen ohne einen Arbeitsablauf.

Sonderfall "Löschen von Objekten": Beim Löschen eines Objekts über einen Arbeitsablauf in Kombination mit der Rechtekonfiguration über die Option "aus dem Objekt über" greift ein Sonderfall. Wird das Element gelöscht, können die Rechte nicht mehr aus dem Objekt ermittelt werden. Im Beispiel aus 4.9.4 steht das Objekt und damit auch die auf dem Objekt definierten Rechte auf der letzten Transition "ende" nicht mehr zur Verfügung. In diesem Fall gilt: auf einem gelöschten Objekt ist immer "alles erlaubt". Ist das nicht erwünscht, muss die Rechtekonfiguration für die entsprechenden Transitionen geändert werden (z. B. auf "fest definierte" Gruppen oder Benutzer).

Werden "Sonderrechte" für die Weiterschaltung eines Arbeitsablaufs auf einem Element definiert (siehe Abbildung 4-64), werden dadurch die Rechte, die vom Vorlagen-Entwickler für diesen Arbeitsablauf definiert wurden, überschrieben.

4.9.4 Beispiel: Arbeitsablauf "Delete"

Der Arbeitsablauf zum Löschen von Elementen besteht aus dem Arbeitsablauf und den zugehörigen Skripten "clientdelete" (zum Löschen einzelner Objekte) und "serverdelete" (zum Löschen von Teilbäumen).

Wird die Aktion "clientdelete" ausgeführt, wird das Element über das entsprechende Skript gesperrt und anschließend gelöscht. Nach dem Löschen wird der Arbeitsablauf in den nachfolgenden Status "Ende" weitergeschaltet.

Wird die Aktion "serverdelete" ausgeführt, wird das Element über das entsprechende Skript rekursiv gesperrt. Über die Aktion "serverdelete" können damit nicht nur einzelne Element, sondern auch Teilbäume gelöscht werden. Das Löschen erfolgt hier über ein ServerHandle, welches einen Ergebnis-Report zurückliefert und im Fehlerfall eine Exception wirft.

Nach erfolgreichem Löschen wird der Arbeitsablauf in den nachfolgenden Status "Ende" weitergeschaltet. Für beide Aktionen gilt: im Fehlerfall wird der Arbeitsablauf nicht in den Endstatus, sondern in einen Fehlerstatus geschaltet, der im Arbeitsablauf modelliert wurde.

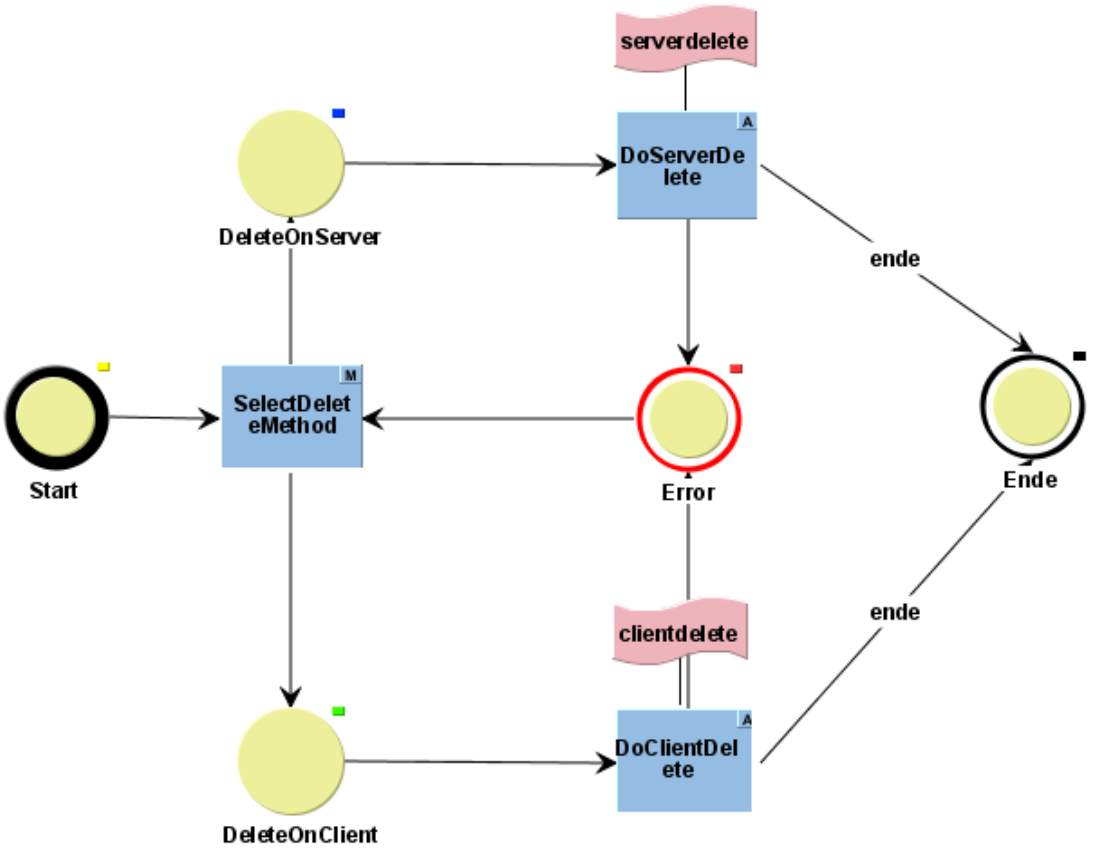


Abbildung 4-66: Beispiel-Arbeitsablauf "Delete"

Skript "clientdelete":

```
//!Beanshell
import de.espirit.firstspirit.common.gui.*;
import de.espirit.firstspirit.ui.operations.RequestOperation;
import de.espirit.firstspirit.agency.OperationAgent;

se = context.getStoreElement();
try {
se.setLock(true);
se.delete();
context.doTransition("->Ende");
} catch (Exception ex) {
text = "Fehler beim Löschen: " + ex;
requestOperation =
context.requireSpecialist(OperationAgent.TYPE).getOperation(Reques
tOperation.TYPE);
requestOperation.setKind(RequestOperation.Kind.INFO);
```

```
requestOperation.addOk();
requestOperation.perform(text);
```

```
context.getSession().put("error", ex.toString());
context.doTransition("->Error");
}
```

Skript "serverdelete":

```
//!Beanshell
import de.espirit.firstspirit.common.gui.*;
import de.espirit.firstspirit.ui.operations.RequestOperation;
import de.espirit.firstspirit.agency.OperationAgent;

se = context.getStoreElement();
parent = se.getParent();
try {
se.setLock(false, false);
handle = de.espirit.firstspirit.access.AccessUtil.delete(se,
true);
handle.getResult();
handle.checkAndThrow();

Set notDeleted = new HashSet();
progress = handle.getProgress(true);
notDeleted.addAll(progress.getDeleteFailedElements());
notDeleted.addAll(progress.getMissingPermissionElements());
notDeleted.addAll(progress.getLockFailedElements());
notDeleted.addAll(progress.getReferencedElements());
if (!notDeleted.isEmpty()) {
          text = "Folgende Elemente konnten nicht gelöscht werden:
" + notDeleted;
          requestOperation =
context.requireSpecialist(OperationAgent.TYPE).getOperation(Reques
tOperation.TYPE);
          requestOperation.setKind(RequestOperation.Kind.INFO);
          requestOperation.addOk();
          requestOperation.perform(text);
}
```

```
if (parent != null) {
          parent.refresh();
          context.getGuiHost().gotoTreeNode(parent);
}

if (!se.isDeleted()) {
          se.setLock(true, false);
}
context.doTransition("->Ende");

} catch (Exception ex) {
text = "Fehler beim Löschen: " + ex;
requestOperation =
context.requireSpecialist(OperationAgent.TYPE).getOperation(Reques
tOperation.TYPE);
requestOperation.setKind(RequestOperation.Kind.INFO);
requestOperation.addOk();
requestOperation.perform(text);

context.getSession().put("error", ex.toString());
context.doTransition("->Error");
}
```

4.9.5 Beispiel: Arbeitsablauf "ContentDeleteDemo"

Neben dem Löschen von einzelnen Elementen und Teilbäumen (siehe Kapitel 4.9.4 Seite 192), ist auch das Löschen von strukturierten Daten über einen Arbeitsablauf möglich.

Dazu muss innerhalb des Arbeitsablaufs (im Skript) zwischen normalen Elementen ("StoreElementen", z. B. Seiten, Medien, Seitenreferenzen) und "Entitäten" unterschieden werden.

Im Skript wird diese Information über den Kontext des Arbeitsablaufs (`WorkflowScriptContext`) geholt (siehe Kapitel 4.8.3 Seite 179):

```
workflowable = context.getWorkflowable()
```

Die Methode getWorkflowable() liefert zurück, ob es sich beim Element auf dem der Arbeitsablauf gestartet wurde, um ein StoreElement, beispielsweise ein Medium, oder um eine Entität, in Form eines Datensatzes handelt (siehe Beispielskript).

Dementsprechend kann beispielsweise die Ausgabe des Skripts angepasst werden:

```
if (workflowable instanceof ContentWorkflowable) {
...
} else {
...
}
```

Im Beispiel wird die Ausgabe, abhängig vom Kontext gesteuert auf dem der Arbeitsablauf gestartet wurde. Wird die Lösch-Funktionalität auf einem Datensatz gestartet, liefert das Skript die Ausgabe:

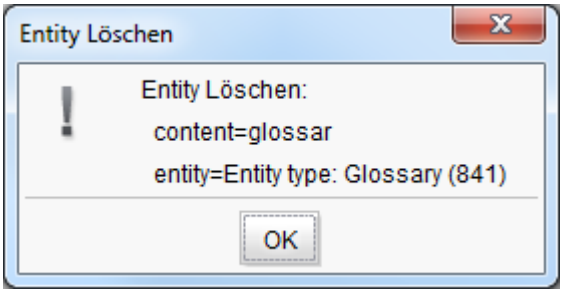


Abbildung 4-67: Entity löschen

Auf einem "StoreElement" die Ausgabe:

Abbildung 4-68: StoreElement löschen

Das Löschen erfolgt in diesem Beispiel ebenfalls direkt über den WorkflowScriptContext (siehe Kapitel 4.8.3 Seite 179):

```
workflowable.delete();
```

Die Methode delete wird hier auf dem Objekt `Workflowable` und nicht, wie im Beispiel aus Kapitel 4.9.4 auf dem `StoreElement` aufgerufen. Über diese Delete-Methode kann sowohl ein `StoreElement` als auch ein Datensatz (`Entity`) gelöscht werden.

Der Arbeitsablauf zum Löschen von Entitäten besteht aus dem Arbeitsablauf und dem zugehörigen Skript "deletecontentdemo" (zum Löschen einzelner Entitäten)

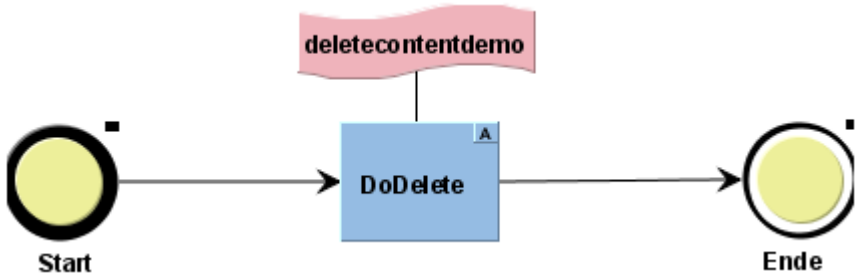


Abbildung 4-69: Beispiel-Arbeitsablauf "DeleteContentDemo"

Nach erfolgreichem Löschen wird der Arbeitsablauf automatisch in den nachfolgenden Status "Ende" weitergeschaltet.

Skript ("deletecontentdemo"):

```
//!Beanshell
import de.espirit.firstspirit.access.*;
import de.espirit.firstspirit.access.store.contentstore.*;
import de.espirit.firstspirit.ui.operations.RequestOperation;
import de.espirit.firstspirit.agency.OperationAgent;


workflowable = context.getWorkflowable();
if (workflowable instanceof ContentWorkflowable) {
message = "Entity Löschen:\n  content=" +
workflowable.getContent().getName() + "\n  entity=" +
workflowable.getEntity().getKeyValue();
requestOperation =
context.requireSpecialist(OperationAgent.TYPE).getOperation(Reques
tOperation.TYPE);
requestOperation.setKind(RequestOperation.Kind.INFO);
requestOperation.addOk();
requestOperation.perform(message);}
else {
message = "StoreElement Löschen:\n  store=" +
workflowable.getStore().getName() + "\n  id=" +
workflowable.getId();
requestOperation =
context.requireSpecialist(OperationAgent.TYPE).getOperation(Reques
tOperation.TYPE);
requestOperation.setKind(RequestOperation.Kind.INFO);
```

```
requestOperation.addOk();
requestOperation.perform(message);
}
workflowable.delete();
context.doTransition("->Ende");
```

4.10 Arbeitsabläufe mit komplexer Funktion

Über die Modellierung von Arbeitsabläufen und die Verwendung von BeanShell-Skripten innerhalb dieser Modelle können sehr komplexe Funktionalitäten realisiert werden. Ein gutes Beispiel ist der zuvor beschriebene Arbeitsablauf zum Löschen von Elementen (siehe Kapitel 4.9.4 Seite 192).

Innerhalb der Skripte werden Aktionen auf bestimmten Projektinhalten über die Access-API von FirstSpirit ausgeführt. Im Unterschied zu den Standard-Funktionen (z. B. beim Standard-Kontextmenüeintrag "Löschen"), muss der Entwickler des Arbeitsablaufs (bzw. der zugehörigen Skripte) hier dafür sorgen, dass alle erforderlichen Randbedingungen, beispielsweise zum Löschen eines Elements, ebenfalls durch das Skript abgedeckt werden. Für die meisten Aktionen ist dazu mindestens ein Schreibschutz ("Lock") auf dem betreffenden Element notwendig. Dabei ist jedoch entscheidend, welche Art von Aktion auf welchen Elementen ausgeführt werden soll. Beim einfachen Löschen eines Elements, z. B. eines Mediums, muss beispielsweise kein Schreibschutz gesetzt werden, beim rekursiven Löschen, z. B. einer Seite mit Absätzen, jedoch schon (vgl. Kapitel 4.9.4 Seite 192).

Die folgenden Kapitel behandeln den Schreibschutz innerhalb von Arbeitsabläufen und erläutern Beispiele zu Arbeitsabläufen mit komplexer Funktionalität.

4.10.1 Beispiel: Arbeitsablauf "RecursiveLock"

Dieser Arbeitsablauf zur rekursiven Sperrung von Teilbäumen besteht aus dem Arbeitsablauf und dem zugehörigen Skript "lockrecursive".

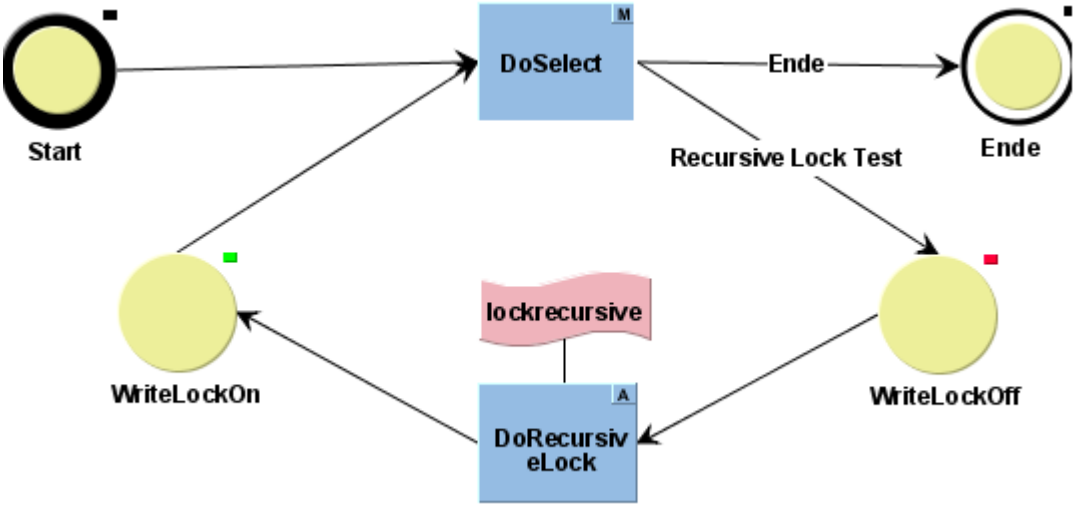


Abbildung 4-70: Beispiel-Arbeitsablauf "RecursiveLock"

Innerhalb des Skripts wird ein rekursiver Schreibschutz auf dem Element ausgeführt, auf dem der Arbeitsablauf gestartet wurde. Damit dies gelingt, muss zuvor der automatisch gesetzte Schreibschutz des Arbeitsablaufs aufgehoben werden. Dazu wird im Status "WriteLockOff" zunächst der Schreibschutz entfernt:

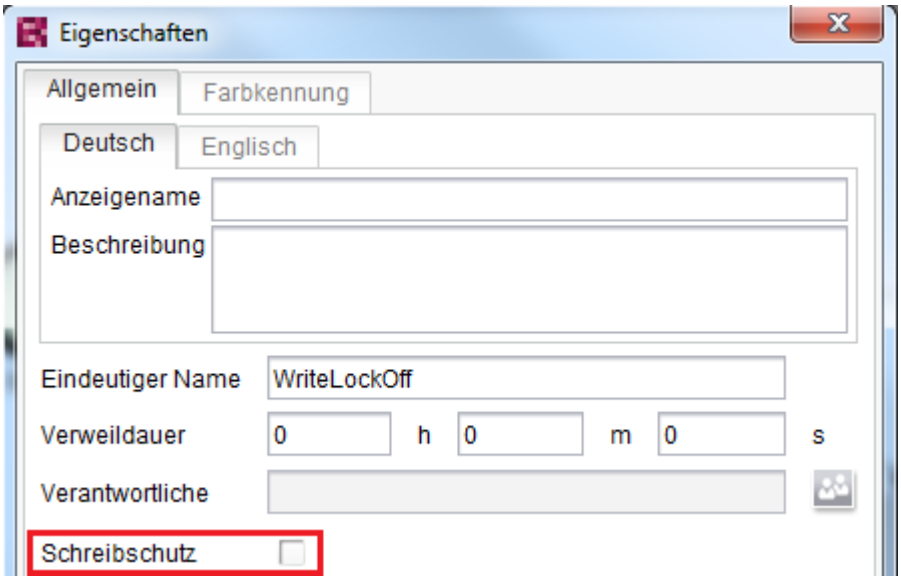


Abbildung 4-71: Schreibschutz über den Status des Arbeitsablaufs entfernen

Die Checkbox "Schreibschutz" ist deaktiviert, der Schreibschutz des Arbeitsablaufs wird beim Schalten der Transition "Recursive Lock Test" nun aufgehoben. Die nachfolgende Aktion "DoRecursiveLock" wird automatisch ausgeführt und ist mit dem Skript "lockrecursive" verknüpft.

Über das Skript kann nun ein rekursiver Schreibschutz auf dem Element gesetzt werden:

```
// set recursive lock
se = context.getStoreElement();
se.setLock(true);
text = "Teilbaum gesperrt!";
    requestOperation =
context.requireSpecialist(OperationAgent.TYPE).getOperation(Reques
tOperation.TYPE);
    requestOperation.setKind(RequestOperation.Kind.INFO);
    requestOperation.addOk();
    requestOperation.perform(text);
```

Die Elemente werden rekursiv gesperrt, dem Bearbeiter wird ein Dialog mit der Meldung "Teilbaum gesperrt" angezeigt:

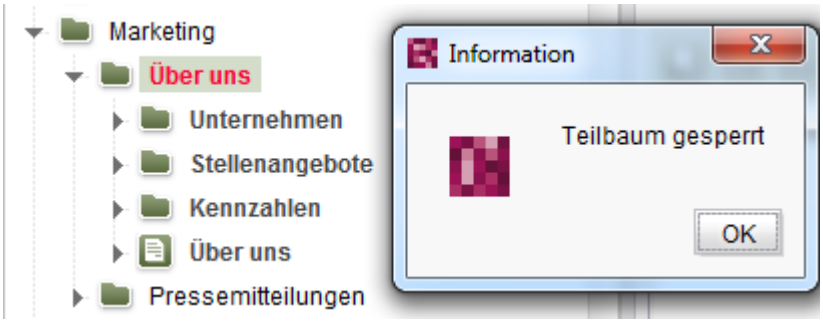


Abbildung 4-72: Schreibschutz auf dem Teilbaum gesetzt

Beim Bestätigen der Meldung wird das Skript weiter ausgeführt, der rekursive Schreibschutz auf dem Teilbaum wird wieder aufgehoben:

```
// reset recursive lock
se.setLock(false);
```

Im nächsten Schritt muss der einfache Schreibschutz auf dem Element wiederhergestellt werden:

```
// non recursive lock, normal state during workflow
se.setLock(true, false);
context.doTransition("->WriteLockOn");
```

Das ist notwendig, um die nachfolgende Transition innerhalb des Arbeitsablaufs schalten zu können.

Abschließend muss der Standard-Schreibschutz des Arbeitsablaufs wiederhergestellt werden. Dazu wird im Status "WriteLockOn" die Checkbox "Schreibschutz" wieder aktiviert:

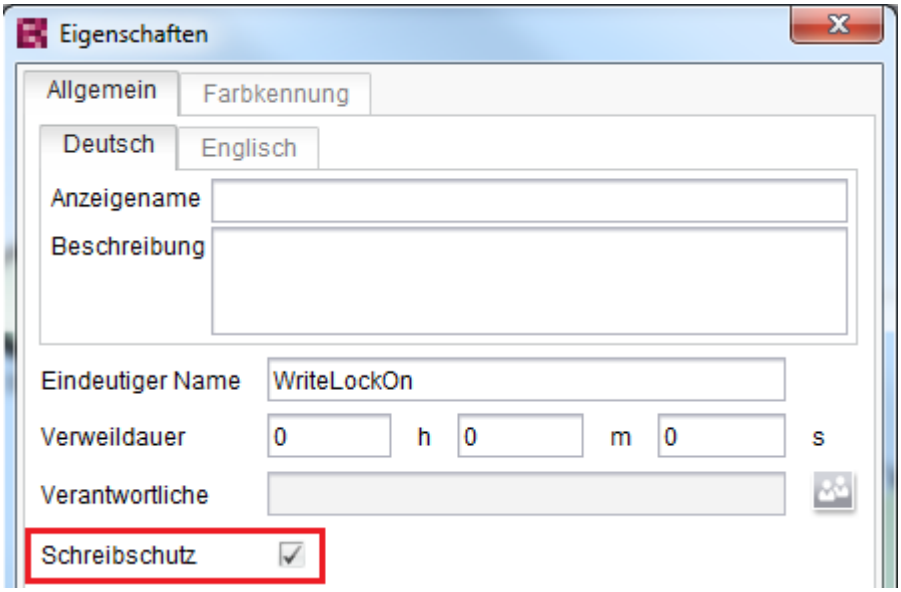


Abbildung 4-73: Schreibschutz über den Status des Arbeitsablaufs setzen

Skript "lockrecursive":

```
//!Beanshell
import de.espirit.firstspirit.common.gui.*;
import de.espirit.firstspirit.ui.operations.RequestOperation;
import de.espirit.firstspirit.agency.OperationAgent;

// set recursive lock
se = context.getStoreElement();
se.setLock(true);
text = "Teilbaum gesperrt!";
requestOperation =
context.requireSpecialist(OperationAgent.TYPE).getOperation(Reques
tOperation.TYPE);
requestOperation.setKind(RequestOperation.Kind.INFO);
requestOperation.addOk();
requestOperation.perform(text);
```

```
// reset recursive lock
se.setLock(false);

// non recursive lock, normal state during workflow
se.setLock(true, false);
context.doTransition("->WriteLockOn");
```

4.10.2 Beispiel: Arbeitsablauf "RecursiveRelease"

Dieser Arbeitsablauf zur rekursiven Freigabe besteht aus dem Arbeitsablauf und dem zugehörigen Skript "serverrelease".

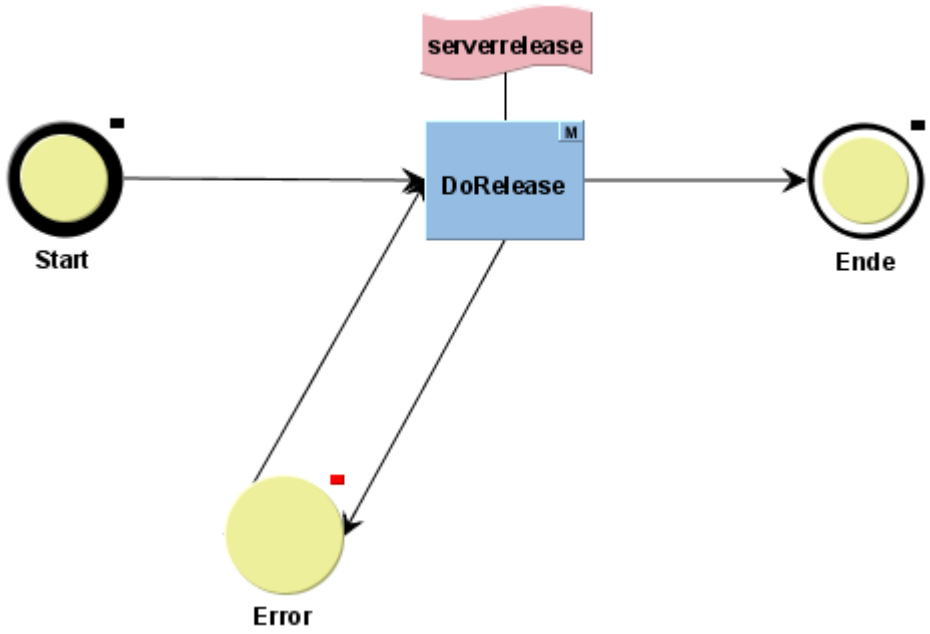


Abbildung 4-74: Beispiel-Arbeitsablauf "RecursiveRelease"

Innerhalb des Arbeitsablaufs soll das Element, auf dem der Arbeitsablauf gestartet wurde, sowie alle abhängigen Elemente, rekursiv freigegeben werden.

Die Serverseitige Freigabe, die innerhalb des Skripts "serverrelease" verwendet wird, regelt sowohl die Freigabe als auch das interne Setzen des Schreibschutzes für die betroffenen Element. Werden dabei Elemente gefunden, die bereits mit einem Schreibschutz versehen sind, kann die Serverseitigen Freigabe nicht ausgeführt werden. Diese betroffenen Elemente können über den Rückgabewert der Serverseitigen Freigabe abgerufen werden (nur im Testmodus):

```
handle.getProgress(true).getLockFailedElements()
```

Für die serverseitige Freigabe muss demnach kein rekursiver Schreibschutz über

das Skript gesetzt werden. Damit die Freigabe erfolgen kann, darf aber kein Schreibschutz durch den Arbeitsablauf gesetzt sein. Der Schreibschutz auf dem Element wird über das Skript daher zunächst aufgehoben:

```
    se.setLock(false, false);
```

Anschließend wird die serverseitige Freigabe über den Aufruf der Methode:

```
AccessUtil.release(IDProvider releaseStartNode, boolean checkOnly,
boolean releaseParentPath, boolean recursive,
IDProvider.DependentReleaseType dependentType)
```

ausgeführt.

Im Beispiel werden die folgenden Übergabeparameter für die Serverseitige Freigabe eingestellt:

```
handle = AccessUtil.release(se, false, false, true,
de.espirit.firstspirit.access.store.IDProvider.DependentReleaseTyp
e.DEPENDENT_RELEASE_NEW_AND_CHANGED);
```

Eine Erläuterung der verwendeten Übergabe-Parameter finden Sie im Kapitel 6 Seite 222.

Rückgabe-Parameter:

```
ServerActionhandle<? extends ReleaseProgress,Boolean >
```

Die serverseitige Freigabe liefert ein `ServerActionHandle` zurück, das alle Informationen über den Freigabeprozess beinhaltet.

Innerhalb des Beispielskripts wird zunächst das Ergebnis des Freigabeprozesses abgefragt:

```
handle.getResult();
handle.checkAndThrow();
```

Anschließend werden die Fehler bei der Freigabe untersucht. Können Elemente nicht freigegeben werden, beispielsweise weil ein Schreibschutz auf dem Element bestand oder der Bearbeiter nicht die entsprechenden Rechte zur Freigabe eines Elements besaß, können diese über die Methoden `progress.getMissingPermissionElements()` bzw. `progress.getLockFailedElements()` abgerufen werden:

```
    progress = handle.getProgress(true);

    notReleased.addAll(progress.getMissingPermissionElements());
    notReleased.addAll(progress.getLockFailedElements());
```

Die Fehlerbehandlung des Skripts zeigt dem Bearbeiter die Elemente an, die nicht freigegeben werden konnten:

```
if (!notReleased.isEmpty()) {
text = "Folgende Elemente konnten nicht freigegeben werden: " +
notReleased;
          requestOperation =
context.requireSpecialist(OperationAgent.TYPE).getOperation(Reques
tOperation.TYPE);
          requestOperation.setKind(RequestOperation.Kind.ERROR);
          requestOperation.addOk();
          requestOperation.perform(text);
}
```

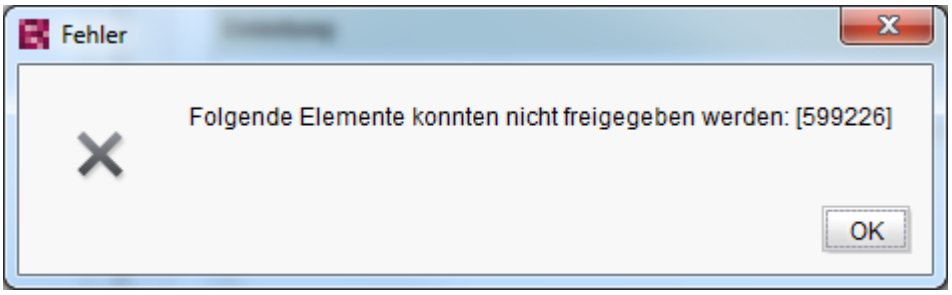


Abbildung 4-75: Fehlermeldung – Nicht freigegebene Elemente

Die Fehlermeldung wird nur im Test-Modus ("`checkOnly`") angezeigt.

Skript: "serverrelease":

```
//!Beanshell
import de.espirit.firstspirit.common.gui.*;
import de.espirit.firstspirit.access.*;
import de.espirit.firstspirit.access.store.*;
import de.espirit.firstspirit.ui.operations.RequestOperation;
import de.espirit.firstspirit.agency.OperationAgent;

se = context.getStoreElement();
try {
se.setLock(false, false);
handle = AccessUtil.release(se, false, false, true,
de.espirit.firstspirit.access.store.IDProvider.DependentReleaseTyp
e.DEPENDENT_RELEASE_NEW_AND_CHANGED);
```

```
handle.getResult();
handle.checkAndThrow();
Set notReleased = new HashSet();
progress = handle.getProgress(true);


notReleased.addAll(progress.getMissingPermissionElements());
notReleased.addAll(progress.getLockFailedElements());
if (!notReleased.isEmpty()) {
          text = "Folgende Elemente konnten nicht freigegeben
werden: " + notReleased;
          requestOperation =
context.requireSpecialist(OperationAgent.TYPE).getOperation(Reques
tOperation.TYPE);
          requestOperation.setKind(RequestOperation.Kind.ERROR);
          requestOperation.addOk();
          requestOperation.perform(text);
}
se.refresh();
context.getGuiHost().gotoTreeNode(se);
se.setLock(true, false);
context.doTransition("->Ende");
} catch (Exception ex) {
text = "Fehler bei der Freigabe: " + ex;
requestOperation =
context.requireSpecialist(OperationAgent.TYPE).getOperation(Reques
tOperation.TYPE);
requestOperation.setKind(RequestOperation.Kind.ERROR);
requestOperation.addOk();
requestOperation.perform(text);


context.getSession().put("error", ex.toString());
context.doTransition("->Error");
}
```

4.11 Mehrfachselektion von Arbeitsabläufen

In FirstSpirit steht innerhalb vieler Dialoge die Möglichkeit zur Mehrfachauswahl von Elementen ausgewählt und bearbeitet werden. Eine Mehrfachselektion ist beispielsweise innerhalb der Baumansicht im FirstSpirit JavaClient möglich (siehe Abbildung 4-76). Damit können mehrere Elemente markiert werden, auf denen anschließend parallel eine bestimmte Aktion (z. B. verschieben, kopieren, löschen) ausgeführt werden kann.

Eine Mehrfachauswahl ist durch gleichzeitiges Drücken der SHIFT- bzw. der STRG-Taste möglich. Außerdem kann die Tastenkombination STRG + A genutzt werden, um alle sichtbaren Elemente eines Verwaltungsbereichs (innerhalb der Baumansicht) oder alle Elemente innerhalb einer Tabelle (z. B. innerhalb der Aufgabenliste) zu selektieren.

Innerhalb der Baumansicht ist die Mehrfachselektion von Elementen auf den jeweils aktuellen Verwaltungsbereich beschränkt. Ist also bereits ein Element, z. B. in der Inhalte-Verwaltung markiert, kann danach kein Element mehr aus einem anderen Verwaltungsbereich selektiert werden.

Wird die Tastenkombination STRG + A innerhalb der Baumansicht des FirstSpirit JavaClients verwendet, werden nur die aktuell sichtbaren (expandierten) Elemente der Baumansicht markiert. Ist beispielsweise ein Ordner der Inhalte-Verwaltung nicht expandiert, sind die darunterliegenden Seiten, kein Bestandteil der Selektion.

4.11.1 Mehrfachselektion von Arbeitsabläufen

Die Mehrfachselektion von Arbeitsabläufen ermöglicht das Starten und Schalten eines Arbeitsablaufs auf einer Menge von Objekten.

Die gewünschten Objekte können dazu innerhalb der Baumansicht markiert werden. Anschließend wird das Kontextmenü wie gewohnt geöffnet und der gewünschte Arbeitsablauf selektiert.

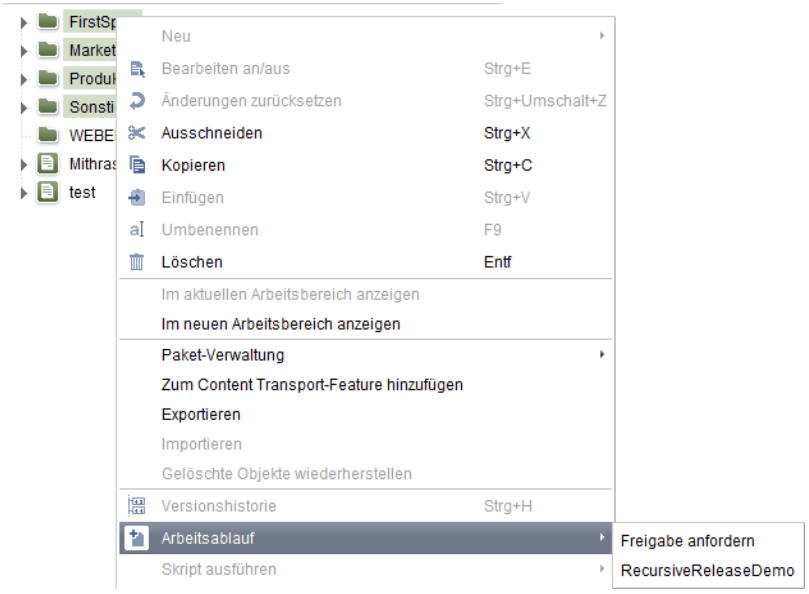


Abbildung 4-76: Starten eines Arbeitsablaufs auf mehreren Elementen

4.11.2 Voraussetzungen für das Starten und Weiterschalten

Das Starten oder Schalten eines Arbeitsablaufs kann nur dann erfolgen, wenn alle Elemente den gleichen Status des Arbeitsablaufs besitzen oder bisher noch kein Arbeitsablauf auf den selektierten Elementen gestartet wurde (siehe Abbildung 4-76).

Bei der Mehrfachauswahl von Elementen wird für jedes Element ermittelt, welche Arbeitsabläufe bzw. welche Transitionen eines Arbeitsablaufs über das Kontextmenü angezeigt werden können. Dabei wird beispielsweise berücksichtigt, ob:

- bereits ein Arbeitsablauf auf dem Element gestartet wurde,
- der Benutzer die erforderlichen Rechte besitzt, um den Arbeitsablauf auf diesem Element zu starten,
- ein Arbeitsablauf auf diesem Element gestartet werden darf,
- auf den Elementen bereits ein Arbeitsablauf gestartet wurde, die Elemente aber nicht den gleichen Status des Arbeitsablaufs erreicht haben.

Sind diese Anforderungen nur für ein Element der Mehrfachselektion nicht erfüllt, liefert das Kontextmenü für alle selektierten Elemente die Berechnung "nicht verfügbar".

Abbildung 4-77: Kontextmenü – nicht verfügbar

In diesem Fall sollte die Mehrfachselektion aufgehoben, die einzelnen Elemente erneut geprüft und ggf. erneut selektiert werden.

4.11.3 Mehrfachselektion über die Aufgabenliste

Neben der Mehrfachselektion über die Baumansicht können einmal gestartete Arbeitsabläufe auch über die Aufgabenliste (siehe Abbildung 4-78) oder über die Übersicht "Arbeitsabläufe" in der Vorlagen-Verwaltung weitergeschaltet werden (siehe Kapitel 4.1 Seite 123).

Die Aufgabenliste stellt alle noch nicht erledigten Aufgaben ("offene Aufgaben") und alle gestarteten Aufgaben ("initiierten Aufgaben") innerhalb einer tabellarischen Ansicht dar. Neben dem Namen des Arbeitsablaufs wird hier auch der aktuelle Status der jeweiligen Instanz des Arbeitsablaufs angezeigt.

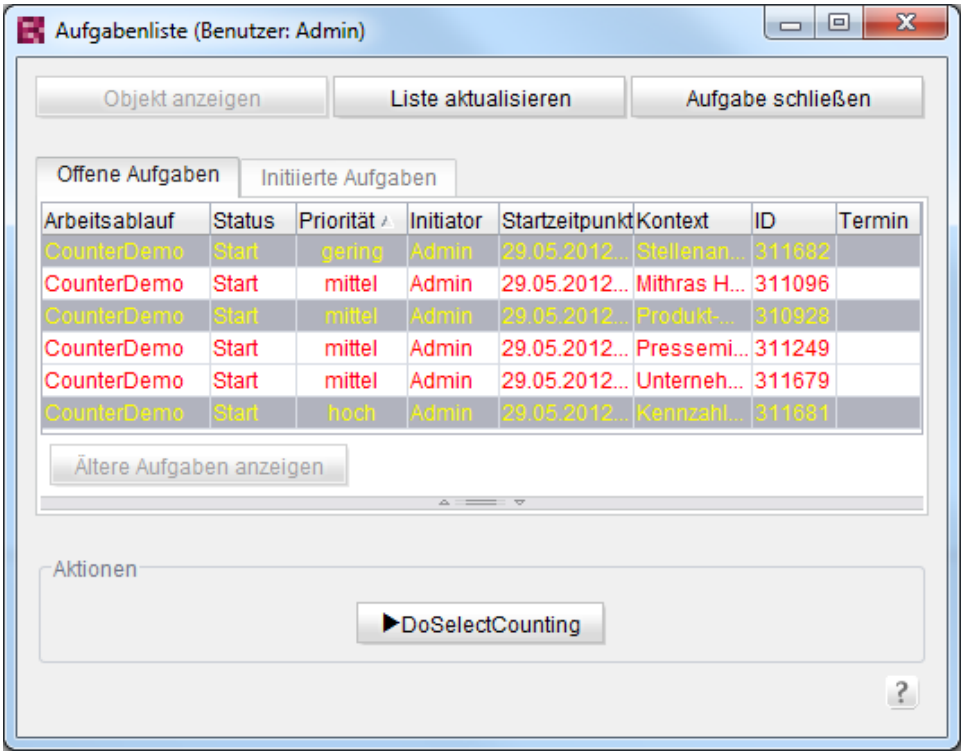


Abbildung 4-78: Multiselektion über die Aufgabenliste

Innerhalb der tabellarischen Aufgabenliste können mehrere Aufgaben selektiert werden. Sofern diese Aufgaben den gleichen Arbeitsablauf und den gleichen Status besitzen und der Benutzer die erforderlichen Rechte zum Ausführen des Arbeitsablaufs für die selektierten Elemente besitzt, werden im unteren Bereich der Aufgabenliste, die möglichen Aktionen angezeigt.

Die Aufgaben können so parallel weitergeschaltet werden. Anders als beim Starten über die Baumansicht, können in der Aufgabenliste auch mehrere Elemente aus unterschiedlichen Verwaltungsbereichen markiert und parallel geschaltet werden.

4.11.4 Mehrfachselektion über die Übersicht "Arbeitsabläufe"

Neben der Aufgabenliste gibt es innerhalb der Vorlagen-Verwaltung auf dem Knoten "Arbeitsabläufe" eine weitere Übersicht über alle bisher gestarteten Aufgaben (siehe Kapitel 4.1 Seite 123). Anders als in der Aufgabenliste, können die Aufgaben hier nach bestimmten Suchkriterien gefiltert und auch bereits geschlossene Aufgaben angezeigt werden (siehe Kapitel 4.1.1 Seite 125).

Innerhalb der Übersicht können mehrere Aufgaben selektiert werden. Ein direktes Weiterschalten der Arbeitsabläufe ist hier zwar nicht möglich, ein Klick auf den Button "Bearbeiten" öffnet jedoch die Aufgabenliste (siehe Kapitel 4.1.2 Seite 127). Die zuvor in der Übersicht markierten Elemente sind nun in der Aufgabenliste direkt selektiert und können dort weitergeschaltet werden (siehe Kapitel 4.11.3 Seite 208).

Neben dem Bearbeiten können über die Übersicht auch mehrere selektierte Aufgaben geschlossen werden (siehe Kapitel 4.1.3 Seite 129).

5 Änderungsverfolgung über Revisions-Metadaten

FirstSpirit bietet eine Möglichkeit zur Änderungsverfolgung über die FirstSpirit Access-API an. Über bestimmte API-Funktionen ist der Zugriff auf die Metadaten einer Revision möglich (siehe Kapitel 5.2 Seite 212). Die Revisions-Metadaten enthalten Informationen über die Art (Welche Änderungen haben stattgefunden?) und den Umfang (Welche Elemente wurden geändert?) einer Änderung im Projekt. Die Informationen, die über die Metadaten der Revision zur Verfügung gestellt werden, sind sehr feingranular. Bei inhaltlichen Änderungen kann beispielsweise ermittelt werden, welche Eigenschaften eines Elements verändert wurden, z. B. ob der Inhalt in einer bestimmten Redaktionssprache geändert wurde, ob ein Kindelement hinzugefügt oder entfernt wurde oder ob bestimmte Attribute, z. B. die Rechte auf dem entsprechenden Element, geändert wurden.

Über die erweiterten Revisionsinformationen können alle Änderungen, die von einer bestimmten Revision bis zu einer bestimmten Revision, innerhalb eines Projekts stattgefunden haben, ermittelt werden.

Ein Zugriff auf diese Informationen ist über die FirstSpirit Access-API, z. B. per BeanShell-Skript, möglich.

Die nachfolgenden Kapitel stellen Methoden vor, um eine oder mehrere Revisionen eines Projekts zu erhalten, die auf Änderungen untersucht werden sollen (siehe Kapitel 5.1 Seite 211) und Methoden, um die entsprechenden Änderungsinformationen zu ermitteln (siehe Kapitel 5.2 Seite 212). Abhängig vom jeweiligen Änderungstyp stehen dabei unterschiedliche Metadaten-Informationen zur Verfügung (siehe Kapitel 5.2.1 Seite 212).

Zusätzlich werden Beispiele zur Verwendung der Änderungsverfolgung im Projekt beschrieben.

Das erste Beispiel ermittelt alle Datenbank-Änderungen, die seit der zuletzt veröffentlichten Revision eines Projekts stattgefunden haben (siehe Kapitel 5.3 Seite 214).

Das zweite Beispiel ermittelt inhaltliche Änderungen, die zwischen einer zu definierenden Start-Revision und einer zu definierenden End-Revision im Projekt stattgefunden haben (siehe Kapitel 5.4 Seite 218).

Bei allen Code-Auszügen in diesem Kapitel handelt es sich um Fragmente, die nicht ausreichen, um das jeweils gesamte Skript zusammen zu stellen!

5.1 Revisionen holen

FirstSpirit arbeitet mit einem revisionsbasierten Repository. Dabei kommt eine spezielle Technik zur Verwaltung der zeitlichen Entwicklung der Daten zum Einsatz: Das so genannte Revisions-Management.

Eine Revision lässt sich als eine Art "Schnappschuss" des gesamten Repositories zu einem bestimmten Zeitpunkt vorstellen. Im Gegensatz zu einer Version, die sich in der Regel nur auf ein einzelnes Objekt bezieht, wird bei einer Revision der Gesamtzustand aller Objekte im Repository beschrieben.

Revisionen werden durch eine fortlaufende Nummerierung beschrieben (Revisions-ID), wobei es immer genau eine aktuelle Revision für das gesamte Repository gibt. Wenn ein Repository bearbeitet wird, werden alle vorgenommen Änderungen mit einer neuen Revisionsnummer verknüpft. Die Revisionsnummer ergibt sich aus der um eins erhöhten, zuletzt aktuellen Revisionsnummer des Gesamt-Repositories. Alle nicht veränderten Objekte behalten ihre alten Revisionsnummern bei. Wird ein Objekt verändert, wird es im Repository nicht überschrieben, sondern als neues Objekt (mit einer höheren Revisionsnummer) eingefügt.

Um festzustellen, in welchem Zeitraum bestimmte Änderungen innerhalb des Projekts stattgefunden haben, muss zunächst die entsprechende Revision des Repositories geholt werden.

Die Revision kann direkt über das Projekt geholt werden. Dabei kann entweder die gewünschte eindeutige Revisions-ID, z. B.:

```
project.getRevision(revisionId);
```

oder das Datum der gewünschten Revision übergeben werden, z. B.:

```
project.getRevision(context.getStartTime());
```

Das übergebene Datum muss nicht eindeutig einer Revision zugeordnet sein. Es kann ein beliebiger Datumswert übergeben werden. Falls zu diesem Datum eine Revision existiert, wird diese zurückgeliefert, andernfalls liefert die Methode die nächst kleinere Revision zurück.

Eine Auswahl von Revisionen innerhalb eines bestimmten Zeitraums kann über die Methode:

```
project.getRevisions(Revision from, Revision to, int maxCount,
Filter<Revision> filter);
```

bereitgestellt werden. Dabei werden zwei Revisionen übergeben. Die erste Revision ("from") definiert die untere und die zweite Revision ("to") definiert die obere Revisionsgrenze. Neben diesen beiden Revisionen, werden alle Revisionen

geliefert, die eine höhere Revisions-ID als die untere Revisionsgrenze und eine geringere Revisions-ID als die obere Revisionsgrenze besitzen.

Die jeweils aktuellste Revision, kann über:

```
getRevision(new Date());
```

geholt werden.

```
start = project.getRevision(context.getStartTime());
end = project.getRevision(new Date());
revisions = project.getRevisions(start, end, 0, null);
```

Optional kann noch der Parameter "maxCount" übergeben werden, der die Anzahl der zurück gelieferten Revisionen auf einen Höchstwert begrenzt, und – ebenfalls optional – ein Filter zur weiteren Eingrenzung.

5.2 Änderungen in einer Revision ermitteln

Über die Revisionen (siehe Kapitel 5.1 Seite 211) können anschließend die Metadaten, mit erweiterten Informationen zu den Änderungen geholt werden:

```
revision.getMetaData();
```

Die Metadaten verwalten unterschiedliche Informationen, die abhängig von der Art der jeweiligen Änderung sind (siehe Kapitel 5.2.1 Seite 212). Dabei werden nicht nur sprachabhängige inhaltliche Änderungen eines Elements berücksichtigt, sondern auch strukturelle Änderungen (z. B. Verschieben) oder eine Änderung der Element-Attribute (z. B. Name, Rechtedefinition, usw.)(siehe Kapitel 5.2.2 Seite 213).

5.2.1 Änderungstyp ermitteln

Die Änderungen, die in einer Revision stattgefunden haben, können über:

```
metaData.getOperation();
```

geholt werden.

Die gelieferte Revisionsoperation (`RevisionOperation`) liefert beispielsweise Informationen zum Änderungstyp (`RevisionOperation.OperationType`):

```
operation.getType();
```

Dabei stehen für unterschiedliche Projektinhalte unterschiedliche Änderungstypen zur Verfügung.

Für Inhalte vom Typ `IDProvider` sind folgende Änderungstypen möglich:

- *CREATE* ein Objekt wurde neu im Projekt angelegt
- *MODIFY* ein Objekt wurde im Projekt geändert
- *MOVE* ein Objekt wurde im Projekt verschoben
- *DELETE* ein Objekt wurde im Projekt gelöscht
- *RELEASE* ein Objekt wurde im Projekt freigegeben
- *SERVER_RELEASE* ein Objekt wurde auf dem Server freigegeben

Die entsprechende Revisions-Operation (z. B. `ModifyOperation`) liefert ein Objekt vom Typ `BasicElementInfo` mit weiteren Informationen zum betroffenen Objekt zurück (z. B. den UniqueIdentifier).

Für Inhalte vom Typ `Entity` sind folgende Änderungstypen möglich:

- *CONTENT_COMMIT* es wurden Datenbank-Inhalte geändert

Die entsprechende Revisions-Operation (z. B. `ContentOperation`) liefert ein Objekt vom Typ `EntityInfo` mit weiteren Informationen zu den betroffenen Datensätzen zurück (z. B. die ID des Datensatzes oder die ID des zugehörigen Datenbankschemas).

5.2.2 Geänderte Elemente ermitteln

Abhängig von der jeweiligen Änderungsoperation können weitere Informationen zu den Änderungen abgerufen werden, beispielsweise welche Datensätze innerhalb des Projekts freigegeben wurden (für Operation-Type: *CONTENT_COMMIT*):

```
operation.getReleasedEntities();
```

oder z. B. welche Inhalte neu im Projekt angelegt wurden (für Operation-Type: *CREATE*):

```
operation.getCreatedElement()
```

Weitere Methoden befinden sich in den Beispielen der beiden nachfolgenden Kapitel (siehe Kapitel 5.3 und Kapitel 5.4).

Für eine Übersicht aller verfügbaren Methoden siehe Dokumentation zur FirstSpirit Access-API [5].

[5] Über die FirstSpirit Online Dokumentation im Bereich Vorlagenentwicklung – FirstSpirit API

5.3 Änderungen seit der letzten Veröffentlichung

Im ersten Beispiel sollen die Änderungen, die seit dem Zeitpunkt der letzten Veröffentlichung im Projekt vorgenommen wurden, dargestellt werden. Im nachfolgenden Beispiel geht es um ein Projekt, in dem Gutachten über die Datenquellen-Verwaltung von FirstSpirit verwaltet werden. Bei jeder Veröffentlichung des Projekts soll eine Übersicht der Änderung an den Datensätzen bereitgestellt werden. Zum Ermitteln der Änderungen wird innerhalb der Veröffentlichungsaufträge zunächst ein Post-Deployment-Skript angelegt:

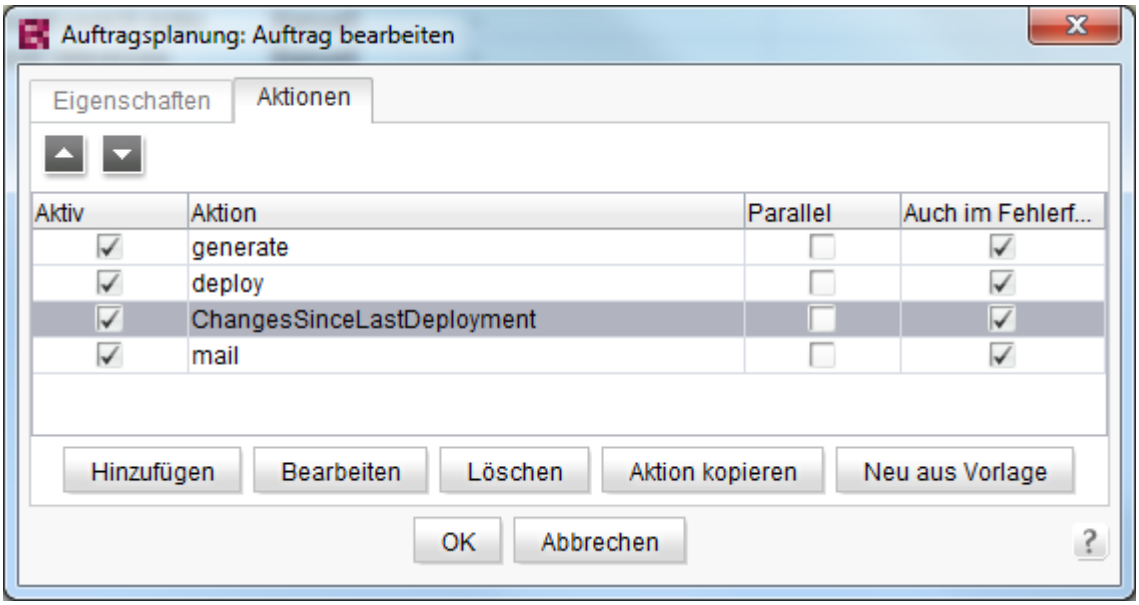


Abbildung 5-1: Konfiguration Post-Deployment-Skript

Das Skript ermittelt zuerst die ID der Revision, die zum Zeitpunkt der letzten Veröffentlichung des Projekts aktuell war:

```
task = context.getTask();

lastExecutionRevisionId = (Long) context.getVariable(task.getName() +
".revision");

if (lastExecutionRevisionId != null) {

    context.logInfo("revision of last execution=" +
lastExecutionRevisionId);

    revId = lastExecutionRevisionId.longValue();}
```

Anschließend werden alle Revisionen des Projekts seit der letzten Veröffentlichung geholt. Als untere Revisionsgrenze ("startRev") wird dabei die Revision mit der soeben ermittelten Revisions-ID geholt. Als obere Revisionsgrenze wird die aktuelle Revision zum Startzeitpunkt der Veröffentlichung ermittelt:

```
startRev = project.getRevision(revId);
endRev = project.getRevision(context.getStartTime());
context.logInfo("startRev=" + startRev.id + ", endRev=" + endRev.id);
if (startRev.id == endRev.id) {
    context.logInfo("no changes detected");
}
revisions = project.getRevisions(startRev, endRev, 0, null);
```

Innerhalb einer Schleife werden anschließend alle ermittelten Revisionen auf Änderungen untersucht:

```
checkChanges(revisions) {
    for (revision : revisions) {
           metaData = revision.getMetaData();
           operation = metaData.getOperation();
           if (operation != null) {
                  type = operation.getType();
                  switch (type) {
                  case RevisionOperation.OperationType.CONTENT_COMMIT:
                         ..
                         ..
                         break;
                  }
           }
    }
```

Dabei sollen hier nur Änderungen an Datenbankinhalten – also vom Operation-Type CONTENT_COMMIT – berücksichtigt werden und zwar nur die neu angelegten und die geänderten Datensätze einer bestimmten Datenbank-Tabelle.

Über:

```
createdEntities = operation.getCreatedEntities();
releasedEntities = operation.getReleasedEntities();
```

werden zunächst alle neu erzeugten und alle freigegebenen Datensätze ermittelt. Anschließend wird diese Auswahl auf eine bestimmte Datenbank-Tabelle (hier: `MyEntityTypName`) eingeschränkt:

```
ENTITY_TYPE = "MyEntityTypName";
if (ENTITY_TYPE.equals(created.getEntityTypeName())){
..
}
if (ENTITY_TYPE.equals(released.getEntityTypeName())) {
..
}
```

Zusammenhängend:

```
case RevisionOperation.OperationType.CONTENT_COMMIT:
    createdEntities = operation.getCreatedEntities();
    for (created : createdEntities) {
           if (ENTITY_TYPE.equals(created.getEntityTypeName())) {
           createdCertificates.put(created.getEntityId(), revision);
           context.logInfo("\t created entity " + created.getEntityId() +
" in revision " + getRevisionString(revision));
           }
    }
    releasedEntities = operation.getReleasedEntities();
    for (released : releasedEntities) {
           if (ENTITY_TYPE.equals(released.getEntityTypeName())) {
           releasedCertificates.put(released.getEntityId(), revision);
           context.logInfo("\t released entity " + released.getEntityId()
+ " in revision " + getRevisionString(revision));
           }
    }
    break;
```

Anhand der IDs der ermittelten geänderten und freigegeben Datensätze ("created" und "released") werden die benötigten Informationen (z. B. die Gutachtennummern) geholt und anschließend zur weiteren Verwendung gespeichert.

```
context.setProperty("created", createdList);
context.setProperty("updated", updatedList);
```

Dabei sind die über `context.setProperty(..)` gespeicherten Werte nur innerhalb des aktuellen Auftrags persistent, können also in einer nachfolgenden Aktion innerhalb des Auftrags mit `context.getProperty(..)` weiterverwendet werden. In diesem Beispiel werden diese Inhalte in der nachfolgenden Aktion "Mail" (vgl. Abbildung 5-1), innerhalb der Mailvorlage weiterverwendet:

```
Hallo,
$CMS_SET(created, #context.getProperty("created"))$$CMS_SET(updated,
#context.getProperty("updated"))$
es wurden neue$CMS_IF(created !=
```

```
null)$($CMS_VALUE(created.size)$)$CMS_END_IF$ und
geänderte$CMS_IF(updated != null)$($CMS_VALUE(updated.size)$)$CMS_END_IF$
Gutachten auf

http://www.gutachten-online.de

veröffentlicht.

$CMS_IF(created.size > 0)$Neue Gutachten:

=====================

$CMS_FOR(entity, created)$ * $CMS_VALUE(entity.Gutachtennr)$
($CMS_VALUE(entity.Datum.format("dd.MM.yy"))$) -
$CMS_VALUE(entity.Kennzeichen)$

$CMS_END_FOR$$CMS_END_IF$


$CMS_IF(updated.size > 0)$Aktualisierte Gutachten:

=====================

$CMS_FOR(entity, updated)$ * $CMS_VALUE(entity.Gutachtennr)$
($CMS_VALUE(entity.Datum.format("dd.MM.yy"))$) -
$CMS_VALUE(entity.Kennzeichen)$

$CMS_END_FOR$$CMS_END_IF$

--

Dies ist eine automatisch generierte E-Mail, die nach der
Veröffentlichung neuer Gutachten verschickt wird.

Bei Fragen wenden Sie sich bitte an info@gutachten-online.de
```

Die Vorlage erzeugt nun bei der Ausführung eines Veröffentlichungsauftrags eine Email mit den neu erzeugten und den geänderten Inhalten:

Beispiel (Mail):

```
Hallo,


es wurden neue(4) und geänderte(2) Gutachten auf

http://www.gutachten-online.de

veröffentlicht.


Neue Gutachten:

=====================

 * AZ33048/D (10.08.10) - DO-WZ 1234

 * AZ45134/D (10.08.10) - DO-XY 4321

 * AZ46200/D (11.08.10) - EN-AA 1111

 * AZ50261/D (13.08.10) - BO-YZ 5566


Aktualisierte Gutachten:

=====================

 * AZ44356/D (10.08.10) - DO-ZZ 3388
```

```
 * AZ47709/D (05.05.08) - D-YY 9999
--
Dies ist eine automatisch generierte E-Mail, die nach der
Veröffentlichung neuer Gutachten verschickt wird.
Bei Fragen wenden Sie sich bitte an info@gutachten-online.de
```

Inhalte, die über `context.setVariable(..)` gespeichert wurden, sind auch über die Ausführung des aktuellen Auftragslaufs hinweg persistent (im Gegensatz zum Speichern von Inhalten über `context.setProperty(..)`). Diese Möglichkeit wird im Beispiel verwendet, um die Revision zum Zeitpunkt des aktuellen Auftrags zu speichern:

```
context.setVariable(task.getName() + ".revision",
new Long(endRev.getId()));
```

Beim Starten des nächsten Auftrags können diese Informationen dann verwendet werden, um die ID der Revision zu holen, die zum Zeitpunkt der letzten Veröffentlichung des Projekts aktuell war:

```
lastExecutionRevisionId = (Long) context.getVariable(task.getName() +
".revision");
```

Das vollständige Skript und die hier beschriebenen Vorlagen können bei Bedarf über das FirstSpirit Helpdesk angefordert werden.

5.4 Änderungen zwischen zwei Revisionen

Im zweiten Beispiel können die Revisionen komfortabel über eine GUI ausgewählt werden. Dazu muss zunächst ein neues Skript in der Vorlagen-Verwaltung des Projekts angelegt werden.

Im Formularbereich des Skripts können Eingabekomponenten zur Auswahl eines Start- und eines End-Datums für die gewünschten Revisionsgrenzen konfiguriert werden:

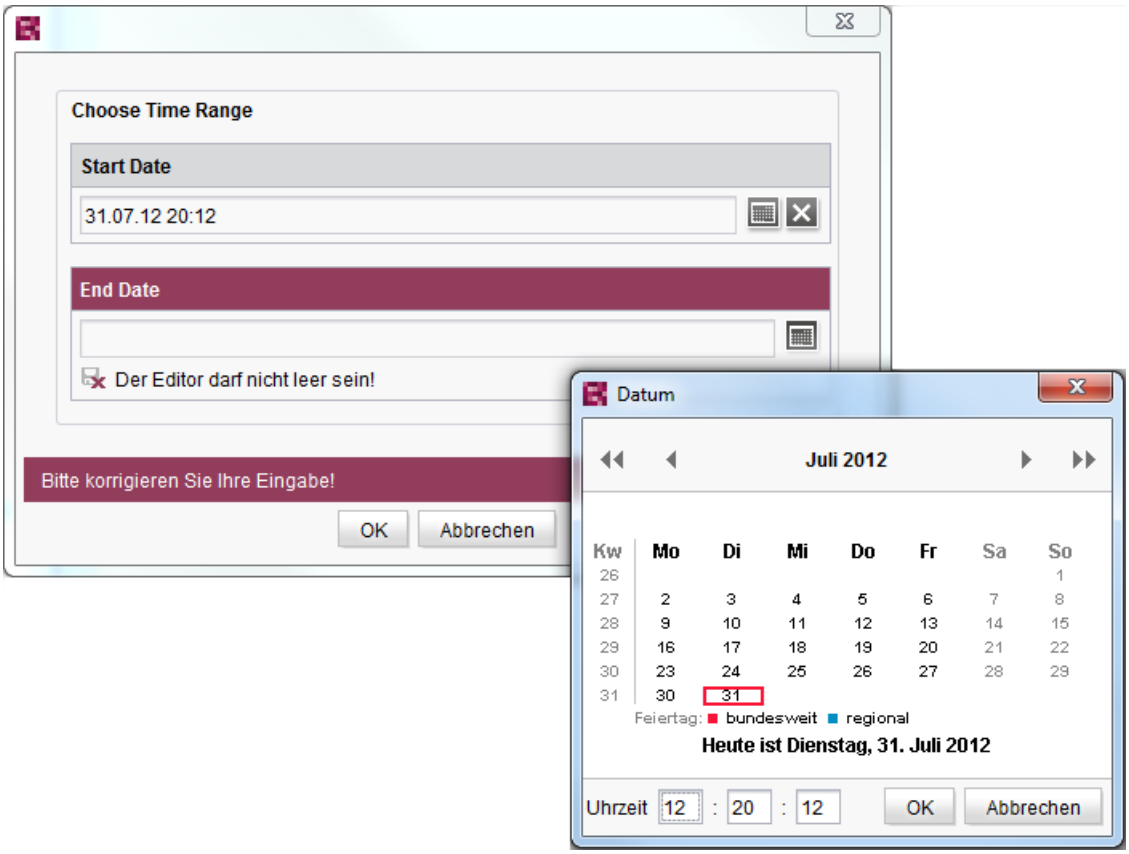


Abbildung 5-2: GUI zur Auswahl der Revisionsgrenzen

Die XML-Datei zur Konfiguration des Formularbereichs kann bei Bedarf über FirstSpirit Helpdesk angefordert werden.

Analog zum ersten Beispiel können über die ausgewählten Daten anschließend die entsprechenden Revisionen geholt werden:

```
data = context.showForm();
if (data != null) {
    context.logInfo("data=" + data);
    from = data.get(context.getProject().getMasterLanguage(),
           "from").get();
    to = data.get(context.getProject().getMasterLanguage(),
         "enddate").get();

    if (from != null) {
           context.logInfo(from  + " -- " + to);
           start = project.getRevision(from);
           end = project.getRevision(to);
           context.logInfo("startRev=" + start.id + ", endRev=" + end.id);
```

```
            if (start.id <= end.id) {
                  revisions = project.getRevisions(start, end, 0, null);
            } else {
                  revisions = project.getRevisions(end, start, 0, null);
            }
            context.logInfo("found '" + revisions.size() + "' revisions ->
first=" + revisions.get(0) + ", last=" + revisions.get(revisions.size() -
1));
            checkChanges(revisions);
      }}
```

In `checkChanges` (siehe Kapitel 5.3 Seite 214) werden die Änderungen ermittelt, die innerhalb dieser Revisionen stattgefunden haben. In diesem Beispiel werden Änderungen an Projektinhalten vom Operation-Type DELETE und CREATE (innerhalb der Inhalte-Verwaltung) berücksichtigt. Außerdem werden auch hier Änderungen an Datenbankinhalten des Projekts ermittelt:

Änderung durch Entfernen von Elementen in der Inhalte-Verwaltung:

```
case RevisionOperation.OperationType.DELETE:
    deleteRoot = operation.getDeleteRootElement();
    if (deleteRoot.getStoreType() == Store.Type.PAGESTORE) {
    // include only pagestore
    context.logInfo("found delete in pagestore (deleted node=" +
     deleteRoot.getUid() + ") in revision=" +
     getRevisionString(revision));
    }
    break;
```

Änderung durch Hinzufügen von Elementen im Projekt:

```
case RevisionOperation.OperationType.CREATE:
    created = operation.getCreatedElement();
    parent = operation.getParent();
     context.logInfo("found created element in store '" +
      created.getStoreType() + "' (created node=" +
      created.getUid() + ", parent node=" + parent.getUid() + ") in
      revision=" + getRevisionString(revision));
    break;
```

Änderung durch Anlegen, Löschen, Ändern oder Freigeben von Datenbank-Inhalten:

```
case RevisionOperation.OperationType.CONTENT_COMMIT:
    created = operation.getCreatedEntities();
    changed = operation.getChangedEntities();
```

```
    deleted = operation.getDeletedEntities();
    released = operation.getReleasedEntities();
    context.logInfo("found content changes in revision=" +
     getRevisionString(revision));
    context.logInfo("\t created entities(" + created.size() + ") "
     + created);
    context.logInfo("\t changed entities(" + changed.size() + ") "
     + changed);
    context.logInfo("\t deleted entities(" + deleted.size() + ") "
     + deleted);
    context.logInfo("\t released entities(" + released.size() + ")
     " + released);
    break;
```

Die Ausgabe des Skripts erfolgt über die Javakonsole:

```
..
INFO  20.05.2008 15:44:50.317
startRev=6804, endRev=7677
found created element in store 'PAGESTORE' (created
node=Testpage_131, parent node=Test_6C22873) in revision=7335
 - Thu May 15 16:02:51 CEST 2008 (importStoreElement) - Admin -
CREATE
..
```

6 Serverseitige Freigabe

Neben der Freigabe über einen Arbeitsablauf, können alle Objekte in FirstSpirit serverseitig über die Access-API freigegeben werden. Dazu existieren Methoden, um die unterschiedlichen Freigabeeinstellungen für ein Objekt zu definieren. So können über die spezifische Freigabe weitere, vom aktuellen Objekt abhängige Objekte freigegeben werden, z. B. die vollständige Vaterkette oder die Kindelemente des freizugebenden Objekts.

Allgemein wird zwischen folgenden Freigabeoptionen unterschieden:

- Standard-Freigabe (siehe Kapitel 6.1 Seite 222):
  Freigabe für das freizugebende Objekt inklusive zusätzlicher, festgelegter Freigabeoptionen für den Standardfall. Diese vorgegebenen Freigabeoptionen sind objektabhängig unterschiedlich. So werden über die Standard-Freigabeoptionen einer Seite in der Inhalte-Verwaltung, zusätzlich die untergeordneten Absätze und die noch niemals freigegeben Vaterelemente freigegeben. Die Standard-Freigabe einer Seitenreferenz in der Struktur-Verwaltung berücksichtigt dagegen nur die Seitenreferenz selbst. Die Standard-Freigabeoptionen können nicht geändert werden.
- Spezifische Freigabe (siehe Kapitel 6.2 Seite 223):
  Freigabe für das freizugebende Objekt inklusive optionaler Freigabeoptionen, die durch den Benutzer festgelegt werden. Die unterschiedlichen Freigabeoptionen können beliebig miteinander kombiniert werden, um eine umfangreiche Freigabe innerhalb kurzer Zeit zu realisieren. Die Freigabe aller am Freigabeprozess beteiligten Objekte ist aber unter Umständen nicht in allen Fällen erwünscht und sollte daher mit Bedacht ausgeführt werden.

6.1 Standard-Freigabe

Über diese Option wird die Freigabe für das aktuelle Objekt (z. B. Seite oder Ordner der Inhalte-Verwaltung), inklusive festgelegter, objektabhängiger Freigabeoptionen für den Standardfall, ausgeführt.

Die direkte Freigabe eines Objektes wird über die folgende API-Methode ausgeführt:

```
AccessUtil.release(IDProvider toRelease, boolean checkOnly)
```

Übergabe-Parameter:

`toRelease`: Freizugebendes Element

`checkOnly`: Wird der Wert `true` übergeben, wird die Standard-Freigabe nur getestet. Die freizugebenden Objekte werden dabei nicht in den Freigabestand überführt. Stattdessen wird die Standard-Freigabe durchlaufen, um z. B. Fehler vor der realen Freigabe eines Objekts aufzudecken.

Rückgabe-Parameter:

```
ServerActionhandle<? extends ReleaseProgress,Boolean >
```

Die serverseitige Freigabe liefert ein `ServerActionHandle` zurück, das alle Informationen über den Freigabeprozess beinhaltet und beispielsweise den Status der Freigabe oder die Log-Infos enthält.

6.2 Spezifische Freigabe

Die spezifische Freigabe berücksichtigt, abhängig von den definierten Freigabeparametern, noch weitere (abhängige) Objekte innerhalb des Freigabeprozesses.

- **Erreichbarkeit sicherstellen (Vaterkette):** Ausgehend vom ausgewählten Objekt werden alle neuen (niemals freigegebenen) übergeordneten Knoten ebenfalls freigegeben (siehe Kapitel 6.2.4 Seite 230). Diese Option ist beispielsweise dann sinnvoll, wenn eine neue Seite unterhalb eines neuen Ordners in der Inhalte-Verwaltung angelegt wurde und beide gemeinsam freigegeben werden sollen. Im Gegensatz zur rekursiven Freigabe würden andere neue Seiten unterhalb des Ordners nicht freigegeben. Während die Kombination dieser Option mit der Option "rekursive Freigabe" auf die aktuelle Verwaltung beschränkt bleibt (siehe Kapitel 6.2.5 Seite 232), wirkt sie sich in Kombination mit der Option "abhängige Freigabe" auch auf die Vaterketten der abhängigen Objekte und damit auf weitere Verwaltungen aus (siehe Kapitel 6.2.6 Seite233).
- **Rekursiv freigeben:** ausgehend vom ausgewählten Objekt werden alle untergeordneten Knoten ebenfalls freigegeben. Diese Selektion ist beispielsweise dann sinnvoll, wenn unterhalb eines Ordners in der Inhalte-Verwaltung viele Seiten geändert wurden und nun alle Änderungen gemeinsam freigegeben werden sollen. Diese Option bleibt auf den aktuellen Verwaltungsbereich beschränkt (vgl. Kapitel 6.2.1 Seite 226).
- **Nur neue abhängige Objekte freigeben:** ausgehend vom ausgewählten Objekt werden zusätzlich alle Objekte freigegeben, die abhängig vom ausgewählten Objekt sind (z. B. ein Medium, das in einer Bildeingabekomponente verwendet

wird) und die noch nie freigegeben wurden (neu angelegte Objekte). Wird diese Freigabeoption mit weiteren Optionen (z. B. der Freigabe der Vaterkette) kombiniert, wirkt sich die abhängige Freigabe auch auf andere am Freigabeprozess beteiligte Objekte und Verwaltungen aus.

- **Neue und geänderte abhängige Objekte freigeben:** ausgehend vom ausgewählten Objekt werden zusätzlich alle Objekte freigegeben, die abhängig vom ausgewählten Objekt sind (z. B. ein Medium, das in einer Bildeingabekomponente verwendet wird.). Dabei werden sowohl Objekte berücksichtigt, die noch nie freigegeben wurden (neu angelegte Objekte) als auch Objekte, die nach einer bereits erfolgten Freigabe zwischenzeitlich erneut bearbeitet wurden (geänderte Objekte). Wird diese Freigabeoption mit weiteren Optionen (z. B. der Freigabe der Vaterkette) kombiniert, wirkt sich die abhängige Freigabe auch auf andere, am Freigabeprozess beteiligte Objekte und Verwaltungen aus.

Die spezifische Freigabe eines Objektes wird über die folgende API-Methode ausgeführt:

```
AccessUtil.release(IDProvider releaseStartNode, boolean checkOnly,
boolean releaseParentPath, boolean recursive,
IDProvider.DependentReleaseType dependentType)
```

Übergabe-Parameter:

`releaseStartNode`: Startknoten für die Freigabe

`checkOnly`: Wird der Wert `true` übergeben, wird die spezifische Freigabe nur getestet. Die freizugebenden Objekte werden dabei nicht in den Freigabestand überführt. Stattdessen werden die definierten Freigabeoptionen durchlaufen, um z.B. Fehler vor der realen Freigabe aufzudecken.

`releaseParentPath`: Wird der Wert `true` übergeben, wird die vollständige Vaterkette des freizugebenden Objekts ermittelt und alle zuvor niemals freigegebenen Objekte ebenfalls freigegeben. Wird die Option `releaseParentPath=false` gesetzt, wird die Vaterkette nicht freigegeben, die freizugebenden Elemente werden aber der Release-Kindliste des Vaterknotens hinzugefügt. Dabei gilt:

- Bei *geänderten* Vaterknoten: Das freizugebende Objekt ist im Freigabe-Stand erreichbar. Das Vaterelement wird aber nicht freigegeben.
- Bei *neuen* Vaterknoten: Da der Vaterknoten niemals freigegeben wurde, ist auch das freizugebende Objekt im Freigabe-Stand nicht erreichbar. Das kann zu ungültigen Referenzen im Freigabe-Stand führen (siehe Kapitel 6.2.4 und Kapitel 6.2.5).

`recursive`: Wird der Wert `true` übergeben, werden rekursiv alle Kindelemente des freizugebenden Objekts ermittelt und ebenfalls freigegeben. Wird der Wert `false`

übergeben werden die Kindelemente bei der Freigabe nicht berücksichtigt (siehe Kapitel 6.2.1, Kapitel 6.2.3 und Kapitel 6.2.5).

`dependentType`: Über diesen Parameter werden abhängige Objekte des freizugebenden Objekts ermittelt und ebenfalls freigegeben. Wird beispielsweise auf einer Seite ein Medium referenziert, kann dieses Medium bei der spezifischen Freigabe der Seite ebenfalls direkt freigegeben werden. Es können folgende Abhängigkeiten berücksichtigt werden (siehe Kapitel 6.2.2, Kapitel 6.2.3 und Kapitel 6.2.6):

- `DEPENDENT_RELEASE_NEW_AND_CHANGED`: neue und geänderte abhängige Objekte werden berücksichtigt.
- `DEPENDENT_RELEASE_NEW_ONLY`: nur neu angelegte (noch nie freigegebene Objekte) werden berücksichtigt
- `NO_DEPENDENT_RELEASE`: abhängige Objekte werden nicht berücksichtigt und müssen ggf. gesondert freigegeben werden (Standardeinstellung).

Die unterschiedlichen Freigabeoptionen können beliebig miteinander kombiniert werden, um eine umfangreiche Freigabe innerhalb kurzer Zeit zu realisieren. Die Freigabe aller am Freigabeprozess beteiligten Objekte ist aber unter Umständen nicht in allen Fällen erwünscht und sollte daher mit Bedacht ausgeführt werden.

Die serverseitige Freigabe soll daher in den folgenden Kapiteln anhand einiger Beispiele erläutert werden.

Rückgabe-Parameter:

```
ServerActionhandle<? extends ReleaseProgress,Boolean >
```

Die serverseitige Freigabe liefert ein `ServerActionHandle` zurück, das alle Informationen über den Freigabeprozess beinhaltet.

6.2.1 Rekursive Freigabe

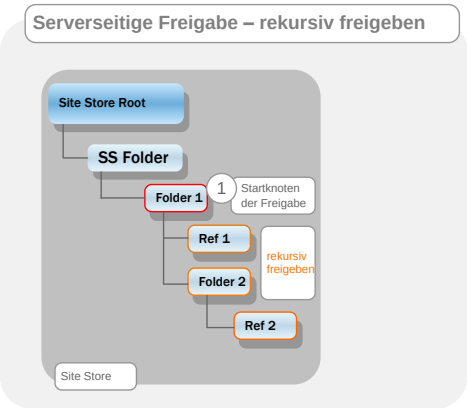


Abbildung 6-1: Serverseitige Freigabe – rekursiv

Für den Aufruf von `AccessUtil.release(...)` wurden die folgenden Parameter gesetzt:

```
releaseStartNode:  Folder 1
releaseParentPath: false
boolean recursive:  true
DependentReleaseType: NO_DEPENDENT_RELEASE
```

Der ausgewählte Startknoten für die Freigabe ist die Menüebene "Folder 1".

Rekursive Freigabe: Am Startpunkt der Freigabe "Folder 1" wird die Option `recursive` ausgewertet. Die rekursive Freigabe wirkt sich *ausschließlich* auf die Kindelemente des Startpunkts der Freigabe aus. Im Beispiel aus Abbildung 6-1 werden durch die Option also die Kindelemente "Ref 1", "Folder 2" und "Ref 2" freigegeben.

Rekursive Freigaben weiterer abhängiger Elemente werden – auch in Kombination mit anderen Freigabeoptionen – nicht ausgeführt. Die rekursive Freigabe wirkt sich also *nicht* auf die Freigabe von Kindelementen abhängiger Objekte in anderen Verwaltungen aus.

6.2.2 Abhängige Freigabe

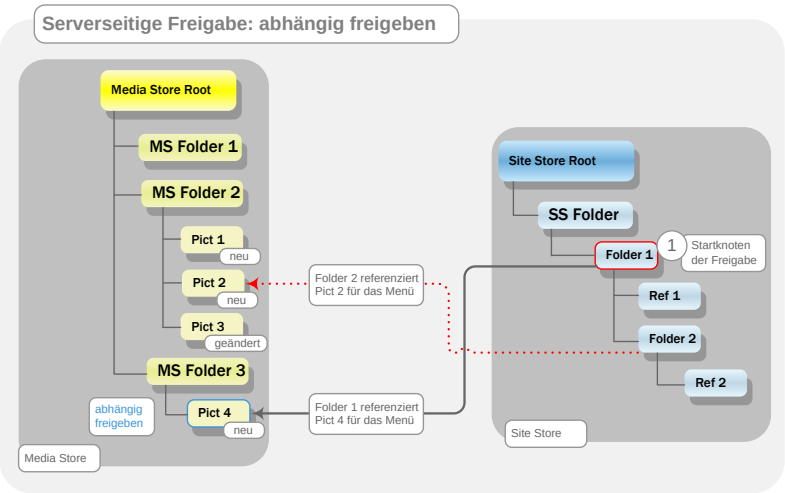


Abbildung 6-2: Serverseitige Freigabe – nur neue oder neue und geänderte freigeben

Für den Aufruf von `AccessUtil.release(...)` wurden die folgenden Parameter gesetzt:

```
releaseStartNode:  Folder 1
releaseParentPath: false
boolean recursive:  false
DependentReleaseType:
DEPENDENT_RELEASE_NEW_AND_CHANGED||DEPENDENT_RELEASE_NEW_ONLY
```

Der ausgewählte Startknoten für die Freigabe ist die Menüebene "Folder 1".

Abhängige Freigabe: Die Optionen `DEPENDENT_RELEASE_NEW_ONLY` und `DEPENDENT_RELEASE_NEW_AND_CHANGED` wirken sich grundsätzlich auf *alle* abhängigen Objekte in der Inhalte-, der Struktur- und der Medien-Verwaltung aus. Diese Freigabeoption betrifft damit nicht nur den Starknoten, sondern alle Objekte, die während des Freigabeprozesses betrachtet werden. Im Beispiel aus Abbildung 6-2 werden durch die Option alle ausgehenden Referenzen der Menüebene "Folder 1" untersucht und freigegeben. Ist nur die abhängige Freigabe aktiviert (ohne rekursive Freigabe) würde nur "Pict 4" abhängig freigegeben werden (siehe Abbildung 6-2), sind weitere Freigabeoptionen aktiviert, kann die Freigabe jedoch wesentlich umfangreicher sein (siehe Kapitel 6.2.3 Seite228, Kapitel 6.2.6 Seite 233 und Kapitel 6.2.7 Seite 235).

Alle ausgehenden Referenzen für die abhängige Freigabe werden nur in einer Richtung vollständig berücksichtigt. Sollen alle abhängigen Objekte im Freigabeprozess enthalten sein, muss die Freigebe also in einer bestimmten Reihenfolge erfolgen (siehe Kapitel 6.2.8 Seite 236).

6.2.3 Abhängige Freigabe mit rekursiver Freigabe

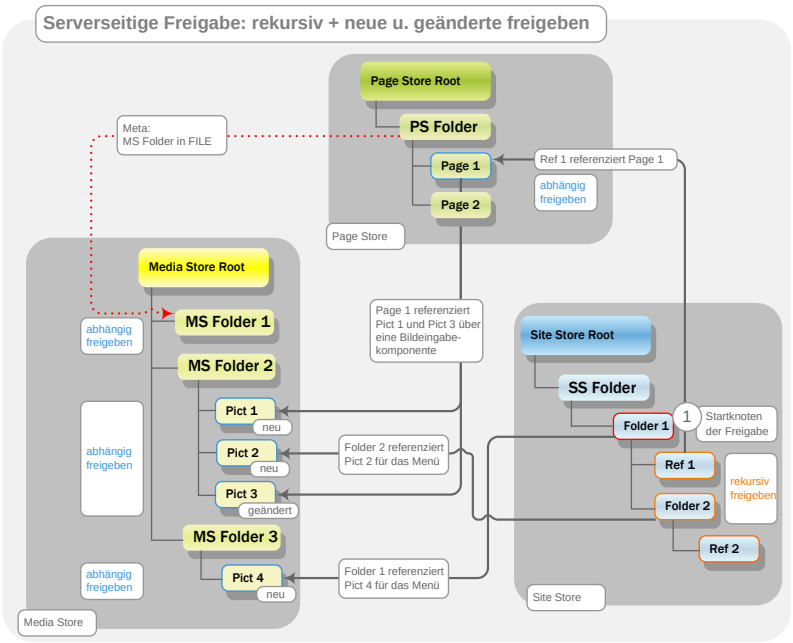


Abbildung 6-3: Serverseitige Freigabe – rekursiv und abhängig freigeben

Für den Aufruf von `AccessUtil.release(...)` wurden die folgenden Parameter gesetzt:

```
releaseStartNode:  Folder 1
releaseParentPath: false
boolean recursive:  true
DependentReleaseType:
DEPENDENT_RELEASE_NEW_AND_CHANGED||DEPENDENT_RELEASE_NEW_ONLY
```

Der ausgewählte Startknoten für die Freigabe ist die Menüebene "Folder 1".

Abhängige Freigabe und rekursive Freigabe: Werden die Optionen `DEPENDENT_RELEASE_NEW_ONLY` oder `DEPENDENT_RELEASE_NEW_AND_CHANGED` mit der Option `recursive` kombiniert, wirkt sich die abhängige Freigabe auch auf alle Objekte aus, die unterhalb des Startknotens liegen. Im Beispiel aus Abbildung 6-3 werden damit nicht nur die ausgehenden Referenzen der Menüebene "Folder 1" untersucht (vgl. Kapitel 6.2.2 Seite 227), sondern auch die ausgehenden Referenzen

der darunterliegenden Kindobjekte:

- Bezogen auf das Beispiel wird damit wird auch die unterhalb von "Folder 1" liegenden "Ref 1" untersucht, die eine Referenz in die Inhalte-Verwaltung besitzt. Durch die rekursive Freigabe wird die Seitenreferenz "Ref 1" freigegeben, durch die abhängige Freigabe wird zusätzlich die Seite "Page 1" freigegeben.
- Die Menüebene "Folder_2", die über die rekursive Freigabeoption ein Bestandteil des Freigabeprozesses geworden ist, besitzt eine Referenz auf ein Medium in der Medien-Verwaltung. Durch die rekursive Freigabe wird der Ordner "Folder 2" sowie die untergeordnete Seitenreferenz "Ref 2" freigegeben. Durch die abhängige Freigabe wird zusätzlich das referenzierte Medium "Pict 2" freigegeben.
- Die Seite "Page 1" die abhängig freigegeben wurde, besitzt ebenfalls ausgehende Referenzen in die Medien-Verwaltung. Die referenzierten Medien "Pict 1" und "Pict 3" werden ebenfalls abhängig freigegeben.

Weitere abhängige oder rekursive Objekte werden nicht mehr berücksichtigt, da sie durch keine der Freigabeoptionen mehr abgedeckt werden.

Alle ausgehenden Referenzen für die abhängige Freigabe werden nur in einer Richtung vollständig berücksichtigt. Sollen alle abhängigen Objekte im Freigabeprozess enthalten sein, muss die Freigebe also in einer bestimmten Reihenfolge erfolgen (siehe Kapitel 6.2.8 Seite 236).

6.2.4 Erreichbarkeit sicherstellen (Vaterkette)

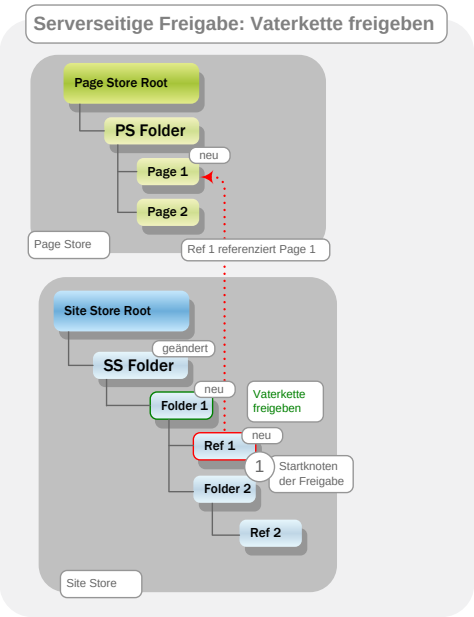


Abbildung 6-4: Serverseitige Freigabe – Vaterkette freigeben

Für den Aufruf von `AccessUtil.release(...)` wurden die folgenden Parameter gesetzt:

```
releaseStartNode:  Ref 1
releaseParentPath: true
boolean recursive:  false
DependentReleaseType: NO_DEPENDENT_RELEASE
```

Der ausgewählte Startknoten für die Freigabe ist die Seitenreferenz "Ref 1".

Erreichbarkeit sicherstellen (Vaterkette): Ausgehend vom Startknoten der Freigabe "Ref 1" wird die vollständige Vaterkette des Objekts bis zum Wurzelknoten der Verwaltung betrachtet. Durch die Option `releaseParentPath` werden alle Knoten der Vaterkette freigegeben, die *noch niemals freigegeben* wurden. Konkret bedeutet dies, Objekte die bereits einmal freigegeben wurden (geänderte Objekte), werden durch die Option `releaseParentPath` nicht freigeben und zwar auch dann nicht, wenn sie sich durch das Hinzufügen, beispielsweise einer Seitenreferenz, geändert haben (siehe Abbildung 6-4).

Wird die Option `releaseParentPath=false` gesetzt, wird die Vaterkette nicht freigegeben, die freizugebenden Elemente werden aber der Release-Kindliste des Vaterknotens hinzugefügt. Dabei gilt:

- Bei *geänderten* Vaterknoten: Das freizugebende Objekt ist im Freigabe-Stand erreichbar. Das Vaterelement wird aber nicht freigegeben.

- Bei *neuen* Vaterknoten: Da der Vaterknoten niemals freigegeben wurde, ist auch das freizugebende Objekt im Freigabe-Stand nicht erreichbar. Das kann zu ungültigen Referenzen im Freigabe-Stand führen.

Hintergrund: Soll eine Seitenreferenz freigegeben werden, obwohl der Redakteur kein Recht zur Freigabe innerhalb der übergeordneten Menüebene besitzt, sollte die Seitenreferenz trotzdem in den Freigabestand übernommen werden. Eventuelle inhaltliche Änderungen innerhalb der Menüebene (z. B. weitere Referenzen), sollen durch die Option `releaseParentPath` aber nicht freigegeben werden (siehe Abbildung 6-5).

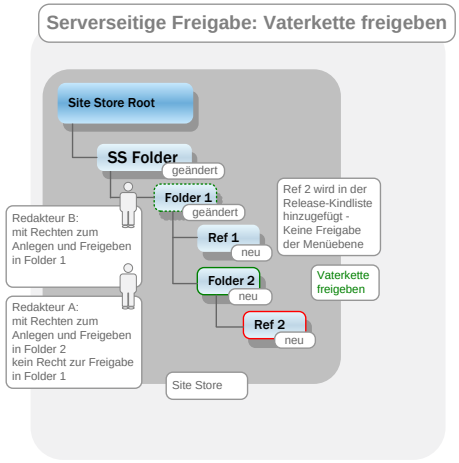


Abbildung 6-5: Serverseitige Freigabe – Vaterkette freigeben (Release-Kindliste)

Im Beispiel aus Abbildung 6-5 würde durch eine Freigabe der Seitenreferenz "Ref 2" von "Redakteur A" sowohl die neu angelegte Seitenreferenz als auch die neu angelegte Menüebene "Folder 2" freigegeben. Die Menüebene "Folder 1" wird *nicht* freigegeben. Damit die neue Menüebene "Folder 2" (und damit auch die Seitenreferenz "Ref 2") im Freigabestand erreichbar ist, wird "Folder 2" durch die Option `releaseParentPath` aber zur Release-Kindliste der Menüebene "Folder 1" hinzugefügt. Damit ist die Seitenreferenz "Ref 2" innerhalb des Freigabestands erreichbar, nicht aber die neu angelegte Seitenreferenz "Ref 1". Gibt nun "Redakteur B" die Seitenreferenz "Ref 1" frei, wird auch diese zur Release-Kindliste der Menüebenen "Folder 1" hinzugefügt. Da "Folder 1" als geändertes Objekt bereits über die Release-Kindliste des ebenfalls geänderten Folders "SS Folder" erreichbar ist, ist die Freigabe hiermit abgeschlossen. Die beiden Menüebenen ("Folder 1" und "SS Folder") werden über die Option nicht freigegeben, da es sich hier nicht um neu angelegte Objekte handelt.

6.2.5 Erreichbarkeit sicherstellen (Vaterkette) und rekursiv freigeben

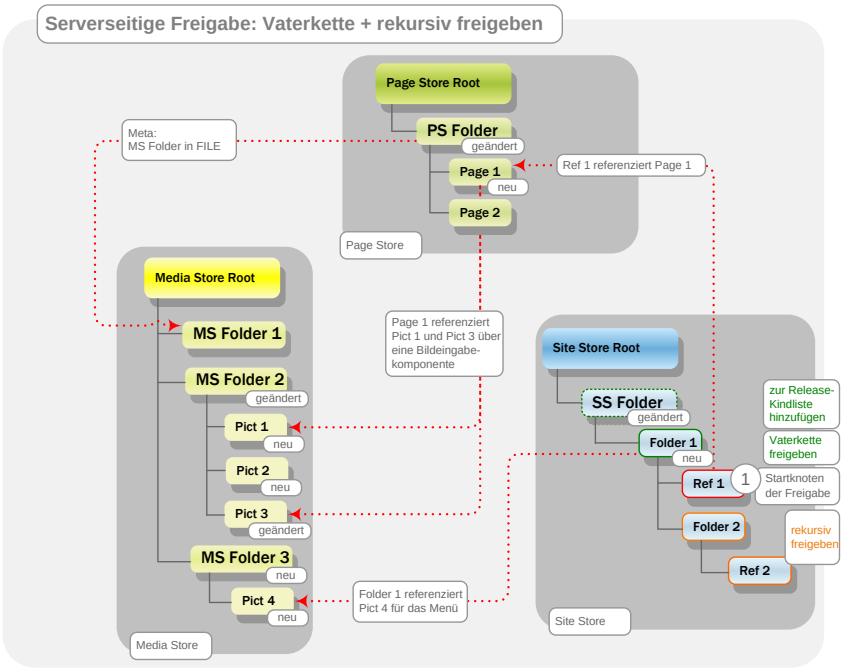


Abbildung 6-6: Serverseitige Freigabe – Vaterkette und rekursiv freigeben

Für den Aufruf von `AccessUtil.release(...)` wurden die folgenden Parameter gesetzt:

```
releaseStartNode:  Ref 1
releaseParentPath: true
boolean recursive:  true
DependentReleaseType: NO_DEPENDENT_RELEASE
```

Der ausgewählte Startknoten für die Freigabe ist die Seitenreferenz "Ref 1".

Erreichbarkeit sicherstellen (Vaterkette) und rekursive Freigabe: Ausgehend vom Startknoten der Freigabe "Ref 1" wird die vollständige Vaterkette des Objekts bis zum Wurzelknoten der Verwaltung betrachtet. Durch die Option `releaseParentPath` werden alle Knoten der Vaterkette freigegeben, die *noch niemals freigegeben* wurden (vgl. Kapitel 6.2.4 Seite 230). Zusätzlich werden alle Kindelemente des Startpunkts rekursiv freigegeben (vgl. Kapitel 6.2.1 Seite 226). Anhand des Beispiels in Abbildung 6-6 ist deutlich zu erkennen, dass diese Freigabe auf die Struktur-Verwaltung beschränkt ist, da hier keine Abhängigkeiten berücksichtigt werden (im Unterschied zu Abbildung 6-7). Bei dieser Freigabe ist zu beachten, dass defekte Referenzen entstehen, wenn ein in der Struktur-Verwaltung referenziertes Objekt neu angelegt wurde, im Beispiel also "Page 1" und die Medien "Pict 1" und "Pict 4". Die aktuelle Konfiguration im Beispiel (vgl. Abbildung 6-6) wird damit zu einem

Fehler innerhalb der Freigabe führen, da die referenzierte Seite "Page 1" niemals freigegeben wurde. Verweisen die Referenzen stattdessen auf Objekte, die bereits einmal freigegeben wurden ("geändert"), werden jeweils die zuletzt freigegebenen Versionen der Objekte referenziert. In diesem Fall könnte die Freigabe aus dem Beispiel erfolgreich ausgeführt werden.

6.2.6 Erreichbarkeit sicherstellen (Vaterkette) und abhängig freigeben

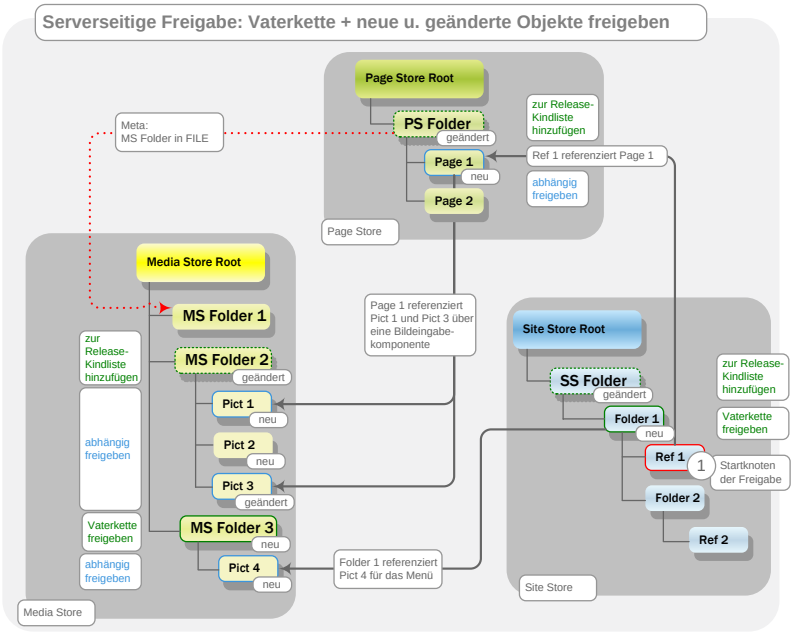


Abbildung 6-7: Serverseitige Freigabe – Vaterkette und abhängige Objekte freigeben

Für den Aufruf von `AccessUtil.release(...)` wurden die folgenden Parameter gesetzt:

```
releaseStartNode:  Ref 1
releaseParentPath: true
boolean recursive:  false
DependentReleaseType:
DEPENDENT_RELEASE_NEW_AND_CHANGED||DEPENDENT_RELEASE_NEW_ONLY
```

Der ausgewählte Startknoten für die Freigabe ist die Seitenreferenz "Ref 1".

Erreichbarkeit sicherstellen (Vaterkette) und abhängig freigeben: Werden die Optionen `DEPENDENT_RELEASE_NEW_ONLY` oder `DEPENDENT_RELEASE_NEW_AND_CHANGED` mit der Option `releaseParentPath` kombiniert, wirkt sich die abhängige Freigabe sowohl auf den aktuellen Startknoten als auch bei der Freigabe von noch niemals freigegebenen Elementen der Vaterkette aus. Das bedeutet beispielsweise für die Freigabe einer Seitenreferenz, dass die dort referenzierte Seite freigegeben wird.

Auch für die referenzierte Seite wird nun die gesamte Vaterkette durchlaufen und nach niemals freigegebenen Elementen durchsucht. Diese Elemente werden ebenfalls freigegeben. Gleiches gilt für abhängige Objekte in der Medien-Verwaltung.

- Für die Seitenreferenz "Ref 1" wird die gesamte Vaterkette durchlaufen. Dort werden alle niemals freigegebenen Objekte freigegeben, im Beispiel also die neue Menüebene "Folder 1", nicht aber die geänderte Menüebene "SS Folder".
- Die Menüebene "Folder 1" besitzt eine ausgehende Referenz in die Medien-Verwaltung. Durch die abhängige Freigabe wird zusätzlich das Medium "Pict 4" freigegeben.
- Für das Medium "Pict 4" wird nun wiederum die gesamte Vaterkette durchlaufen und alle niemals freigegebenen Objekte freigegeben. Im Beispiel wird nur der neue Medienordner "MS Folder 3" freigegeben.
- Bei der Freigabe der Seitenreferenz "Ref 1" wird die referenzierte Seite "Page 1" freigegeben.
- Für die Seite "Page 1" wird nun wiederum die gesamte Vaterkette durchlaufen und alle niemals freigegebenen Objekte freigegeben. Im Beispiel trifft das auf kein Objekt zu, da der Vaterknoten "PS Folder" bereits einmal freigegeben wurde. Abhängige Objekte des Ordners "PS Folder" werden daher bei der abhängigen Freigabe nicht berücksichtigt.
- Die Seite "Page 1" die abhängig freigegeben wurde, besitzt aber noch ausgehende Referenzen in die Medien-Verwaltung. Die referenzierten Medien "Pict 1" und "Pict 3" werden ebenfalls abhängig freigegeben.
- Für die beiden Medien wird nun ebenfalls die Vaterkette untersucht. Da sich der gemeinsame Vaterknoten "MS Folder 2" nur geändert hat, wird hier keine Freigabe ausgeführt.

Alle ausgehenden Referenzen für die abhängige Freigabe werden nur in einer Richtung vollständig berücksichtigt. Sollen alle abhängigen Objekte im Freigabeprozess enthalten sein, muss die Freigabe also in einer bestimmten Reihenfolge erfolgen (siehe Kapitel 6.2.8 Seite 236).

6.2.7 Erreichbarkeit sicherstellen (Vaterkette), rekursiv und abhängig freigeben

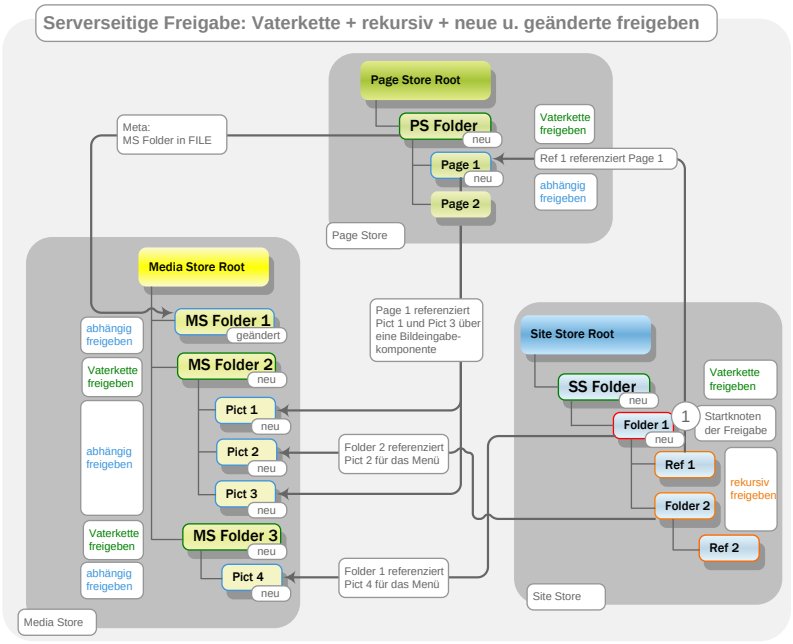


Abbildung 6-8: serverseitige Freigabe inklusive aller Optionen

Für den Aufruf von `AccessUtil.release(...)` wurden die folgenden Parameter gesetzt:

```
releaseStartNode:  Folder 1
releaseParentPath: true
boolean recursive:  true
DependentReleaseType:
DEPENDENT_RELEASE_NEW_AND_CHANGED||DEPENDENT_RELEASE_NEW_ONLY
```

Der ausgewählte Startknoten für die Freigabe ist die Menüebene "Folder 1".

Erreichbarkeit sicherstellen (Vaterkette), rekursiv freigeben und abhängig freigeben:
Die umfangreichste Freigabe wird ausgeführt, wenn alle Freigabeoptionen miteinander kombiniert werden. In diesem Fall werden sowohl alle niemals freigegebene Elemente der Vaterkette als auch alle Elemente unterhalb des Startknotens freigegeben. Zusätzlich werden die abhängigen Objekte aller vom Freigabeprozess betroffenen Knoten freigegeben und auch dort die gesamte Vaterkette untersucht und ggf. freigegeben. Die rekursive Freigabe wirkt sich, anders als die Freigabe der Vaterkette, nicht auf die abhängigen Objekte aus. Anhand des Beispiels in Abbildung 6-8 wird klar, dass die Freigabe sich annähernd auf alle dargestellten Objekte auswirkt – lediglich "Page 2" ist nicht betroffen:

- Am Startpunkt der Freigabe "Folder 1" wird die Option `recursive` ausgewertet. Die rekursive Freigabe wirkt sich *ausschließlich* auf die Kindelemente des

Startpunkts der Freigabe aus. Im Beispiel aus Abbildung 6-8 werden durch die Option also die Kindelemente Ref 1, Folder 2 und Ref 2 freigegeben.

- Es werden alle ausgehenden Referenzen der freigegebenen Objekte freigeben. Im Beispiel sind das die Objekte "Pict 4" (über die Referenz innerhalb der Menüebene "Folder 1"), "Page 1" (über die Seitenreferenz "Ref 1"), "Pict 2" (über die Referenz innerhalb der Menüebene "Folder 2")
- Von diese freigegeben Objekte werden wieder die ausgehenden Kanten untersucht und freigegeben. Im Beispiel sind das die Medien "Pict 1" und "Pict 3" (über die Referenz innerhalb der Seite "Page 1").
- Von allen freigegebenen Elementen werden nun die vollständigen Vaterketten untersucht und alle niemals freigegebenen Vaterknoten freigegeben. Im Beispiel sind das "SS Folder" (Vaterelement Startknoten), "PS Folder" (Vaterelement "Page 1"), "MS Folder 2" (Vaterelement "Pict 1" und "Pict 2"), "MS Folder 3" (Vaterelement "Pict 4").
- Nun werden die abhängigen Objekte der freigegeben Vaterknoten freigegeben. Im Beispiel ist das "MS Folder 1" (über die Referenz in "PS Folder"). Anders als bei der Freigabeoption `releaseParentPath` wird "MS Folder 1" auch dann freigegeben, wenn er nur "geändert" wurde, also bereits einmal freigegeben war.

Alle ausgehenden Referenzen für die abhängige Freigabe werden nur in einer Richtung vollständig berücksichtigt. Sollen alle abhängigen Objekte im Freigabeprozess enthalten sein, muss die Freigabe also in einer bestimmten Reihenfolge erfolgen (siehe Kapitel 6.2.8 Seite 236).

6.2.8 Reihenfolge für die Freigabe

Alle ausgehenden Referenzen für die abhängige Freigabe werden nur in einer Richtung vollständig berücksichtigt, um zyklische Abhängigkeiten bei der Freigabe zu unterbinden.

Objekte aus der Inhalte- bzw. Medien-Verwaltung, die in der Inhalte-, Struktur- oder Datenquellen-Verwaltung referenziert sind, werden bei der abhängigen Freigabe berücksichtigt.

Die umgekehrte Richtung (Struktur- / Medien-Verwaltung → Inhalte- / Struktur- / Datenquellen-Verwaltung) funktioniert nicht.

Objekte aus Remote-Projekte werden bei der abhängigen Freigabe nicht berücksichtigt.

Sollen alle abhängigen Objekte im Freigabeprozess enthalten sein, muss die folgende Reihenfolge eingehalten werden:

- Freigabe in der Struktur-Verwaltung beinhaltet ausgehende Referenzen in die Inhalte- und in die Medien-Verwaltung
- Freigabe in der Inhalte-Verwaltung beinhaltet ausgehende Referenzen in die Medien-Verwaltung

Nicht berücksichtigt werden:

- Freigabe in der Inhalte-Verwaltung beinhaltet *keine* ausgehenden Referenzen in die Struktur-Verwaltung
- Freigabe in der Medien-Verwaltung beinhaltet *keine* ausgehenden Referenzen in die Struktur-Verwaltung oder in die Inhalte-Verwaltung

Weitere Fälle, in denen abhängige Objekte zwar im Referenzgraphen angezeigt werden, aber bei der abhängigen Freigabe nicht mit freigegeben werden.

- Seite→Seitenreferenz: Seite mit einer FS_REFERENCE-Komponente, in der eine Seitenreferenz referenziert wird.
  ⇨ Nur die Seite wird freigegeben, die abhängige Seitenreferenz nicht.
- Seite→Medium: Seite mit der Seitenvorlage, in der eine Medium-Referenz hartcodiert ist. Beispielsweise: $CMS_REF(media:"XXX")$ im HTML-Kanal.
  ⇨ Nur die Seite wird freigegeben, das abhängige Medium aber nicht.
- Medium→Medium: In einer CSS-Datei (Datei parsen: ja) wird ein weiteres Medium (z.B. ein Bild) hartcodiert referenziert. Beide Medien sind noch nicht freigegeben.
  ⇨ Wird die CSS-Datei freigegeben ("Spezifische Freigabe -> Abhängige Objekte freigeben"), wird das referenzierte Medium nicht mit freigegeben.
- Seite mit LINK/DOM-Editor→Seitenreferenz: Beide referenzierten Objekte sind noch nicht freigegeben.
  ⇨ Das referenzierte Medium (Bild) wurde auch mit freigegeben, aber die referenzierte Seitenreferenz nicht.
- Seite→Datensatz: Seite mit der CONTENTLIST/FS_LIST/... -Komponente, in der Datensätze referenziert werden.
  ⇨ CS-Objekt wird nicht mit freigegeben.

Unter bestimmten Umständen kann es zu zyklischen Abhängigkeiten kommen, die nicht automatisiert freigegeben werden können und daher manuell aufgelöst werden müssen.
Beispiel: *Es exisitieren 2 Seiten in der Inhalte-Verwaltung ("Seite 1" und "Seite 2") mit jeweils einem Absatz und einer Absatzreferenz auf den Absatz der anderen Seite:*
-- Seite 1
-- Absatz A
-- Absatzreferenz auf Absatz B der Seite 2
-- Seite 2
-- Absatz B
-- Absatzreferenz auf Absatz A der Seite 1

Wurden die Absatzreferenzen noch nicht freigegeben, kann weder Seite 1 noch Seite 2 in dieser Konstellation automatisch freigegeben werden. Um die Seiten freizugeben, muss zunächst eine der Absatzreferenzen gelöscht werden, um die zyklische Abhängigkeit aufzulösen, z. B. "Absatzreferenz auf Absatz B der Seite 2". Dann kann Seite 2 freigegeben werden. Anschließend muss die Absatzreferenz wieder hergestellt werden, dann kann Seite 1 ebenfalls freigegeben werden.

Es gibt einige Fälle, in denen die abhängigen Objekte zwar im Referenzgraphen angezeigt werden, aber bei der abhängigen Freigabe nicht mit freigegeben werden.

7 Code-Vervollständigung für Formulare

Um Vorlagenentwickler stärker bei der Erstellung von Formularen zu unterstützen, wird mit FirstSpirit Version 5.0 eine Code-Vervollständigung auf dem Formular-Register eingeführt. Über diese Code-Vervollständigung können alle verfügbaren FirstSpirit-Eingabekomponenten sowie alle zugehörigen Parameter mit den zur Verfügung stehenden Werten per Tastendruck angezeigt und an der Einfügemarke in das Formular-Register eingefügt werden, z. B.

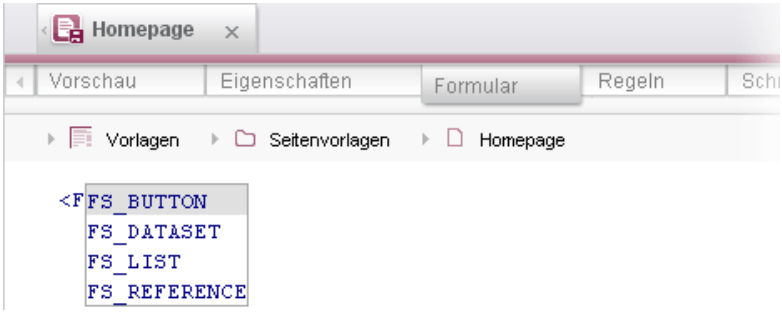


Abbildung 7-1: Auto-Vervollständigung auf dem Formular-Register

Dazu muss die Einfügemarke innerhalb eines `<CMS_MODULE>`-Tag positioniert sein.

Tags und Parameter zu den Eingabekomponenten, Daten- und Gestaltungselementen mit den jeweiligen Werten sowie deren Syntax und Bedeutung können in der FirstSpirit Online Dokumentation, Kapitel "Vorlagenentwicklung" / "Formulare", nachgeschlagen werden.

7.1.1 Einfügen der Eingabekomponenten-Tags

Um die Eingabekomponenten-Tags (`FS_...` bzw. `CMS_...`) zu ermitteln, muss eine spitze Klammer geöffnet (`<`) und die Einfügemarke dahinter positioniert werden. Die Tags werden dann in einer Liste angezeigt, wenn <Strg> und Leertaste gleichzeitig gedrückt werden. Das gewünschte Tag kann dann per Tastatur (Cursor-Taste nach oben bzw. nach unten und <Enter>) oder Maus (doppelter Mausklick oder Mausklick und <Enter>) auf das Formular-Register übernommen werden. Dabei werden das öffnende und schließende Tag sowie Pflichtparameter (in der Regel `name`) eingefügt, z. B. bei Auswahl von `FS_BUTTON`:

```
<FS_BUTTON name=""></FS_BUTTON>
```

Die Einfügemarke befindet sich dann zwischen den Anführungszeichen des

Parameters `name`.

Die Anzahl der eingeblendeten Tags kann durch die Eingabe des oder der ersten Buchstaben der gewünschten Eingabekomponente hinter der spitzen Klammer eingeschränkt werden, z. B. `<C` für die mit "CMS_" beginnenden oder `<F` für die mit "FS_" beginnenden Eingabekomponenten.

Durchgestrichene Einträge in der Liste sind veraltet und sollten nicht verwendet werden.

7.1.2 Einfügen von Tags, Parametern und Schlüsselbegriffen

Um die zur Verfügung stehenden Tags, Parameter und Schlüsselbegriffe einer Eingabekomponente anzuzeigen und auswählen zu können, muss die Einfügemarke je nach Formular-Syntax folgendermaßen positioniert sein:

- **in öffnenden Tags:** Um Parameter innerhalb eines öffnenden Tags anzuzeigen, muss vor der Einfügemarke ein Leerzeichen vorhanden sein.
- **zwischen öffnendem und schließendem Tag:** Um Tags zwischen öffnendem und schließendem Tag anzuzeigen, muss vor der Einfügemarke eine spitze Klammer geöffnet werden (`<`).
- **innerhalb von Anführungszeichen:** Um von FirstSpirit vorgegebene Werte ("Schlüsselbegriffe") zu einem Parameter anzuzeigen, muss die Einfügemarke innerhalb der Anführungszeichen positioniert werden.

Es werden immer nur die Tags, Parameter und Schlüsselbegriffe angezeigt, die zum gewählten Tag oder Parameter verfügbar sind. Bereits für das Formular verwendete Tags oder Parameter, die nur einmal verwendet werden können, werden nicht mehr in der Liste angezeigt.

Sind die gewünschten Tags, Parameter und Schlüsselbegriffe bereits bekannt, können auch der oder die ersten Buchstaben eingegeben werden. Mit <Strg> + Leertaste wird dann die Anzahl der auszuwählenden Einträge reduziert bzw. der Eintrag direkt eingefügt. Soweit möglich, werden auch Pflichtparameter direkt mit eingefügt.

Durchgestrichene Einträge in der Liste sind veraltet und sollten nicht verwendet werden.