FirstSpirit™
Unlock Your Content

FirstSpirit™ FormEdit

FirstSpirit Version 5.0

Version	**1.09**
Status	**RELEASED**
Datum	**2013-06-18**
Abteilung	Product Management
Copyright	2012 e-Spirit AG
Dateiname	FORM50DE_FirstSpirit_Modules_FormEdit

e-Spirit AG

Barcelonaweg 14
44269 Dortmund | Germany

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

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

e-Spirit

Inhaltsverzeichnis

1 Einleitung

Das Modul „FirstSpirit FormEdit“ besteht aus einer redaktionellen Komponente zur Erstellung von Web-Formularen über den FirstSpirit Java- oder WebClient und einer Web-Komponente in Form eines Servlets, das die vom Nutzer eingegebenen Daten entgegennimmt und weiterverarbeitet.

1.1 Übersicht der Funktionen

Folgende Formen der Weiterverarbeitung von Formulardaten werden unterstützt:

- Speichern der Daten in einer Datei im CSV-Format:
  Diese Art der Weiterverarbeitung speichert alle vom Formular gesendeten Werte innerhalb einer freidefinierbaren Datei im CSV-Format ab.
- Speichern der Daten in einer JDBC-fähigen Datenbank:
  Durch Verwendung dieses Typs der Weiterverarbeitung ist es möglich, die Werte innerhalb einer Datenbank zu speichern. Über die Konfigurationseinstellung kann ein individuelles Mapping für die Formularfelder definiert werden.
- Versand der Daten als E-Mail:
  Bei dieser Art der Weiterverarbeitung können beliebige Formulardaten per E-Mail versendet werden. Mittels einer E-Mail-Vorlage kann der Aufbau der E-Mail individuell gestaltet werden. Die E-Mails können u.a. mit Dateianhängen versendet werden, ebenso sind cc- und bcc-Empfänger möglich.
- Ausgabe der Daten in das Log-File der Servlet-Engine:
  Diese Funktion dient dazu, alle Werte des Formulars innerhalb des Log-Files der Servlet-Engine auszugeben, in der das FormServlet initialisiert ist.
- Aufruf einer URL mit Parameterübergabe

  Diese Funktion ermöglicht es, eine URL mit den definierten Parametern aufzurufen, ohne dass der Benutzer diese Seite im Browser sieht. (z.B. Tracking)

Weiterhin ist es möglich, die Formulare über eigene Implementierungen auszuwerten. Die oben genannten Weiterverarbeitungsmethoden können darüber hinaus miteinander kombiniert werden.

1.2 Thema dieser Dokumentation

Kapitel 1: Bietet eine kurze Einführung in Aufbau und Funktionsumfang des Moduls „FirstSpirit FormEdit“ (ab Seite 4).

Kapitel 2: Beschreibt die Installation des Moduls „FirstSpirit FormEdit“ auf dem Server und die Installation der Web-Komponente in einem Projekt (ab Seite 7).

Kapitel 3: In diesem Kapitel wird die Konfiguration des Moduls „FirstSpirit FormEdit“ erläutert (ab Seite 15).

Kapitel 4: Beschreibt die Formularerstellung für das Modul „FirstSpirit FormEdit“ und listet alle verfügbaren Formular-Elemente auf (ab Seite 31).

Kapitel 5: Dieses Kapitel beschreibt die Verwendung der beispielhaften Stylesheet-Datei „formedit_css“, mit der Formular-Komponenten schnell und einfach angepasst werden können. Darüber hinaus werden die Nutzung und Funktionen des Frameworks „jQuery“ und dessen Plugins erläutert, mit deren Hilfe z.B. die verwendeten Formular-Komponenten bereits bei der Eingabe auf inhaltliche Korrektheit überprüft werden können (ab Seite 66).

Kapitel 6: In diesem Kapitel wird das mitgelieferte Beispiel für die Funktion der Auto-Vervollständigung erklärt. Das Kapitel dient als Konzept für die Entwicklung eigener Lösungen zum Finden von sinnvoll vervollständigten Begriffen.

Kapitel 7: Dieses Kapitel beschreibt anhand eines Beispiels die Aktionen, die ein Redakteur durchführen muss, um ein Formular für ein Gewinnspiel zu erstellen, welches sowohl Daten per E-Mail versendet als auch die eingegebenen Daten in einer Datenbank speichert (ab Seite 71).

1.3 Aufbau und Funktion

Die folgende Grafik zeigt den Aufbau und die Funktionsweise des Moduls am Beispiel des Live-Servers. Bei der Nutzung innerhalb der Preview oder des Stagings werden die in FirstSpirit verwendeten Web- und Applikationsserver benutzt.

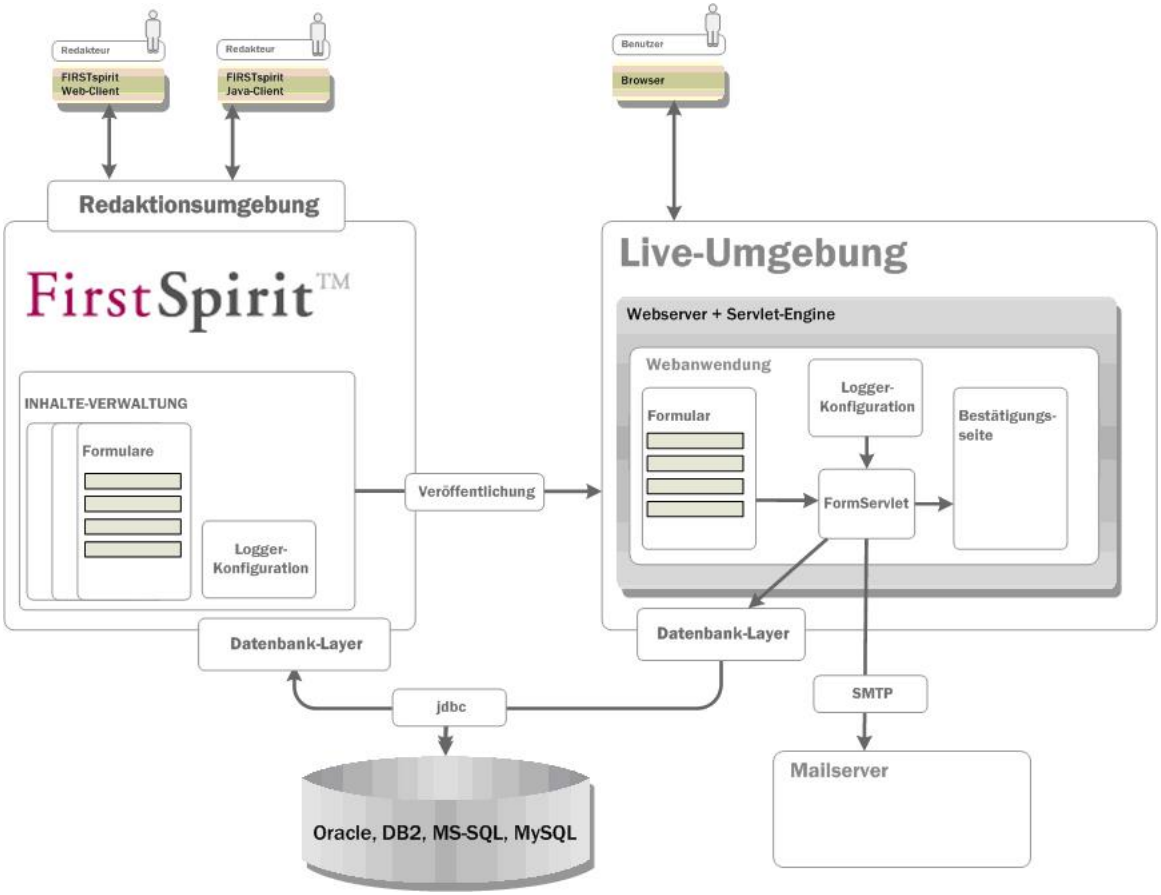

Abbildung 1-1: Aufbau und Funktionsweise

2 Installation und Konfiguration

Die Installation von „FirstSpirit FormEdit“ erfolgt in fünf Schritten:

2.1 Installieren des Moduls auf dem Server

Das Modul „FirstSpirit FormEdit“ muss zunächst innerhalb der Anwendung zur Server- und Projektkonfiguration installiert werden. Im Bereich „Server-Eigenschaften“ wird dazu der Menüeintrag „Module“ selektiert. Mit einem Klick auf den Button „Installieren“ öffnet sich ein Dateiauswahldialog. Hier kann die zu installierende fsm-Datei ausgewählt werden. Das erfolgreich installierte Modul wird anschließend im Dialog „Server-Eigenschaften“ angezeigt:

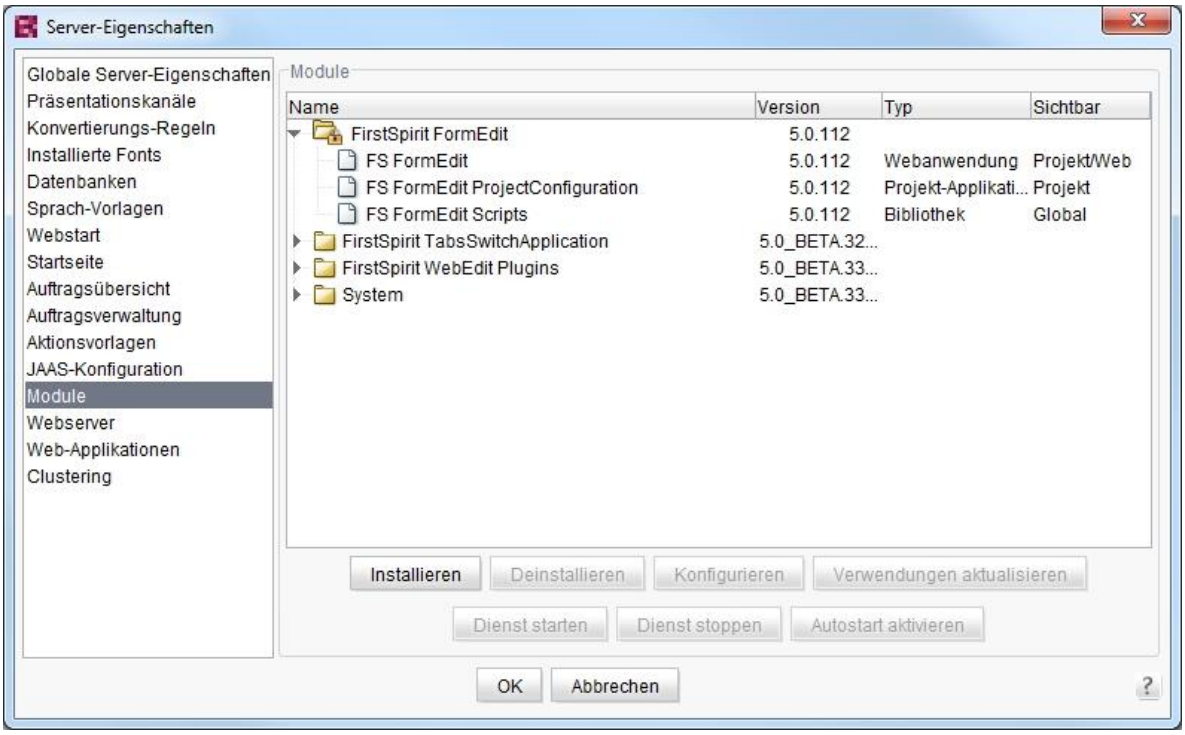

Abbildung 2-1: Installation des Moduls auf dem FirstSpirit-Server

Bestandteil des Moduls „FirstSpirit FormEdit“ ist die Projekt-Applikation „FS FormEdit ProjectConfiguration“, die Webanwendung „FS FormEdit“ und die Bibliothek „FS

FormEdit Scripts“.

Die **Projekt-Applikation** stellt Medien, Seiten-, Absatz-, Skript- und Tabellenvorlagen zur Verfügung, die zum Gestalten von Formularen verwendet werden können. Die Komponente ist „Sichtbar“ für den Bereich „Projekt“. Damit handelt es sich um eine „projekt-lokale“ Komponente. Diese kann nach der Installation den Projekt-Komponenten innerhalb der gewünschten Projekte zugefügt werden (siehe Kapitel 2.2.1 Seite 9).

Die **Webanwendung** stellt Servlets zur Verfügung, die innerhalb des Projekts verwendet und aufgerufen werden können. Die Komponente ist „Sichtbar“ für die Bereiche „Projekt/Web“. Damit handelt es sich um eine „web-lokale“ Komponente. Diese kann nach der Installation den unterschiedlichen Web-Bereichen (preview, staging, live, webedit) innerhalb der gewünschten Projekte zugefügt werden (siehe Kapitel 2.3 Seite 10).

Die **Bibliothek** stellt die für die Skripte benötitgen Klassen zur Verfügung.

Nach der Installation des Moduls müssen diesem über den Button „Konfigurieren“ noch „Alle Rechte“ vergeben werden.

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

2.2 Installation der Projekt-Komponente

2.2.1 Hinzufügen der Projekt-Komponente

Die Projekt-Komponente muss nun im gewünschten Projekt installiert werden. Dazu wird innerhalb der Projekteigenschaften der Menüeintrag „Projekt-Komponenten“ aufgerufen.

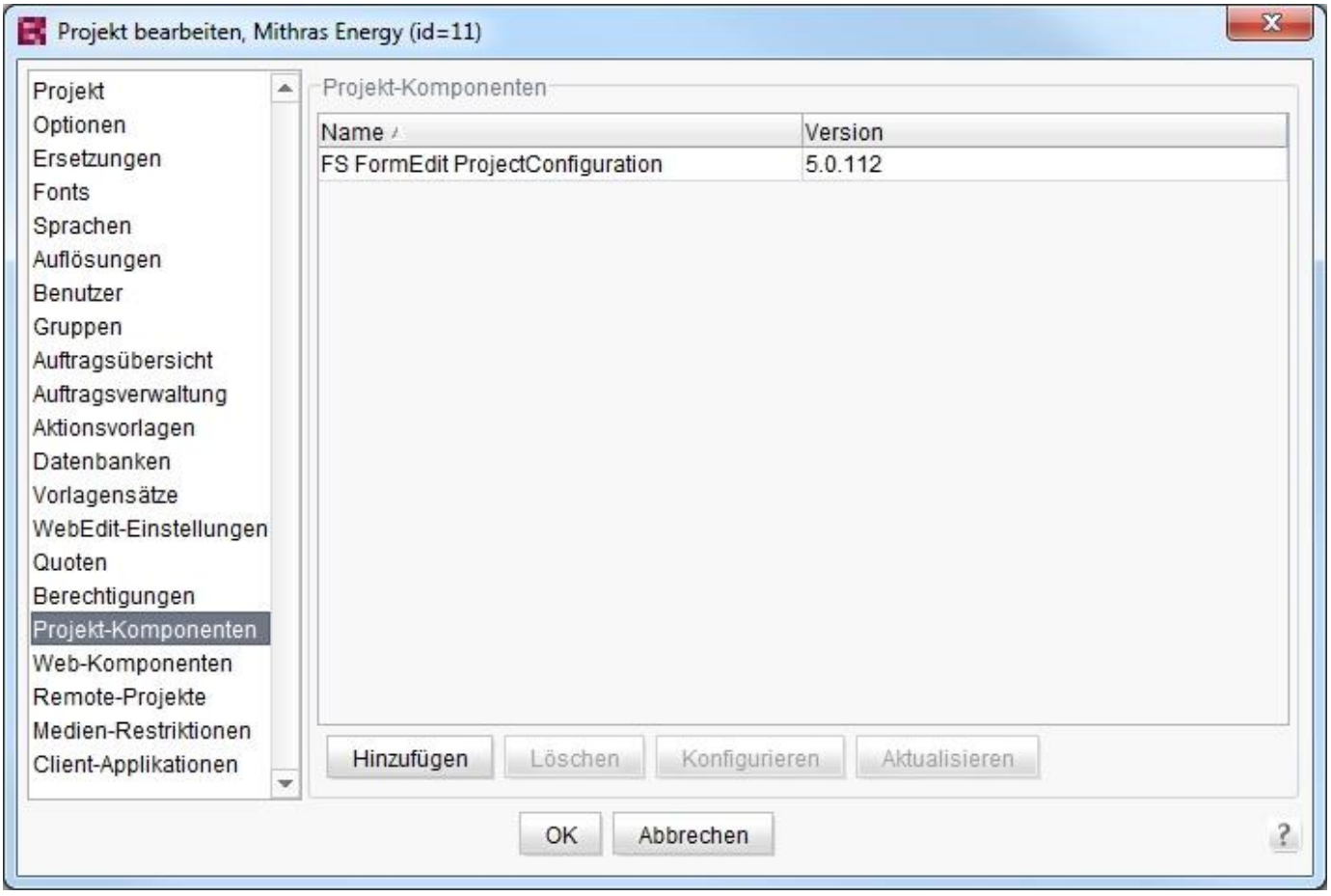

Abbildung 2-2: Installation der Projekt-Komponente

Hinzufügen: Mit einem Klick auf den Button öffnet sich der Dialog „Hinzufügen“. In der Liste werden alle Projekt-Komponenten angezeigt, die auf dem Server installiert sind (siehe Kapitel 2.1 Seite 4). Wählen Sie den Eintrag „FS FormEdit ProjectConfiguration“.

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

2.2.2 Hinzufügen der Vorlagen in das Projekt

Abbildung 2-3: Konfiguration der Projekt-Komponente

Konfigurieren: Selektieren Sie den gerade hinzugefügten Eintrag „FS FormEdit ProjectConfiguration“ und klicken Sie auf den Button „Konfigurieren“. Wählen Sie aus der Combobox „Schema“ einen Datenbank-Layer und klicken Sie auf „Templates importieren“.

In der Auswahlliste befinden sich alle für das Projekt zugelassenen Datenbank-Layer. Sollten Sie bisher keine Layer verwenden oder wünschen Sie für das Modul eine eigene Datenbank, wählen Sie „Neuer Layer“. Ist diese Option gewählt, wird ein neuer Layer erzeugt, der auf die FirstSpirit-interne Derby-Datenbank zeigt.

Weitere Informationen zu Datenbank-Layern finden Sie im *FirstSpirit Handbuch für Administratoren*.

Nachdem Sie den Button „Templates importieren“ angeklickt haben und den Dialog schließen, ist keine Änderung der Konfiguration mehr möglich. Ein erneutes Importieren ist nur mittels „Löschen“ und erneutem „Hinzufügen“ der Projekt-Komponente möglich.

2.3 Installieren der Webanwendung im Projekt

Die Webanwendung muss nun im gewünschten Projekt installiert werden. Dazu wird innerhalb der Projekteigenschaften der Menüeintrag „Web-Komponenten“ aufgerufen. In diesem Bereich können die Web-Komponenten für ein Projekt aktiviert werden.

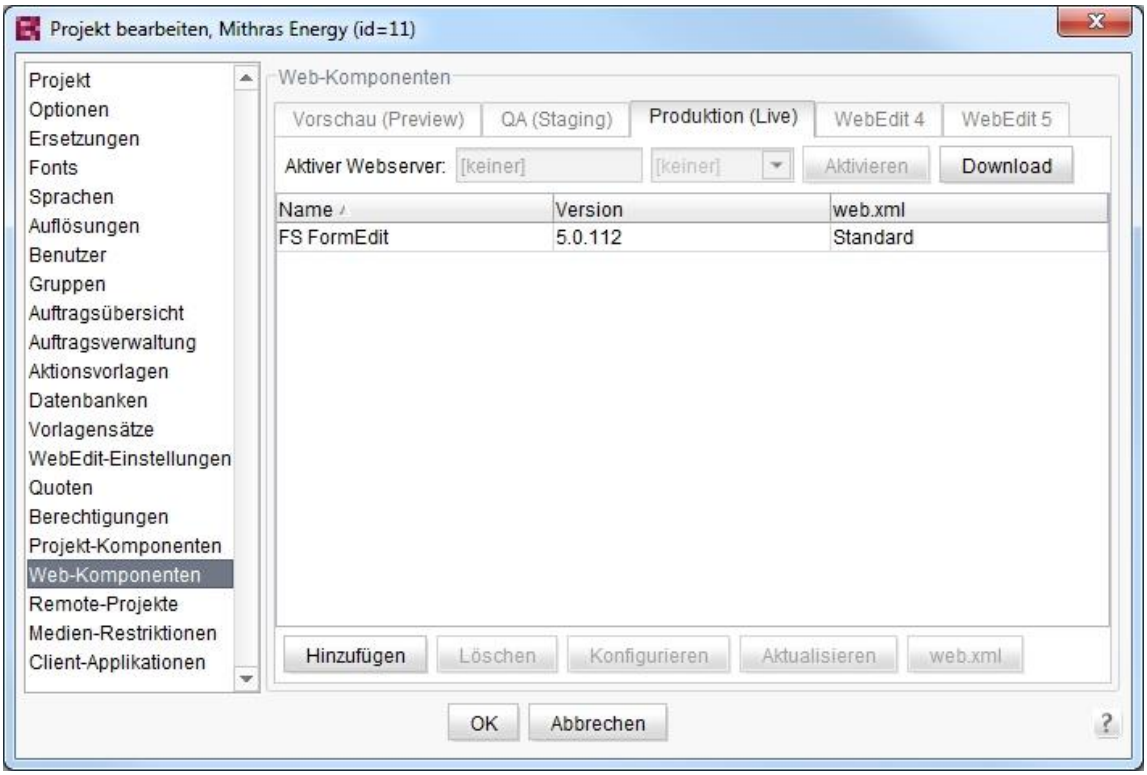

Abbildung 2-4: Installation der Webanwendung innerhalb der Web-Bereiche

Es existieren für jedes Projekt vier unterschiedliche Web-Bereiche. Über die jeweilige Registerkarte können die Web-Komponenten für jeden Bereich einzeln aktiviert und konfiguriert werden:

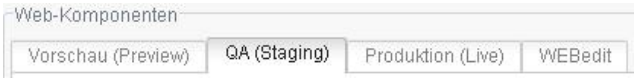

Abbildung 2-5: Web-Bereiche innerhalb eines Projekts

- Vorschau (Preview): Ort für die Projektinhalte, für die eine Vorschau angefordert wurde
- QA (Staging): Ort für die generierten Projektinhalte
- Produktion (Live): Ort für die veröffentlichten Projektinhalte
- WEBedit: Ort für die Projektinhalte bei Nutzung einer projektlokalen WEBedit-Umgebung

Hinzufügen: Mit einem Klick auf den Button öffnet sich der Dialog „Hinzufügen“. In der Liste werden alle Web-Komponenten angezeigt, die auf dem Server installiert sind (siehe Kapitel 2.1 Seite 4).

Nach dem Hinzufügen zu einem Webbereich besteht die Möglichkeit, die Komponenten zu konfigurieren, entweder mit einer von der Komponente erzeugten web.xml oder einer generischen GUI (siehe Kapitel 2.3.1 Seite 12). Nach der Konfiguration müssen die Komponenten noch aktiviert werden. Dabei kann eine

Komponente innerhalb eines Projekts nur für bestimmte Bereiche aktiviert bzw. deaktiviert werden.

Bei der Konfiguration für Produktion (Live) ist zu beachten, dass kein automatisches Deployment der Webanwendungen und deren Konfigurationsdateien stattfindet, sondern das über die Schaltfläche „Download“ erzeugte .war-File muss manuell auf den Live-Server übertragen werden.

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

2.3.1 Konfigurieren der Webanwendung

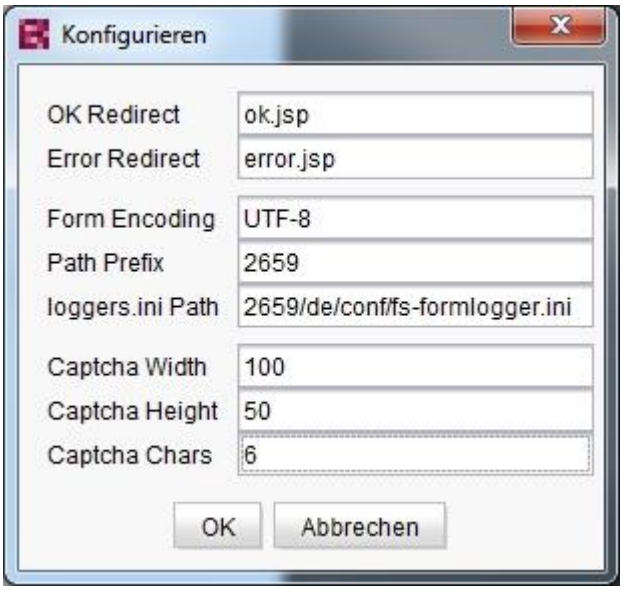

Abbildung 2-6: Konfiguration der Webanwendung

Zur Konfiguration von „FirstSpirit FormEdit“ stehen unterschiedliche Parameter zur Verfügung.

OK Redirect: In diesem Feld wird der Pfad zu der Datei angegeben, die nach erfolgreichem Versenden der Formulardaten angezeigt wird. Auf diesen Wert wird zurückgegriffen, falls im Formular keine spezielle Seite angegeben wurde (siehe Kapitel 4.2.1 Seite 32). Über das Prefix „redirect:“ bzw. „forward:“ kann das Weiterleitungsverhalten beeinflusst werden. „forward:ok.jsp“ würde z.B. eine Weiterleitung mit allen Parametern auf die Seite ok.jsp erzeugen. Ist weder „forward:“ noch „redirect:“ angegeben, erfolgt immer ein Redirect.

Error Redirect: In diesem Feld wird der Pfad zu der Datei angegeben, die nach fehlerhaftem Versenden der Formulardaten angezeigt wird. Auf diesen Wert wird

zurückgegriffen, falls im Formular keine spezielle Seite angegeben wurde (siehe Kapitel 4.2.1 Seite 32). Über das Prefix „redirect:“ bzw. „forward:“ kann das Weiterleitungsverhalten beeinflusst werden. „forward:error.jsp“ würde z.B. eine Weiterleitung mit allen Parametern auf die Seite error.jsp erzeugen. Ist weder „forward:“ noch „redirect:“ angegeben, erfolgt immer ein Redirect.

Form Encoding: In diesem Feld wird das Encoding angegeben, welches zum Versenden der Formulardaten verwendet werden soll. Hierbei handelt es sich ebenfalls um einen Rückgriffswert, wenn in der Konfiguration einer Weiterverarbeitungskomponente kein Encoding angegeben wurde (siehe Kapitel 3.2 Seite 18). Beispiele sind „UTF-8“ oder „ISO-8859-1“.

Achten Sie immer drauf, dass das gewählte Encoding mit dem verwendeten Encoding der generierten Seiten übereinstimmt (MetaTags bzw. Encoding der Sprache in den Server- und Projekteigenschaften).

Path Prefix: In diesem Feld ist ein Prefix anzugeben, welches dem Pfad zur Mail-Vorlage vorangestellt wird, um diese verwenden zu können. Dieses Prefix beschreibt den Teilpfad zwischen dem WebApp-Root und dem von FirstSpirit erstellten Ordner. Für die Staging-Umgebung wäre dies beispielsweise die Auftrags-ID.

Loggers.ini Path: In diesem Feld muss der Pfad zur Konfigurationsdatei `fs-formlogger.ini` angegeben werden. Ist dieses Feld leer oder kann die Datei nicht gefunden werden, wird auf eine **leere** Konfigurationsdatei zurückgegriffen.

Beispiel Staging:
`2708/de/conf/fs-formlogger.ini`

Dem Pfad für die Staging-Umgebung ist die Auftrags-ID – hier `2708` – vorangestellt.

Beispiel Live:
`de/conf/fs-formlogger.ini`

```
Der hier anzugebende Pfad kann auch absolut angegeben warden.
Dieses Beispiel sucht die Datei relativ zum WebAppRoot.
```

Captcha Width: In diesem Feld wird die Anzeigebreite der Captcha-Grafik in Pixel bestimmt. Ist dieses Feld leer, wird ein interner Rückgriffwert des Servlets benutzt: 100.

Captcha Height: In diesem Feld wird die Anzeigehöhe der Captcha-Grafik in Pixel bestimmt. Ist dieses Feld leer, wird ein interner Rückgriffwert des Servlets benutzt:

100.

Captcha Chars: In diesem Feld wird die Anzahl der Zeichen, die in der Captcha-Grafik angezeigt werden, angegeben. Ist dieses Feld leer, wird ein interner Rückgriffwert des Servlets benutzt: 6.

2.4 Liveserver-seitiges Logging

Das FormEdit-Modul kann zur einfacheren Fehleranalyse Logausgaben in eine entsprechende Logdatei ausgeben. Dazu wird auf das log4j-Framework zurückgegriffen, weshalb dieses für die Webanwendung zunächst entsprechend eingerichtet werden muss, falls dies noch nicht der Fall ist. Dies geschieht durch Ablegen der log4j-Konfigurationsdatei `log4j.properties` ins /WEB-INF/classes Verzeichnis. Falls dieses Verzeichnis noch nicht existiert, kann dies einfach manuell angelegt werden.

Eine beispielhafte log4j-Konfigurationsdatei könnte folgendermaßen aussehen:

```
log4j.rootLogger=debug, stdout, R

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

# Pattern to output the caller's file name and line number.
log4j.appender.stdout.layout.ConversionPattern=%5p [%t]
(%F:%L) - %m%n

log4j.appender.R=org.apache.log4j.RollingFileAppender
log4j.appender.R.File=example.log

log4j.appender.R.MaxFileSize=100KB
# Keep one backup file
log4j.appender.R.MaxBackupIndex=1

log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%p %t %c - %m%n
```

Es ist zu beachten, dass abhängig von der Konfiguration auch Logausgaben aus anderen Modulen in der Logdatei auftauchen können.

3 Konfiguration

Für diesen Abschnitt werden Kenntnisse im Umgang mit FirstSpirit im Bereich „Datenquellen“ vorausgesetzt.

Informationen zum Umgang mit Datenquellen können Sie dem *FirstSpirit Handbuch für Entwickler* sowie dem *FirstSpirit Handbuch für Redakteure (JavaClient) entnehmen*.

Die verschiedenen Weiterverarbeitungsmöglichkeiten der Formulardaten werden durch sogenannte Logger im Projekt konfiguriert. Jedem Logger wird ein bestimmter Weiterverarbeitungstyp (z.B. MailLogger) zugeordnet und mit entsprechenden Parametern versehen. Die verschiedenen Logger werden als Datensätze in einer Datenquelle gepflegt. Anhand der Logger-Konfiguration wird bei der Generierung die Logger-Konfigurationsdatei „fs-formlogger.ini“ erzeugt. Das dazu notwendige Datenbankschema und die nötigen Tabellenvorlagen werden bei Installation der Projektkomponente erzeugt. Um Logger anlegen zu können, muss nur noch eine Datenquelle für die Tabellenvorlage „form_edit.formLogger“ erstellt werden. Informationen zu den Logger-Typen und deren Konfigurationsmöglichkeiten finden Sie im Kapitel 3.2 ab Seite 18.

Bitte beachten Sie, dass Sie das Mapping innerhalb der Tabellenvorlage „form_edit.formLogger“ noch für die fehlenden Sprachen setzen. (Im Auslieferungszustand ist nur die Sprache „deutsch“ gemapped). Für die sprachabhängige Spalte „formLogger_description“ sollten weitere Spalten mit „_<Sprachkürzel>“ angelegt werden z. B. „formLogger_description_DE“.

3.1 Anlegen der Logger

Öffnen Sie das Projekt im JavaClient. Nun gibt es zum Anlegen oder Bearbeiten der Logger zwei Möglichkeiten:

1. direkt über die Datenquellen (Datenquelle: FormLogger) in der Datenquellen-Verwaltung

Abbildung 3-1: Neuer Datensatz (Datenquellen)

2. aus dem Absatz „form-start“ (siehe Kapitel 4.2.1 Seite 32) heraus

Abbildung 3-2: Neuer Datensatz (form-start)

Nach einem Klick auf den Button „Neuer Eintrag“ in der Datenquellenansicht oder in der Eingabekomponente innerhalb des „formstart“-Absatzes wird folgendes Formular geöffnet:

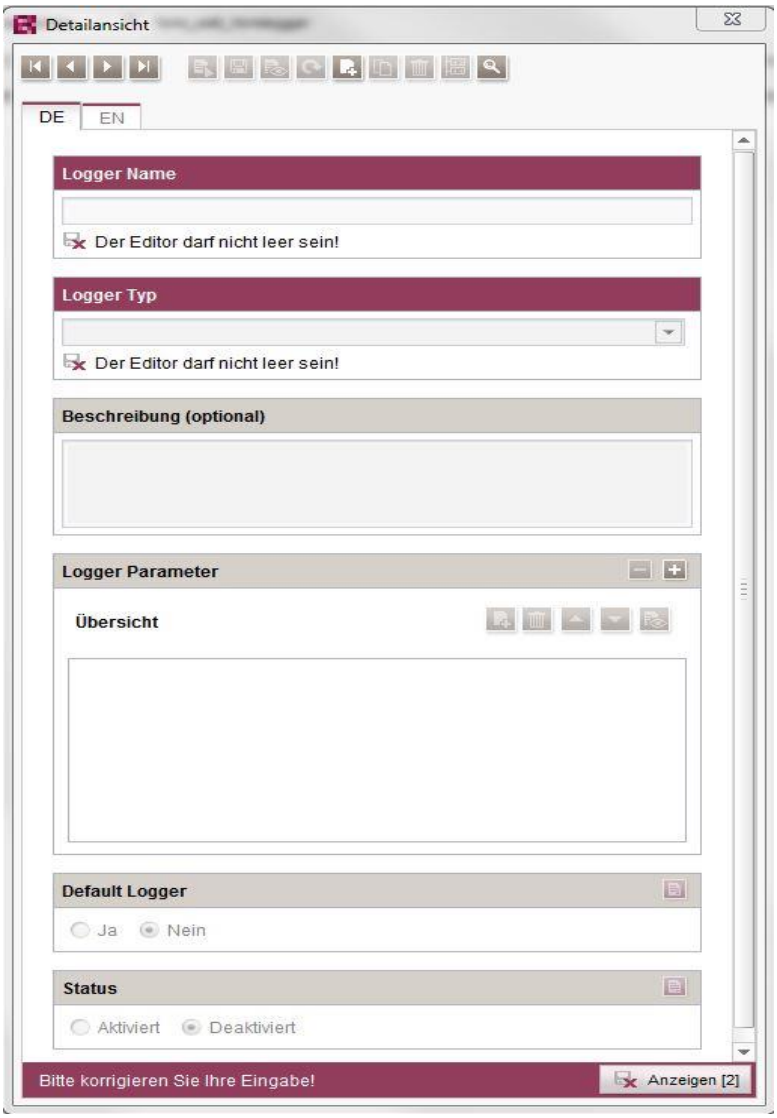

Abbildung 3-3: Anlegen der Logger

In diesem Formular werden alle logger-spezifischen Daten erfasst:

Logger Name: Hier kann der Logger mit einem Namen versehen werden. Der Name wird innerhalb der Konfigurationsdatei als Referenzname verwendet. Der Name darf

keine Leer- oder Sonderzeichen enthalten und muss projektweit eindeutig sen.

Logger Typ: Der Logger Typ wird für die Erstellung der Logger-Konfigurationsdatei „fs-formlogger.ini“ (siehe Kapitel 3.5 Seite 30) benötigt. Folgende Logger-Typen stehen zur Verfügung (Beschreibung siehe Kapitel 3.2 ff. ab Seite 18):

- ConsoleLogger (siehe Kapitel 3.2.1 Seite 18)
- CSVLogger (siehe Kapitel 3.2.2 Seite 19)
- jdbcLogger (siehe Kapitel 3.2.3 Seite 20)
- MailLogger (siehe Kapitel 3.2.4 Seite 22)
- MailUploadLogger
- UrlLogger (siehe Kapitel 3.2.5 Seite 24)

Beschreibung: In diesem Feld kann eine kurze Beschreibung eingegeben werden, um den Logger später einfacher zuordnen zu können. Die Eingabe ist optional und nimmt keinen Einfluss auf die Funktion des Loggers.
Logger Parameter: Hier können die logger-spezifischen Konfigurationsparameter festgelegt werden. Bei einem Eintrag ist zwischen drei Vorlagen zu wählen:

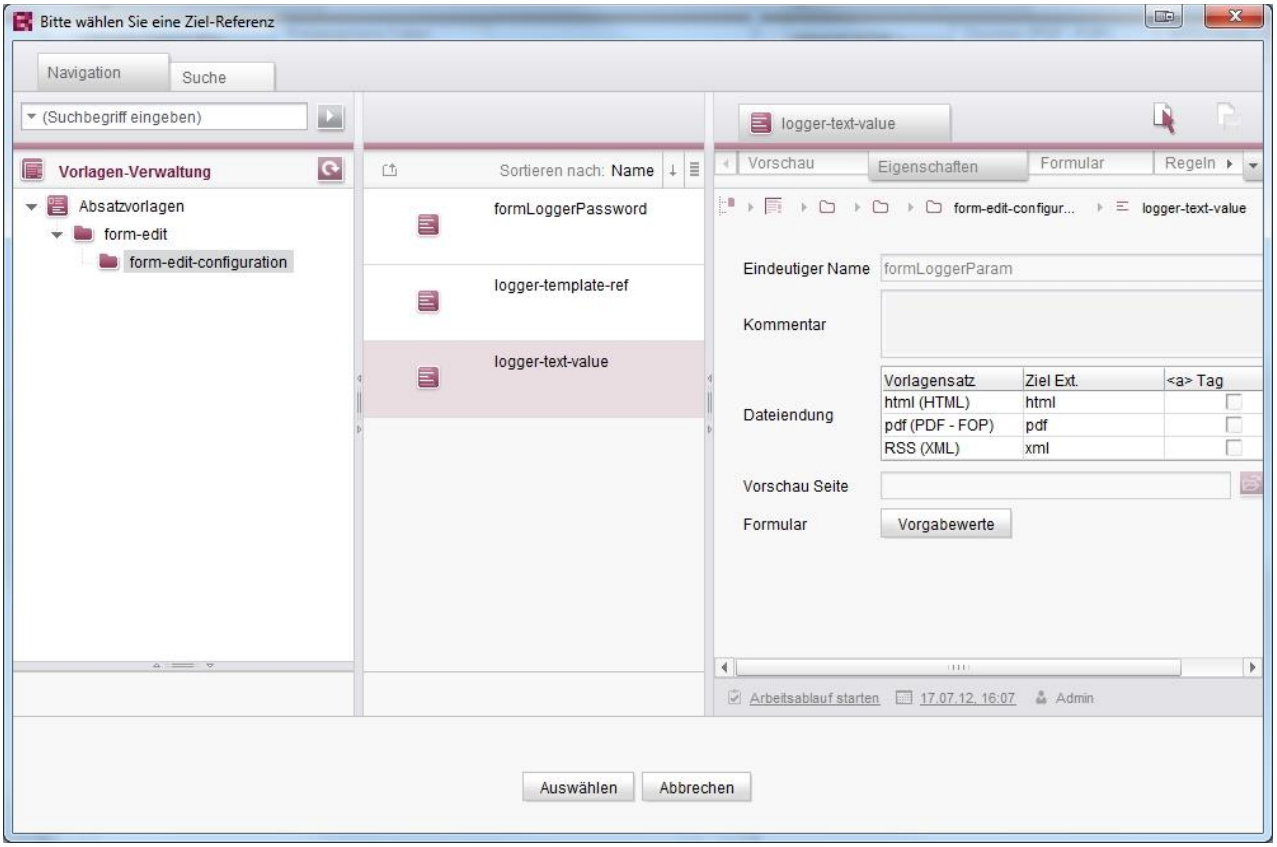

Abbildung 3-4: Parameter hinzufügen

logger-text-value: Diese Vorlage wird bis auf eine Ausnahme – die Auswahl der

Mail-Vorlage – für alle Parameter verwendet.

formLoggerPassword: Diese Vorlage kann für Parameter verwendet werden, deren Inhalt in FirstSpirit nicht sichtbar sein soll.

logger-template-ref: Diese Vorlage wird ausgewählt, wenn eine Mail-Vorlage für den MailLogger bzw. MailUploadLogger aus der Struktur gewählt werden soll.

Die Parameter, die hier verwendet werden können, werden in Kapitel 3.2 ab Seite 18 erläutert.

Default-Logger: Über diese Radio-Buttons kann ein Logger als „default"-Logger markiert werden. Ist ein oder mehrere Default-Logger vorhanden, werden diese auch für Formulare benutzt, denen, z.B. durch ein Konfigurationsfehler, kein Logger zugeordnet werden kann.

Status: Über diese Radio-Buttons kann ein Logger aktiv bzw. inaktiv geschaltet werden. Nur aktive Logger können in einem Formular als Weiterverarbeitungsmöglichkeit ausgewählt werden.

3.2 Logger-Konfiguration der Weiterverarbeitung

Jedem Logger wird ein bestimmter Weiterverarbeitungstyp (z.B. MailLogger) zugeordnet und mit entsprechenden Parametern versehen. Die verschiedenen Logger werden als Datensätze in einer Datenquelle gepflegt. Anhand der Logger-Konfiguration wird bei der Generierung die Logger-Konfigurationsdatei „fs-formlogger.ini" erzeugt.

3.2.1 Konfiguration der Log-File-Weiterverarbeitung

Aufgabe:
Ausgabe der Formulardaten in das Log-File der Servlet-Engine.

Parameter (Parametername, erwarteter Wert):

`class`	de.espirit.firstspirit.opt.formedit.ConsoleLogger
`prefix`	der Log-Ausgabe vorangestellter Text

Beispiel:

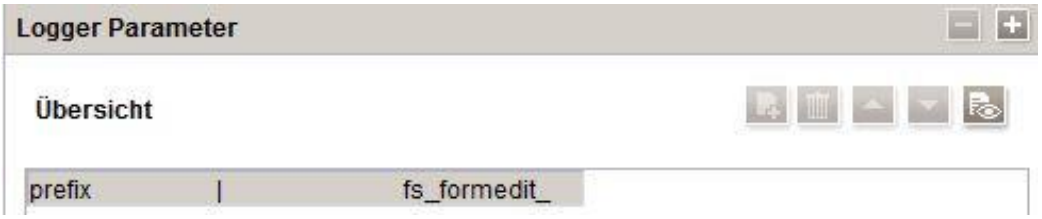

Abbildung 3-5 Parameter ConsoleLogger

Anmerkungen:

`class` Dieser Parameter wird automatisch erzeugt und muss nicht angelegt werden.

3.2.2 Konfiguration der CSV-Weiterverarbeitung

Aufgabe:
Ausgabe der Formulardaten in eine CSV-Datei.

Parameter (Parametername, erwarteter Wert):

`class` de.espirit.firstspirit.opt.formedit.CSVLogger

`logFile` (absoluter) Pfad zur CSV-Datei

`encoding` Encoding für das Versenden der E-Mail, z.B. „UTF-8“

Beispiel:

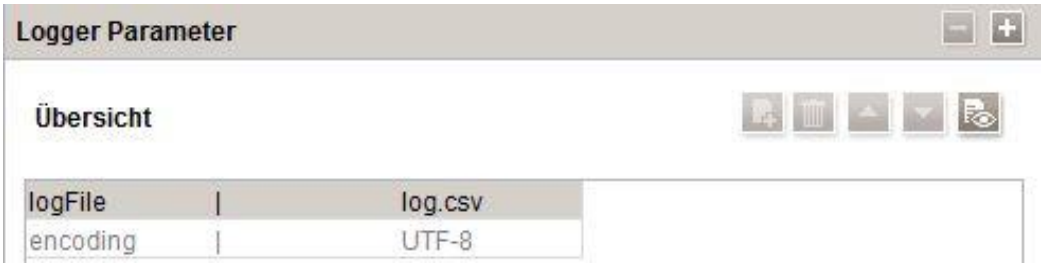

Abbildung 3-6: Parameter CSVLogger

Anmerkungen:

`class` Dieser Parameter wird automatisch erzeugt und muss nicht angelegt werden.

`logFile` Die Pfadangabe kann absolut oder relativ zur Webanwendung erfolgen. (Die angegebene Datei muss manuell angelegt werden)

`encoding` Wird dieser Parameter nicht angegeben, wird der Standardwert aus der Konfiguration der Webanwendung genutzt (siehe Kapitel 2.3.1 Seite 12).

3.2.3 Konfiguration der Datenbank-Weiterverarbeitung

Aufgabe:
Ausgabe der Formulardaten in eine (per JDBC angebundene) Datenbank.

Im Gegensatz zu den „einfachen“ CSV- und Log-File-Loggern verfügt der JdbcLogger über einige Parameter mehr. Folgende Hauptaspekte können konfiguriert werden:

JDBC-Parameter (Parametername, erwarteter Wert):

`class` de.espirit.firstspirit.opt.formedit.JdbcLogger

`driver` JDBC-Treiber z.B. „org.gjt.mm.mysql.Driver“

`user` Datenbank-Benutzer z.B. „cms“

`password` Passwort des Datenbankbenutzers

`url` JDBC-URL zur Datenbank z.B. „jdbc:mysql://localhost:3306/logging“

`table` Name der Tabelle in der Datenbank, in die geloggt werden soll.

Beispiel:

Abbildung 3-7: Parameter JdbcLogger

Anmerkungen:

`class` Dieser Parameter wird automatisch erzeugt und muss nicht angelegt werden.

Abbildungsregeln:

Für jeden Formular-Parameter kann definiert werden, in welche Tabellen-Spalte dieser eingefügt werden soll. Erfolgt keine explizite Zuweisung, so wird versucht, den Parameternamen als Spaltennamen zu verwenden. Schlägt auch dies fehl, so erfolgt nur ein Eintrag in „unmappedColumn“ (siehe unten, Punkt Weitere Parameter).

Hier gilt folgendes Schema:

formularparameter

eindeutiger Bezeichner des Formularelements

spaltenname

Name der Spalte, in die der Wert geschrieben werden soll

Beispiel:

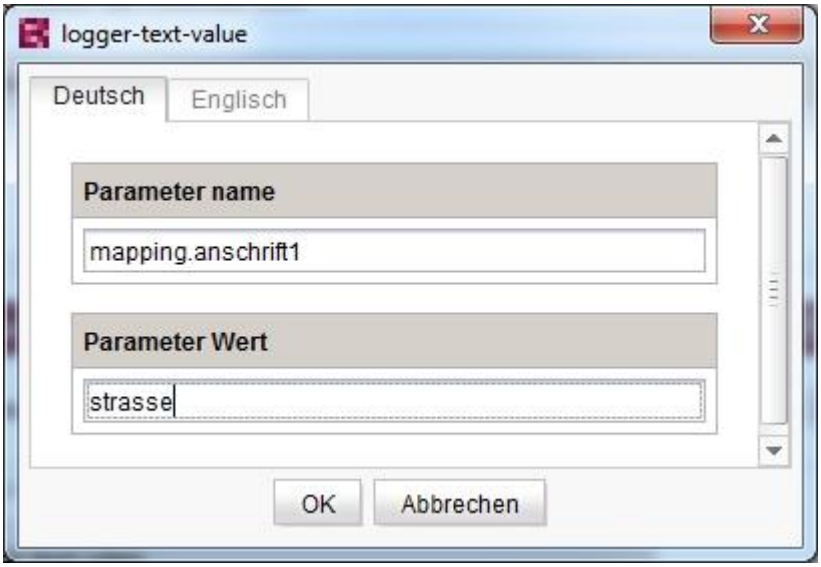

Abbildung 3-8: Parameter JdbcLogger Mapping

Der Wert des Formularelements „anschrift1“ soll in der Datenbank in das Feld „strasse“ geschrieben werden:

Weitere Parameter (Parametername, erwarteter Wert):
Zusätzlich zu den Mapping-Regeln können folgende Sonderregeln angegeben werden:

`csvColum` Name der Tabellenspalte, in die der komplette Formulardatensatz in CSV-Form eingetragen wird.

`unmappedColumn` Name der Tabellenspalte, in die alle NICHT durch Mapping Regeln verarbeiteten Formulardaten in CSV-Form eingetragen werden.

`timestampColumn` Name der Tabellenspalte, in der Datum und Uhrzeit des Request-Eingangs im Timestamp-Format gespeichert werden.

3.2.4 Konfiguration der E-Mail-Weiterverarbeitung

Aufgabe:
Ausgabe der Formulardaten in Form einer (per Datei konfigurierbaren) E-Mail (optional auch mit Dateianhang).

Der E-Mail-Logger dient zum Versand der Formulardaten per E-Mail. Für jedes Formular wird eine separate E-Mail versendet. Das Format bzw. der Text der E-Mail kann in einer (separaten) Datei konfiguriert werden. Aufgrund der formularspezifischen Logger-Konfiguration kann bei Bedarf jedem Formular eine eigene E-Mail-Konfigurationsdatei zugeordnet werden.

Parameter:

`class`	de.espirit.firstspirit.opt.formedit.MailLogger de.espirit.firstspirit.opt.formedit.MailUploadLogger
`smtpHost`	Name des E-Mail-Servers
`sender`	E-Mail-Adresse des Absenders
`mailTemplatePath`	(absoluter) Pfad zur E-Mail-Konfigurationsdatei
`encoding`	Encoding für das Versenden der E-Mail
`smtpAuth`	(optional) „true," wenn der smtp-Server eine Authentifizierung (smtpAuth) benötigt
`smtpAuthUser`	Name des Benutzers für die Authentifizierung
`smtpAuthPassword`	Passwort für die Authentifizierung

Beispiel:

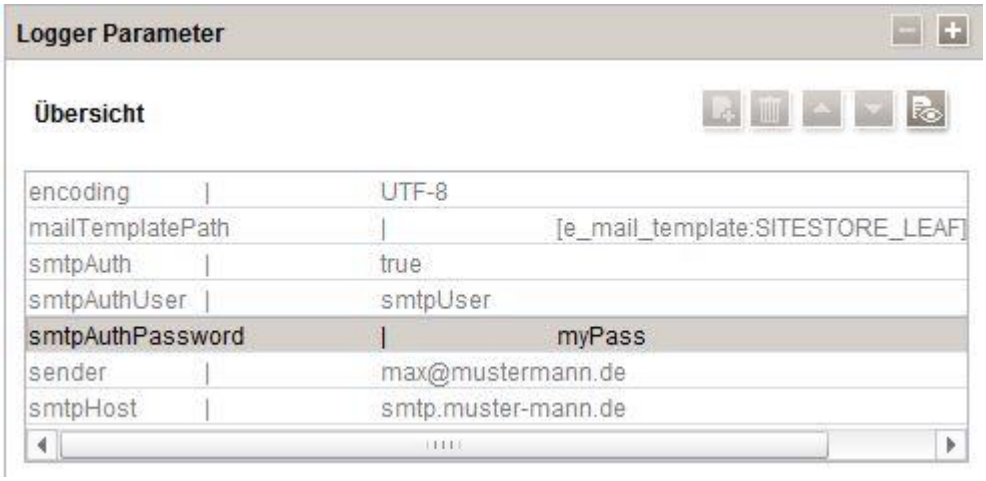

Abbildung 3-9: Parameter MailLogger

Anmerkungen:

`class` Dieser Parameter wird automatisch erzeugt und muss nicht angelegt werden.

`smtpHost` Der Hostname des Mail-Servers.

`sender` Adresse, die als Absender eingetragen wird

`mailTemplatePath` Als Vorlage sollte hier „logger-template-ref“ gewählt werden. Mit dieser kann die Vorlage aus der Struktur gewählt werden.

`encoding` Wird dieser Parameter nicht angegeben, wird der Standardwert aus der Konfiguration der Webapplikation genutzt (siehe Kapitel 2.3.1 Seite 12).

`smtpAuthUser` / `smtpAuthPassword` Ist der Parameter „smtpAuth“ mit dem Wert „true“ gesetzt, sind diese Parameter Pflichtparameter und müssen angegeben werden. Als Vorlage kann hier „logger-text-password“ gewählt werden, um die Daten verdeckt anzuzeigen.

3.2.5 Konfiguration der URL-Weiterverarbeitung

Aufgabe:
URL-Aufruf mit Parameterübergabe an eine andere Seite

Der URL-Logger dient zum Aufrufen einer anderen Seite (URL) mit der Möglichkeit, definierte Parameter des Formulars zu übergeben. Der Anwender ruft diese Seite allerdings nicht in seinem Browser auf. Ein klassischer Anwendungsfall ist z.B. Tracking.

Parameter:

`class`	de.espirit.firstspirit.opt.formedit.URLLogger
`url.sendWithOutParams`	Übergabe von Parametern (true / false)
`url.urlPrefix`	Ziel-URL (z.B. http://myserver.com/tracking.jsp)
`url.param.<bezeichner>`	eindeutiger Bezeichner der Formular-Komponente

Ähnlich wie bei dem JdbcLogger ist es nötig, die Parameter des Formulars auf neue Parameter zu mappen. Beim URLLogger werden nur die Parameter an das UrlPrefix angehängt, die auch in der Konfiguration angegeben wurden. Werden keine URL-Parameter konfiguriert, so werden alle Parameter des Formulars an die URL angehängt.

Beispiel:

Abbildung 3-10: Parameter URLLogger

Anmerkungen:

`class` Dieser Parameter wird automatisch erzeugt und muss nicht angelegt werden.

`url.param.<param>` `<param>` definiert den Parameternamen, wie er an die URL angehängt werden soll.

3.3 E-Mail-Konfigurationsdatei

Aufgabe:
Über die Seitenvorlage „mailtemplate“ wird die zu versendende E-Mail konfiguriert. Die Oberfläche ähnelt der eines E-Mail-Programms.

Abbildung 3-11: Mail-Vorlage (Kopf)

Empfänger, Cc, Bcc: Hier können eine oder mehrere Empfänger E-Mail-Adressen angegeben werden. Bei Verwendung von mehreren Adressen sind diese durch ein “;“ (Semikolon) zu trennen.

Reply-To: Hier kann eine Antwort-E-Mail-Adresse eingegeben werden. Diese wird benutzt, wenn der Anwender in seinem E-Mail-Programm auf „Antworten“ klickt.

Sender: Hier kann eine E-Mail-Adresse definiert werden, die als Absender angezeigt wird. Alternativ kann aber auch über %parameter% (hier: %email%) auf ein Formularelement zurückgegriffen werden, die eine gültige E-Mail-Adresse enthält. Wird hier ein Wert angegeben, wird der in der Logger-Konfiguration angegebene Wert „sender“ überschrieben.

Betreff: Hier kann der Betreff der zu versendenden E-Mail angegeben werden.

Upload Attachments: Hier können die Dateianhänge konfiguriert werden. Dies kann zum einen über %parameter% geschehen oder über %all%.

%parameter%	Bei der Angabe dieses Parameters wird nur die Datei an die E-Mail angehängt, die über das Formularelement mit dem Bezeichner „parameter“ übertragen wurden. Mehrere Dateien müssen mit Komma (,) getrennt werden. Beispiel: %lebenslauf%,%foto%
%all%	Bei der Eingabe werden alle mit dem Formular abgesendeten Dateien an die E-Mail angehängt.

Text: Nach den für den E-Mail-Header benötigten Angaben folgt dann der Text der E-Mail. Dieser Text kann Platzhalter der Form %name% enthalten, mit deren Hilfe auf Werte aus dem Formular zugegriffen werden kann.

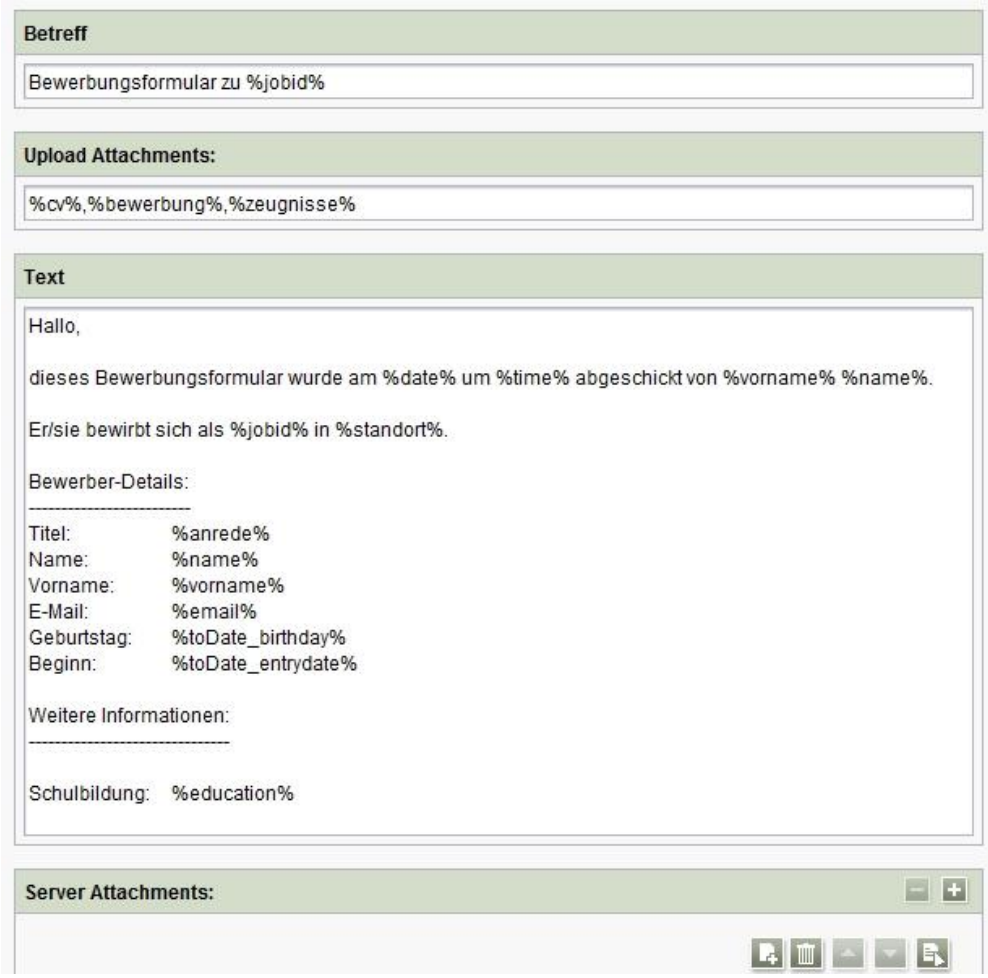

Abbildung 3-12: Mail-Vorlage (Nachrichtenteil)

Folgende Parameter stehen zur Verfügung:

%date%	Datum der Formular-Absendung
%time%	Uhrzeit der Formular-Absendung
%csv%	Liste alle Formularparameter als CSV
%unmapped%	Liste aller nicht bearbeiteten Formularparameter als CSV (s.u.)

Zusätzlich zu diesen Parametern kann auf jeden Formular-Parameter zugegriffen werden. Dazu wird die %-Notation und der Parameter-Name verwendet: %parameter%

Formulardaten, die zwar gesendet wurden, allerdings nicht in der Mail-Vorlage über %-Notation ausgegeben wurden, können über den Parameter %unmapped% ausgegeben werden.

Server Attachments: Über eine FS_LIST in einer Seite vom Typ *mailtemplate* kann der Redakteur Dateien auswählen, die automatisch mit einer Mail versendet werden.

Ein Szenario wäre beispielsweise eine Anmeldung, die über ein Formular erfolgt. Als Antwort könnte dann eine Bestätigungs-Mail versendet werden, die beispielsweise AGBs enthält oder ähnliches. Diese ABGs könnten vorher in die FS_LIST eingepflegt werden.

Zur Auswahl der Dateiordner muss das Linktemplate der FS_LIST-Elemente angepasst werden. Hierbei muss für die FS_REFERENCE-Komponente der Ordner angegeben werden, aus dem gewählt werden soll, z.B. würde <FOLDER name="root" …> eine Auswahl aller Medien in der Medienverwaltung erlauben.

Weiter müssen unter Globale Einstellungen/Projekteinstellungen folgende Variablen als Textfelder (CMS_INPUT_TEXT) definiert werden:

ps_webappContentPath	Pfad zu Webanwendung
ps_webappContentFolder	Ordner innerhalb *ps_webappContentPath*, in den deployt wird

Diese Variablen werden in der Linkvorlage im html Kanal genutzt.

3.4 Autocompleterequest

Bei dieser Seitenvorlage handelt es sich um eine funktionsfähige **Beispiel-Vorlage.** Es ist auch möglich, Daten gegen eine Datenbank oder ähnliches zu prüfen. Diese Datei ist **nicht allgemein gültig und muss projektspezifisch angepasst werden**, sofern es sich bei der Quelle nicht um eine XML-Datei handelt. Weitere Informationen siehe Kapitel 6 Seite 69.

Aufgabe:
Diese Seitenvorlage dient zur Verarbeitung einer Eingabequelle wie z.B. XML. Bei einer Eingabe des Benutzers in ein Autovervollständigen-Formularfeld wird die in dieser Seite angegebene Quelle durchsucht und die Ergebnisse dem Benutzer eingeblendet.

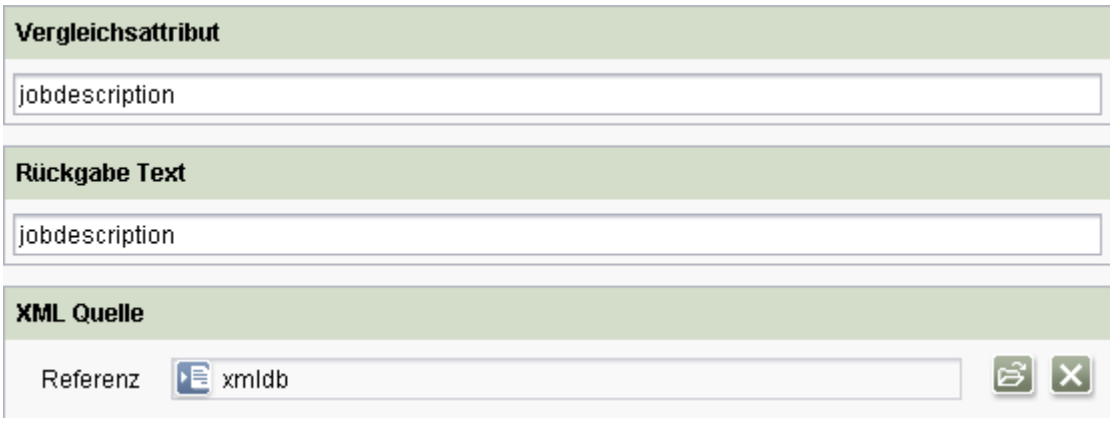

Abbildung 3-13: Autocompleterequest

Vergleichsattribut: Hier wird das Attribut innerhalb der XML-Quelldatei angegeben, welches mit den Eingaben des Nutzers verglichen werden soll.

Rückgabe Text: Hier kann definiert werden, welches Attribut der XML-Quelldatei dem Benutzer als Rückgabewert angezeigt werden soll.

XML Quelle: Hier muss die XML-Quelldatei ausgewählt werden, die die Ergebnisse bereitstellen soll.

3.5 Konfigurationsdatei „fs-formlogger.ini“

Diese Konfigurationsdatei beinhaltet alle Informationen ihrer Formulare bzw. die Zuordnung zu den Weiterverarbeitungskonfigurationen und diese selbst. Der Inhalt dieser Datei wird generisch von FirstSpirit bei der Generierung erzeugt.

Damit diese Datei richtig erzeugt werden kann, muss in der Inhalte-Verwaltung eine Seite auf Basis der Vorlage „logger-ini-file“ erstellt werden, und in diese Seite muss die UID der „form start“-Vorlage bzw. den von Ihnen erzeugten Vorlagen, die die Funktion der „form start“-Vorlage erfüllen, eingetragen werden. Die UID wird mit dem Tastaturkürzel „ALT + P“ bei selektierter Absatzvorlage angezeigt. Außerdem ist ebenfalls die UID der Datenquelle anzugeben, die die Logger-Konfiguration enthält.

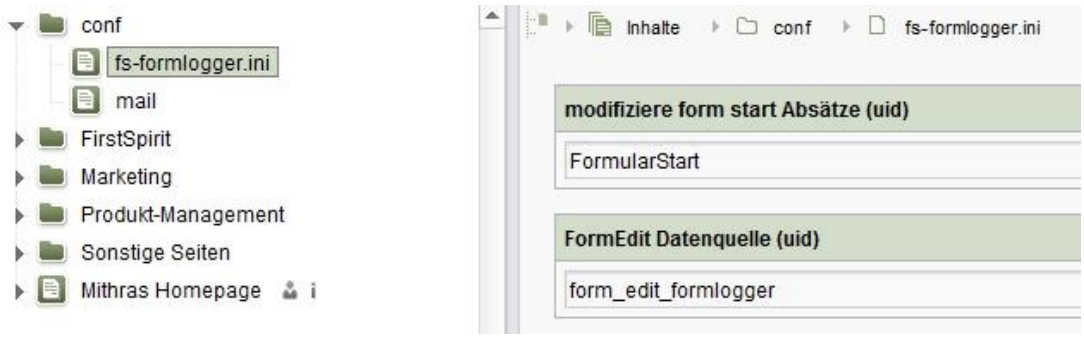

Abbildung 3-14: Seite „fs-formlogger.ini“

Bei der Referenzierung in der Struktur ist zu beachten, dass die Datei an der Stelle abgelegt ist, die bei der Konfiguration der Webkomponente (siehe Kapitel 2.3.1 Seite 12) angegeben wurde.

Weiterhin ist der Dateiname der Seite in der Strukturverwaltung richtig zu setzen. Innerhalb dieses Dokuments wird der Name „fs-formlogger“ verwendet:

Abbildung 3-15: Dateiname der Konfigurationsdatei

4 Formularerstellung

4.1 Formularaufbau

Um die Funktionalität eines Formulars zu gewährleisten, muss eine bestimmte Reihenfolge der Absatzvorlagen bei der Erstellung des Formulars eingehalten werden. Die Absatzvorlage **form-start** (Kapitel 4.2.1 Seite 32) markiert immer den Anfang und die Absatzvorlage **form-end** (Kapitel 4.2.4 Seite 37) immer das Ende eines Formulars. Innerhalb dieser beiden Absätze können beliebig viele Absätze der Typen **form-block** (Kapitel 4.2.2 Seite 36) und **form-divider** (Kapitel 4.2.3 Seite 36) auftreten.

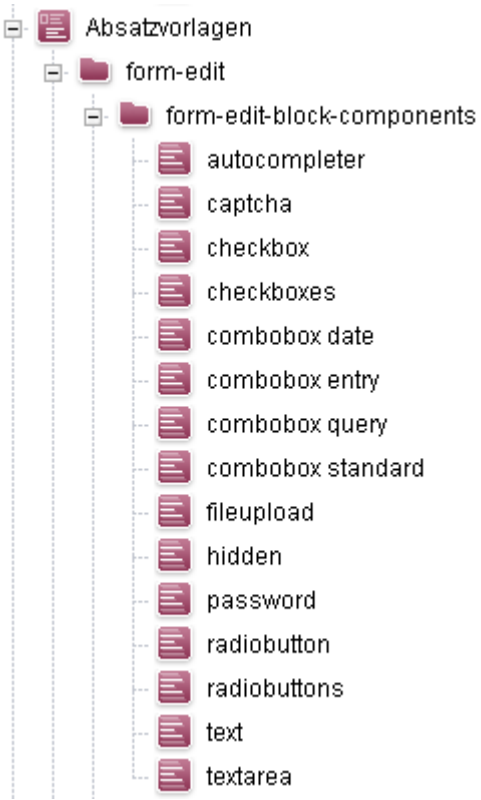

Abbildung 4-1: Formularvorlagen

Innerhalb eines **form-block**-Elements können beliebig viele Formular-Elemente enthalten sein.

Die Absatzvorlagen des Formular-Editors stellen alle in HTML vorhandenen Formular-Elemente zur Verfügung und geben außerdem ausreichende Möglichkeiten, die Komponenten zu konfigurieren. Für die Verwendung der

Komponenten muss zuvor eine Anpassung an spezifische Designvorgaben erfolgen. Dies kann einerseits durch die direkte Anpassung des HTML-Codes in den Absatzvorlagen und / oder durch die Definition von Cascading Style Sheets in der eingebundenen Stylesheet-Datei geschehen. Jeder Formular-Komponente kann individuell eine Stylesheet-Klasse zugewiesen werden.

4.2 Formularkonfiguration

4.2.1 form-start

Die Absatzvorlage „form-start” leitet ein neues Formular ein. Hier können grundlegende Konfigurationen vorgenommen werden, die ausschließlich dieses Formular betreffen.

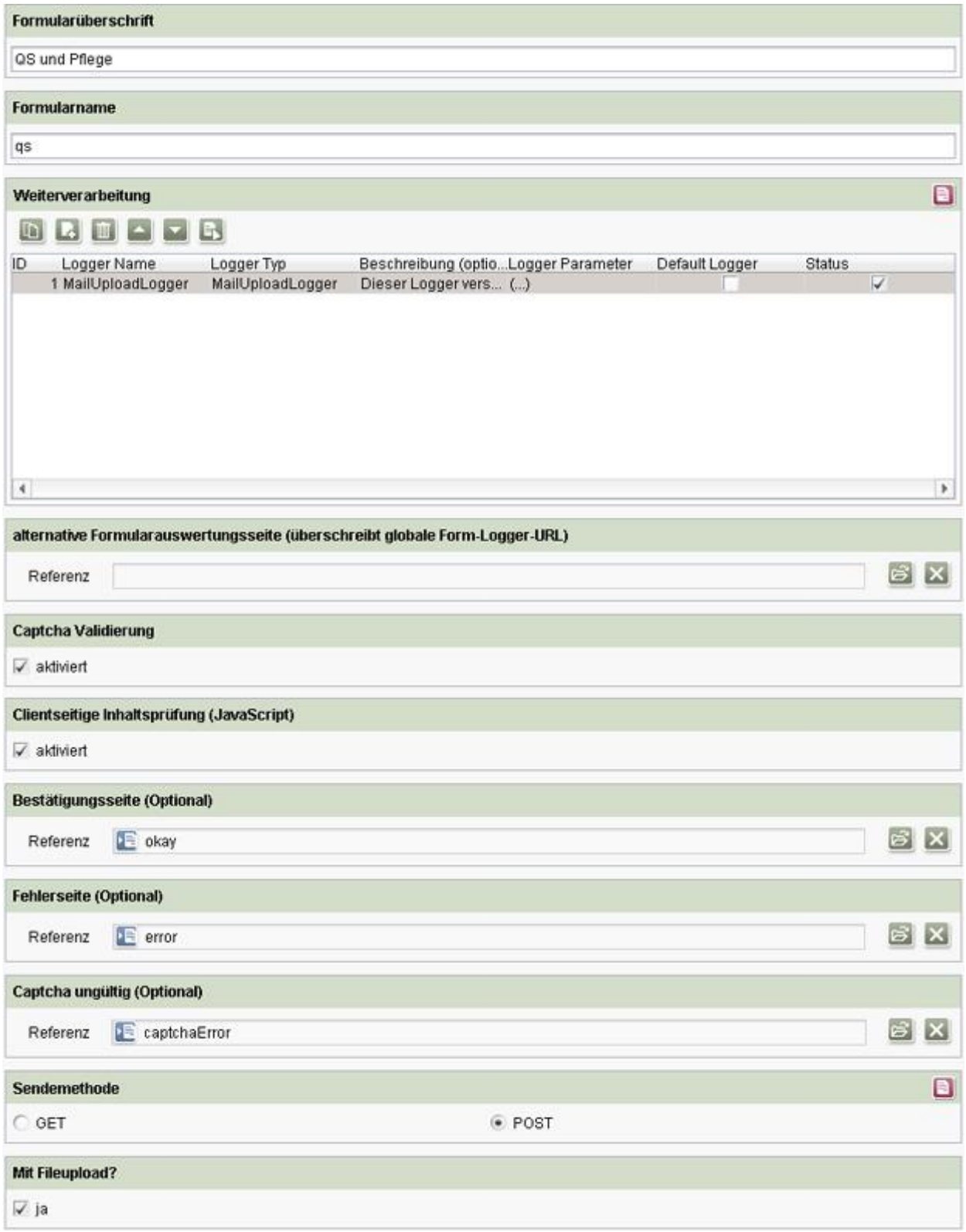

Abbildung 4-2: form-start Absatz

Formularüberschrift (Text): Darstellung der Überschrift des Formulars

Formularname (Text): Eindeutiger Bezeichner für das Formular („name“-Attribut des „form“-Elementes)

DieserBezeichner darf weder Leerzeichen noch Sonderzeichen enthalten, da der Formularname als Teil des Servletaufrufs verwendet wird.

Weiterverarbeitung (ContentList): Auswahl und Anzeige der Logger für dieses Formular

alternative Formularauswertungsseite (Seitenreferenz): Wenn kein Logger gewünscht ist, kann hier optional eine Formular-Auswertungsseite angegeben werden.

Captcha Validierung (Checkbox): Aktivierung der serverseitigen Captcha-Validierung. Ist diese Checkbox aktiviert, muss das Formular das Formelement Captcha beinhalten.

Der Status wird dem Servlet standardmäßig per verstecktem Formularfeld (input type=“hidden“) mitgeteilt. Dies kann von Spamversendern erkannt werden. Es wird empfohlen, in der Vorlage statt

<input type="hidden" name="useCaptcha" value="true" />

folgenden jsp-Code zu verwenden:

<% session.setAttribute(“useCaptcha“,“true“); %>

Clientseitige Inhaltsüberprüfung (Checkbox): Aktivierung der clientseitigen Inhaltsüberprüfung

Bestätigungsseite (Optional) (Seitenreferenz): Diese formularspezifische Bestätigungsseite wird angezeigt, wenn die E-Mail erfolgreich versendet wurde. Wird hier eine Seite ausgewählt, wird die globale Konfiguration in der web.xml überschrieben. In der Vorlage kann durch ein der Referenz vorangestelltes „forward:“ oder „redirect:“ die Art der Weiterleitung bestimmt werden. Ist kein Prefix angegeben, erfolgt ein Redirect auf die Zielseite.

```
<input type="hidden" name="okRedirect"
```

```
value="forward:$CMS_REF(st_noerrorPage)$" />

<input type="hidden" name="errorRedirect"
value="redirect:$CMS_REF(st_errorPage)$" />
```

Standardmäßig ist in der Vorlage kein Prefix gesetzt.

Fehlerseite (Optional) (Seitenreferenz): Diese formularspezifische Bestätigungsseite wird angezeigt, wenn die E-Mail nicht erfolgreich versendet wurde. Wird hier eine Seite ausgewählt, wird die globale Konfiguration in der web.xml überschrieben. In der Vorlage kann durch ein der Referenz vorangestelltes „forward:“ oder „redirect:“ die Art der Weiterleitung bestimmt werden. Ist kein Prefix angegeben, erfolgt ein Redirect auf die Zielseite.

Captcha ungültig (Optional) (Seitenreferenz): Diese formularspezifische Seite wird angezeigt, wenn der Anwender keine oder eine falsche Eingabe tätigt. Wird hier keine Seite ausgewählt, wird stattdessen die „Fehlerseite“ (s. o.) angezeigt.

Sendemethode (RadioButton): Mit dieser Komponente legen Sie den Übertragungsmodus der Formularwerte fest. Vorbelegung: „POST“

Fileupload (Checkbox): Wenn diese Option gesetzt ist, können Dateien über das Formular an den Webserver übergeben werden.

4.2.2 form-block

Innerhalb eines „form-block“-Elements können beliebig viele Formular-Elemente angelegt werden. Dabei ist die Reihenfolge der Komponenten unerheblich für die Funktionalität des Formulars. Jede Komponente kann individuell konfiguriert werden. So können beispielsweise Stylesheet-Klassen verwendet und die Anzeige-Breite und Anzeige-Höhe des Formulars definiert werden. Genaue Informationen und Konfigurationsanleitungen finden sich in Kapitel 4.3 ab Seite 38.

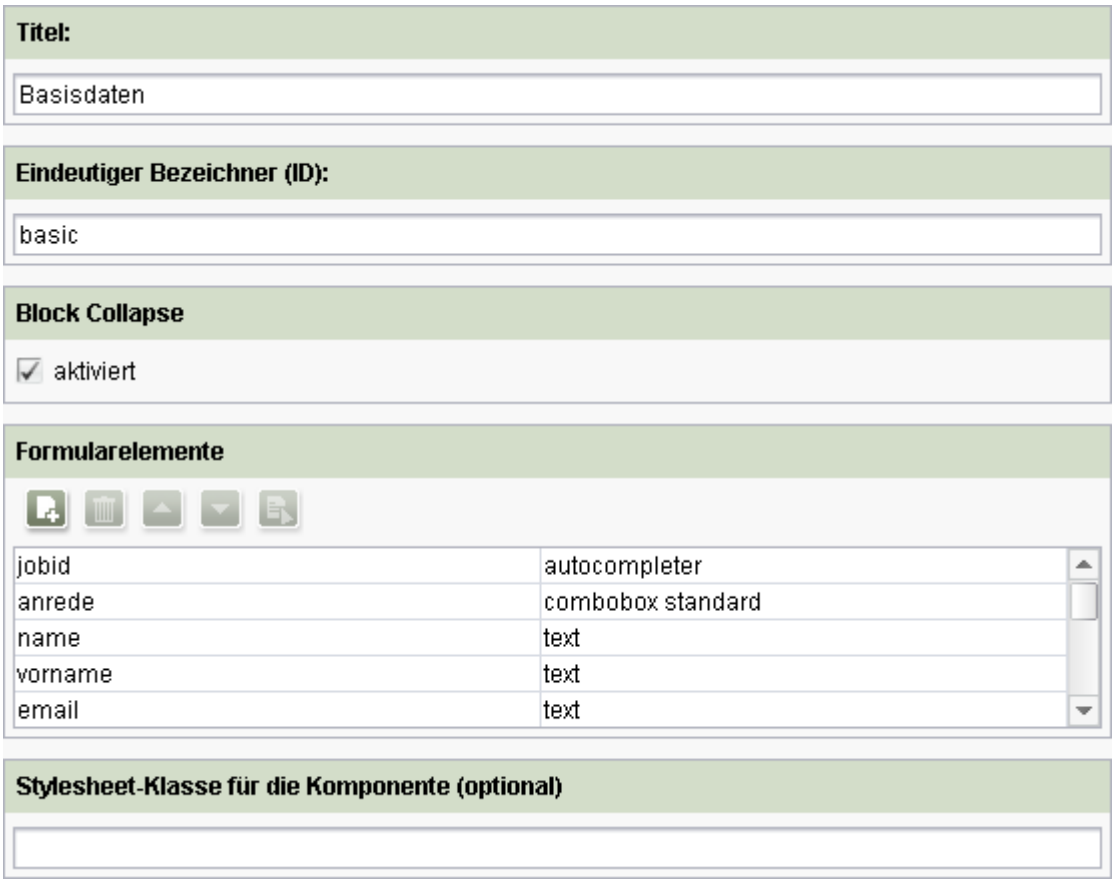

Abbildung 4-3: form-block Absatz

4.2.3 form-divider

Das Element „form-divider“ erzeugt eine grafische Trennung innerhalb des Formulars und bietet ansonsten keine Funktionalität.

4.2.4 form-end

Die Absatzvorlage „form-end“ definiert das Ende des Formulars.

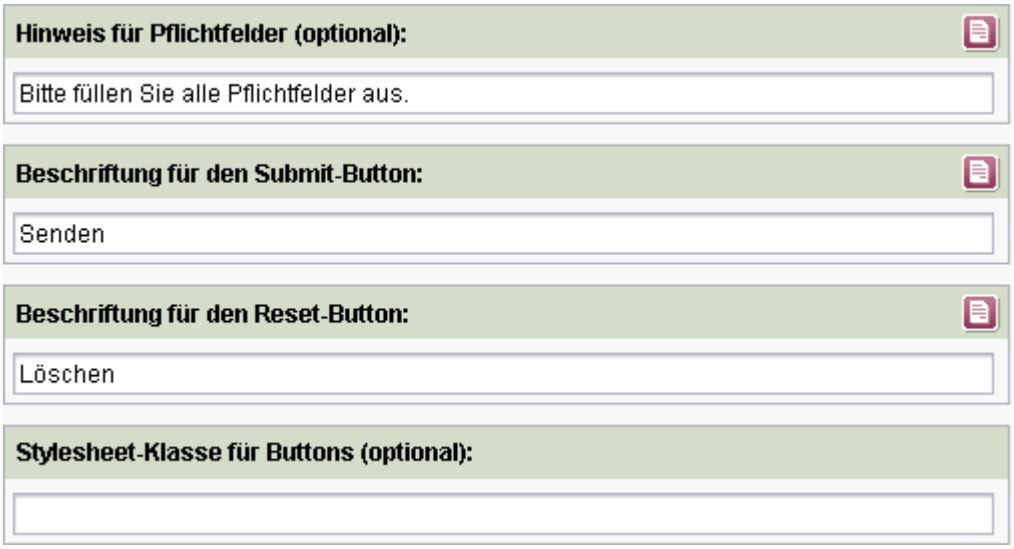

Abbildung 4-4: form-end Absatz

Hinweis für Pflichtfelder: Hier kann ein Hinweis für Pflichtfelder angeben werden, der unterhalb der Formularfelder angezeigt wird.

Beschriftung für den Submit-/Reset-Button: Geben Sie die Bezeichnung für den Button ein, mit dem die Formulardaten abgeschickt bzw. alle Eingaben des Formulars gelöscht werden. Werden hier keine Eingaben vorgenommen, wird kein Button generiert.

Stylesheet-Klasse für Buttons (optional): Hier können Sie den Namen einer Stylesheet-Klasse für das Design der Buttons angeben. Wird keine Klasse angegeben, so werden die Buttons im Standard-Look-and-Feel des Browsers bzw. Ihrer Stylesheet-Datei, angezeigt.

4.3 Verfügbare Formular-Elemente

Der Formular-Editor ermöglicht es dem Redakteur, das komplette Set an HTML-Formular-Elementen zu verwenden. Wie bereits erwähnt, empfiehlt es sich vor der Verwendung der Komponenten, eine Anpassung im HTML-Quellcode im Zusammenspiel mit der Stylesheet-Datei vorzunehmen. So können individuelle Designvorlagen eingehalten werden und stehen an jeder Stelle der Webseite zur Verfügung. Die HTML-Tags der Formular-Elemente sollten aber von Änderungen unberührt bleiben, um die korrekte Funktionalität des JavaScript-Checks zu gewährleisten.

Die Formular-Komponenten können nur in der Absatzvorlage „form-block“ verwendet werden!

Um ein neues Formular-Element hinzuzufügen, wählen Sie einen bereits erstellten Absatz des Typs „form-block“ aus und fügen der Content-Area-List ein neues Formular-Element hinzu:

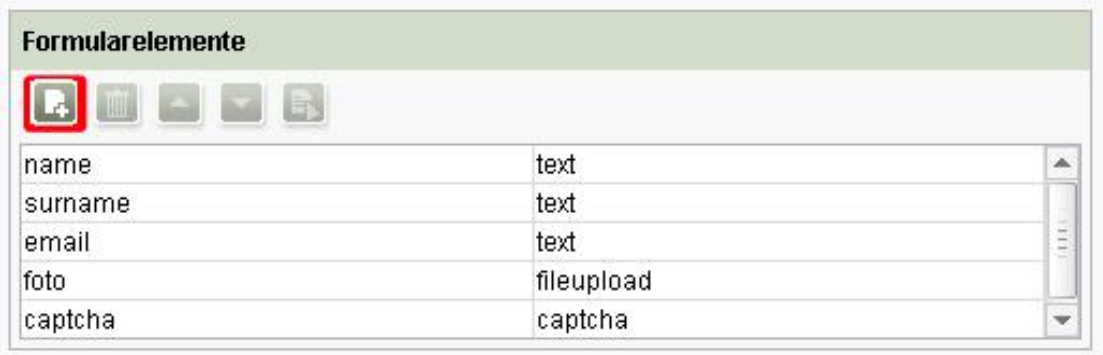

Abbildung 4-5: Formular-Elemente

Im Folgenden finden Sie eine Übersicht über die verfügbaren Standard-Elemente und deren Konfigurationsmöglichkeiten und Funktionen.

Über den eindeutigen Bezeichner bzw. den eindeutigen Gruppenbezeichner kann später auf den Inhalt des Formularfelds zugegriffen werden. Wenn z.B. ein Mail- / MailUploadLogger benutzt wird, geschieht das über %eindeutigerBezeichner%

4.3.1 Formular-Komponente „Text“

Diese Komponente stellt dem Benutzer ein Formular-Textfeld zur Verfügung. Damit die Auswertung des Formulars und auch die Inhaltsüberprüfung ordnungsgemäß funktionieren kann, muss der Komponente ein in diesem Formular eindeutiger Bezeichner („name“-Attribut) zugewiesen werden.

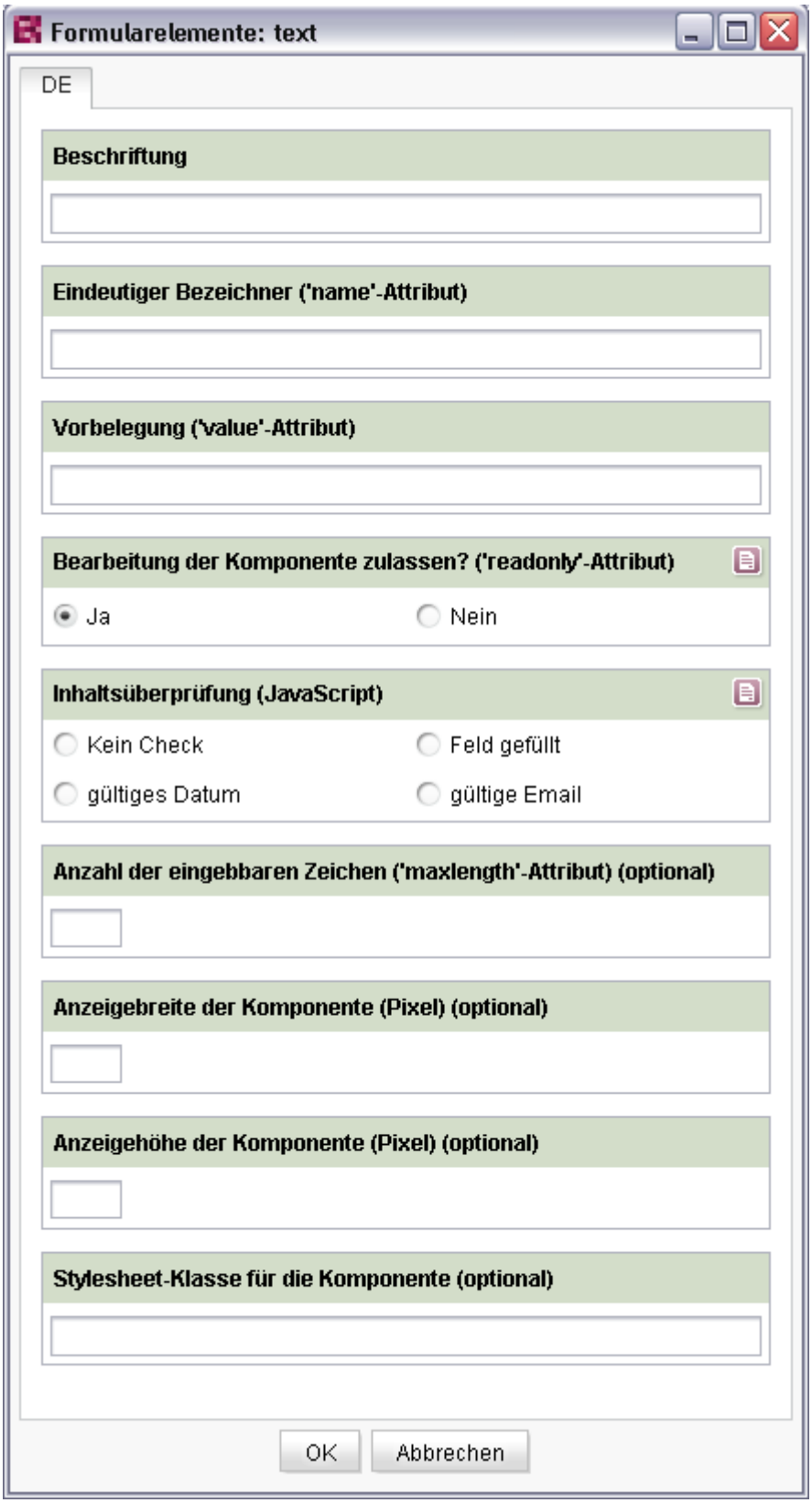

Abbildung 4-6: Formular-Komponente „Text“

Beschriftung: In dieses Textfeld wird der Titel des Formularfeldes eingegeben.

Eindeutiger Bezeichner (‚name'-Attribut): In dieses Textfeld wird der eindeutige Bezeichner eingetragen.

Vorbelegung (‚value'-Attribut): In dieses Textfeld kann Text eingegeben werden, mit dem das Formularfeld gefüllt ist.

Bearbeitung zulassen (‚readonly'-Attribut): Über diese Radiobuttons wird gesteuert, ob der Benutzer das Formularfeld bearbeiten darf.

Inhaltsprüfung (gefüllt bzw. gültiger Wert): Über diese Radiobuttons kann eine Prüfung erstellt werden, die auf Inhalt bzw. auf gültige Daten prüft.

Anzahl Zeichen (‚maxlength'-Attribut): Mithilfe dieses Textfelds kann die Anzahl der Zeichen definiert werden, die maximal eingegeben werden darf.

Anzeigebreite: In diesem Feld kann die Breite des Formularfeldes in Pixel angegeben werden.

Anzeigehöhe: In diesem Feld kann die Höhe des Formularfeldes in Pixel angegeben werden.

Stylesheet-Klasse (‚class'-Attribut: In dieses Textfeld kann der Name einer CSS-Klasse eingegeben werden.

4.3.2 Formular-Komponente „Textarea"

Die Formular-Komponente „Textarea" stellt dem Benutzer ein mehrzeiliges Texteingabefeld zur Verfügung. Damit die Auswertung des Formulars und auch die Inhaltsüberprüfung ordnungsgemäß funktionieren kann, muss der Komponente ein in diesem Formular eindeutiger Bezeichner („name"-Attribut) zugewiesen werden.

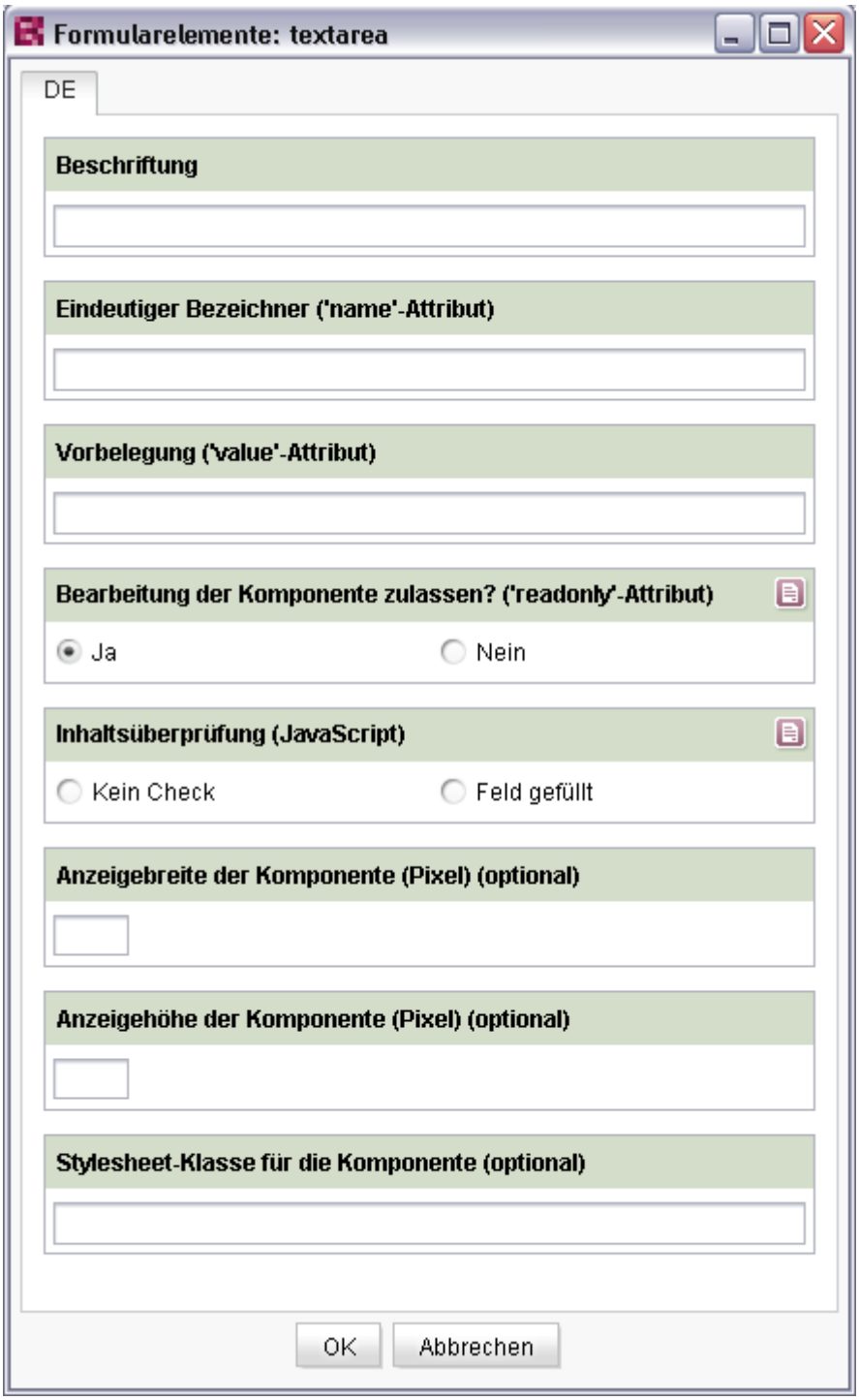

Abbildung 4-7: Formular-Komponente „Textarea“

Beschriftung: In dieses Textfeld wird der Titel des Formularfeldes eingegeben.

Eindeutiger Bezeichner (‚name’-Attribut): In dieses Textfeld wird der eindeutige Bezeichner eingetragen.

Vorbelegung (‚value’-Attribut): In dieses Textfeld kann Text eingegeben werden,

mit dem das Formularfeld gefüllt ist.

Bearbeitung zulassen (‚readonly'-Attribut): Über diese Radiobuttons wird gesteuert, ob der Benutzer das Formularfeld bearbeiten darf.

Inhaltsprüfung (gefüllt bzw. gültiger Wert): Über diese Radiobuttons kann eine Prüfung erstellt werden, die auf Inhalt bzw. auf gültige Daten prüft.

Anzeigebreite: In diesem Feld kann die Breite des Formularfeldes in Pixel angegeben werden.

Anzeigehöhe: In diesem Feld kann die Höhe des Formularfeldes in Pixel angegeben werden.

Stylesheet-Klasse (‚class'-Attribut): In diesem Textfeld kann der Name einer CSS-Klasse angegeben werden.

4.3.3 Formular-Komponente „RadioButtons"

Mit dieser Formular-Komponente können HTML-Radiobuttons erzeugt werden. Beim Hinzufügen der Radiobuttons im Bereich „Optionen" sollte darauf geachtet werden, dass für jeden Radiobutton eine aussagekräftige Vorbelegung angegeben wird. Die Radiobuttons werden nacheinander mit der entsprechenden Bezeichnung (Beschriftung) auf der rechten Seite der Komponente dargestellt.

4.3.3.1 Teilkomponente „RadioButtons"

Diese Komponente kann innerhalb eines „form-block"-Absatzes eingefügt werden. In diesem Absatz werden die für alle Radiobuttons gleichen Werte gesetzt. Das Hinzufügen der eigentlichen RadioButtons erfolgt innerhalb einer ContentAreaList in diesem Absatz.

Abbildung 4-8: Formular-Komponente „RadioButtons“

Beschriftung: In dieses Textfeld wird der Titel des Formularfeldes eingegeben.

Gruppenbezeichner (‚name’-Attribut): In dieses Textfeld wird der eindeutige Bezeichner eingetragen.

Inhaltsprüfung (gefüllt): Über diese Radiobuttons kann eine Prüfung erstellt werden, die prüft, ob ein Eintrag ausgewählt wurde.

Pflichtfeldhinweis: Für diesen Formularfeldtyp muss eine Meldung angegeben werden, die erscheint, wenn die Inhaltsprüfung fehlgeschlagen ist.

Stylesheet-Klasse (‚class'-Attribut): In dieses Textfeld kann der Name einer CSS-Klasse eingegeben werden.

4.3.3.2 Teilkomponente „RadioButton“

Diese Komponente kann ausschließlich innerhalb der Formular-Komponente „RadioButtons“ verwendet werden. Sie dient dazu, die eigentlichen RadioButtons innerhalb der o.g. Komponente hinzuzufügen.

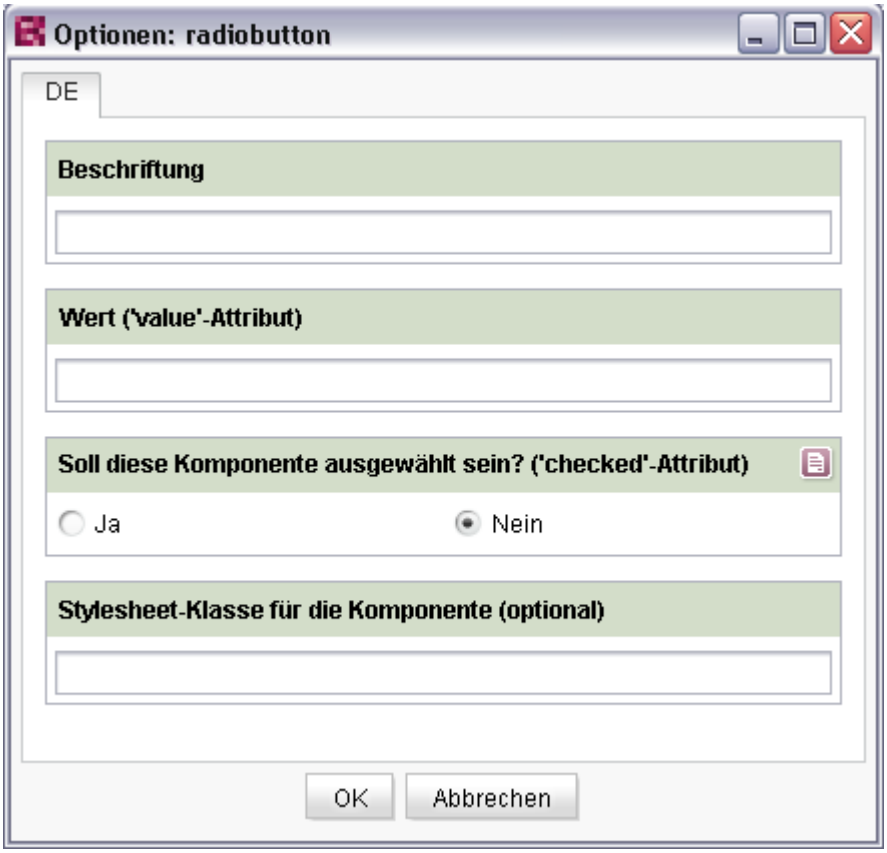

Abbildung 4-9: Formular-Komponente „RadioButton“

Beschriftung: In diesem Textfeld wird die Beschriftung der Option eingegeben.

Wert (‚value'-Attribut): In diesem Textfeld kann der Wert dieser Option angegeben werden.

Vorauswahl (‚checked'-Attribut): Über diese Radiobuttons kann eine Vorauswahl getroffen werden, ob dieses Formularfeld ausgewählt sein soll.

Stylesheet-Klasse (‚class'-Attribut): In diesem Textfeld kann der Name einer CSS-Klasse eingegeben werden.

4.3.4 Formular-Komponente „Checkboxes“

Mit dieser Formular-Komponente können HTML-Checkboxen erzeugt werden. Beim Hinzufügen der Checkboxen im Bereich „Optionen“ sollte darauf geachtet werden, dass für jede Checkbox ein aussagekräftiger Wert angegeben wird. Die Checkboxen werden nacheinander mit der entsprechenden Bezeichnung (Beschriftung) auf der rechten Seite der Komponente dargestellt.

4.3.4.1 Teilkomponente „Checkboxes“

Diese Komponente kann innerhalb eines „form-block“-Absatzes eingefügt werden. In diesem Absatz werden die für alle Checkboxes gleichen Werte gesetzt. Das Hinzufügen der eigentlichen Checkboxen erfolgt innerhalb einer ContentAreaList in diesem Absatz.

Abbildung 4-10: Formular-Komponente „Checkboxes“

Beschriftung: In diesem Textfeld wird der Titel des Formularfeldes eingegeben.

Gruppenbezeichner (‚name’-Attribut): In diesem Textfeld wird der eindeutige Bezeichner eingetragen.

Inhaltsüberprüfung (gefüllt): Über diese Radiobuttons kann eine Prüfung erstellt werden, die prüft, ob mindestens eine Checkbox dieser Gruppe ausgewählt wurde.

Pflichtfeldhinweis: Für diesen Formularfeldtyp muss eine Meldung angegeben werden, die erscheint, wenn die Inhaltsprüfung fehlgeschlagen ist.

Stylesheet-Klasse (‚class'-Attribut): In diesem Textfeld kann der Name einer CSS-Klasse eingegeben werden.

4.3.4.2 Teilkomponente „Checkbox“

Diese Komponente kann ausschließlich innerhalb der Formularkomponente „Checkboxes“ verwendet werden. Sie dient dazu, die eigentlichen Checkboxen innerhalb der o.g. Komponente hinzuzufügen.

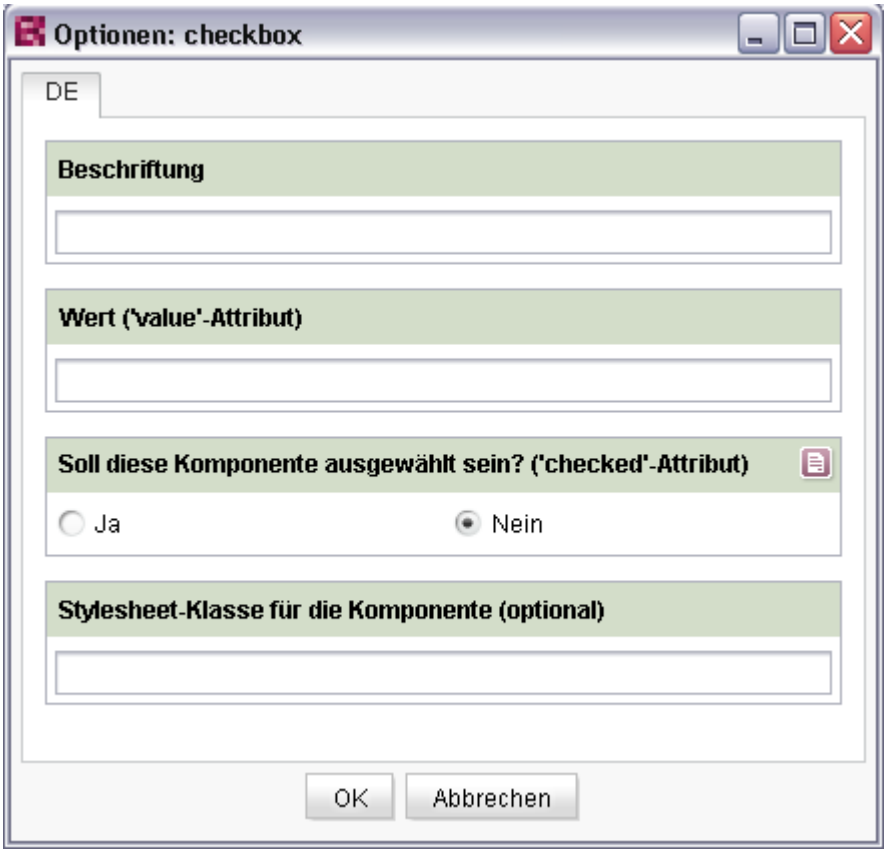

Abbildung 4-11: Formular-Komponente „Checkbox“

Beschriftung: In diesem Textfeld wird die Beschriftung der Option eingegeben.

Wert (‚value'-Attribut): In diesem Textfeld kann der Wert dieser Option angegeben werden.

Vorauswahl (‚checked'-Attribut): Über diese Radiobuttons kann eine Vorauswahl getroffen werden, ob diese Checkbox ausgewählt sein soll.

Stylesheet-Klasse (‚class'-Attribut): In diesem Textfeld kann der Name einer CSS-Klasse eingegeben werden.

4.3.5 Formular-Komponente „Password“

Mit dieser Komponente kann dem Benutzer ein Passwortfeld zur Verfügung gestellt werden. Auch hier muss der Redakteur einen eindeutigen Bezeichner für die Komponente vergeben. Die Eingabe von Zeichen wird in der Komponente als „*“ dargestellt.

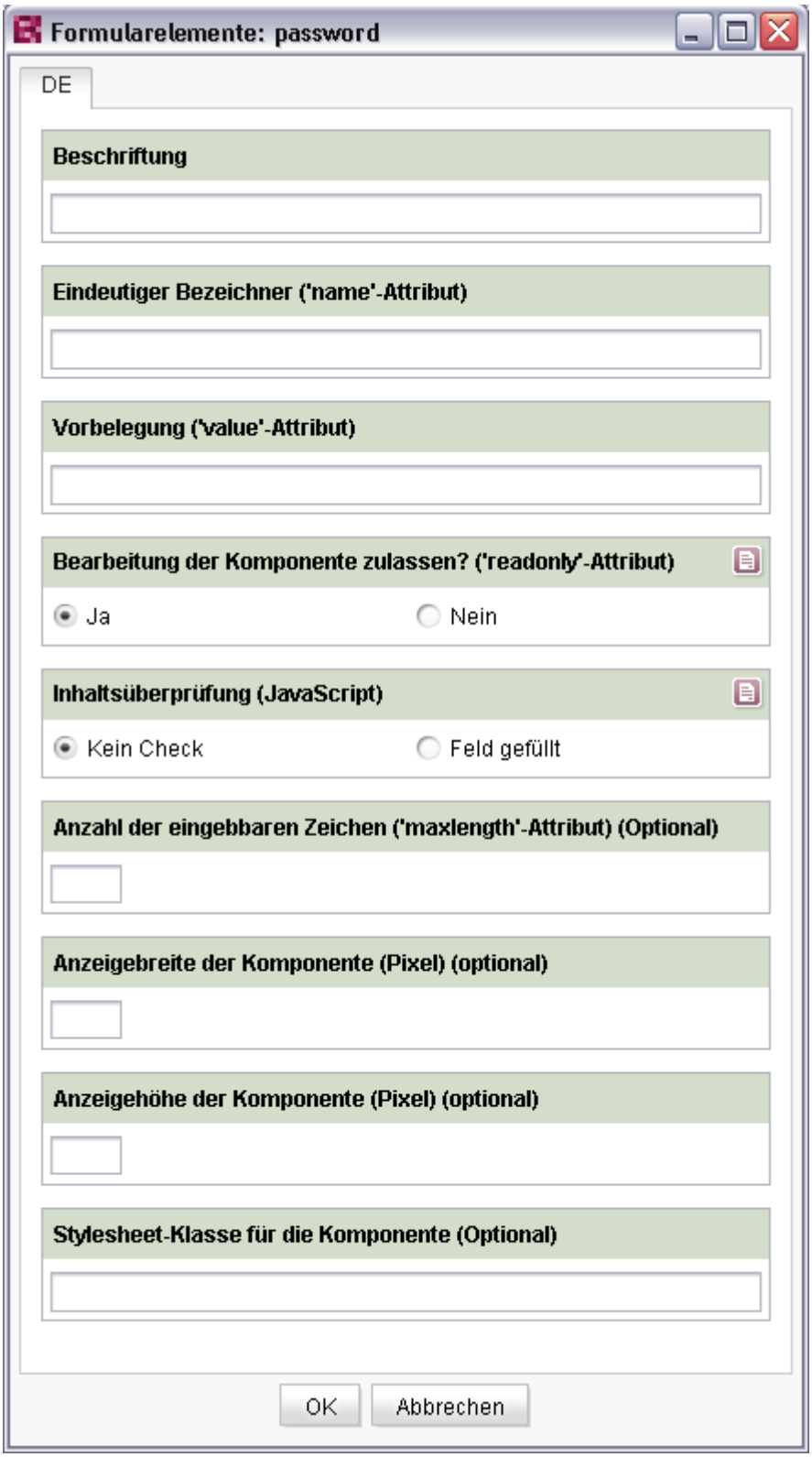

Abbildung 4-12: Formular-Komponente „Password“

Beschriftung: In dieses Textfeld wird der Titel des Formularfeldes eingegeben.

Eindeutiger Bezeichner (‚name'-Attribut): In dieses Textfeld wird der eindeutige Bezeichner eingetragen.

Vorbelegung (‚value'-Attribut): In dieses Textfeld kann Text eingegeben werden, mit dem das Formularfeld gefüllt ist.

Bearbeitung zulassen (‚readonly'-Attribut): Über diese Radiobuttons wird gesteuert, ob der Benutzer das Formularfeld bearbeiten darf.

Anzahl Zeichen (‚maxlength'-Attribut): Mithilfe dieses Textfelds kann die Anzahl der Zeichen definiert werden, die maximal eingegeben werden darf.

Anzeigebreite: In diesem Feld kann die Breite des Formularfeldes in Pixel angegeben werden.

Anzeigehöhe: In diesem Feld kann die Höhe des Formularfeldes in Pixel angegeben werden.

Stylesheet-Klasse (‚class'-Attribut): In diesem Textfeld kann der Name einer CSS-Klasse eingegeben werden

4.3.6 Formular-Komponente „Hidden“

Mit dieser Formular-Komponente können Felder angelegt werden, die im Browser nicht angezeigt werden, aber im Servlet oder PHP-Skript ausgewertet werden können, d.h. zusätzliche nicht sichtbare Angaben. Auch hier muss der Redakteur einen eindeutigen Bezeichner für die Komponente vergeben.

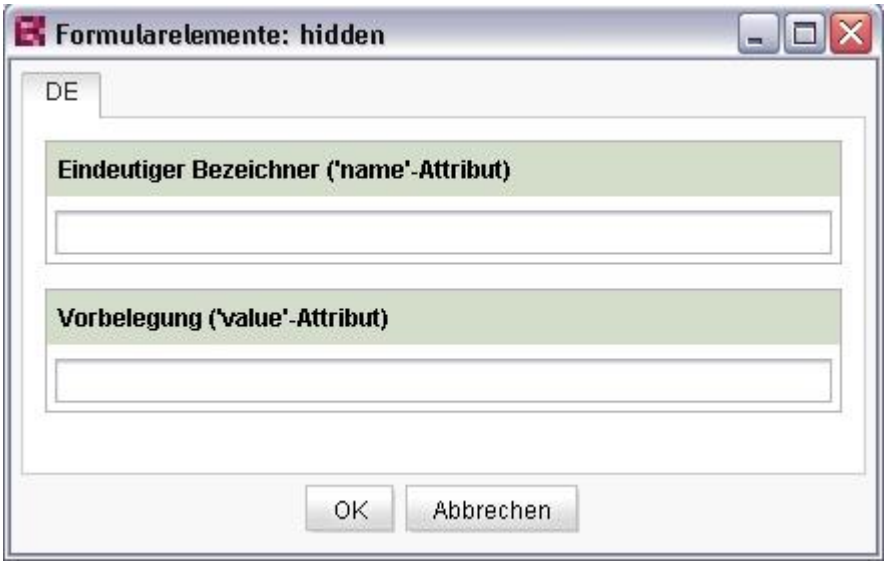

Abbildung 4-13: Formular-Komponente „Hidden“

Eindeutiger Bezeichner (‚name'-Attribut): In diesem Textfeld wird der eindeutige Bezeichner eingetragen.

Vorbelegung (‚value'-Attribut): In diesem Textfeld kann Text eingegeben werden, mit dem das Formularfeld gefüllt ist.

4.3.7 Formular-Komponente „Autocompleter"

Mit dieser Formular-Komponente kann dem Benutzer die Komfortabilität geboten werden, dass ihm während seiner Eingabe Vorschläge zur Vervollständigung gemacht werden. Auch hier muss der Redakteur einen eindeutigen Bezeichner für die Komponente vergeben.

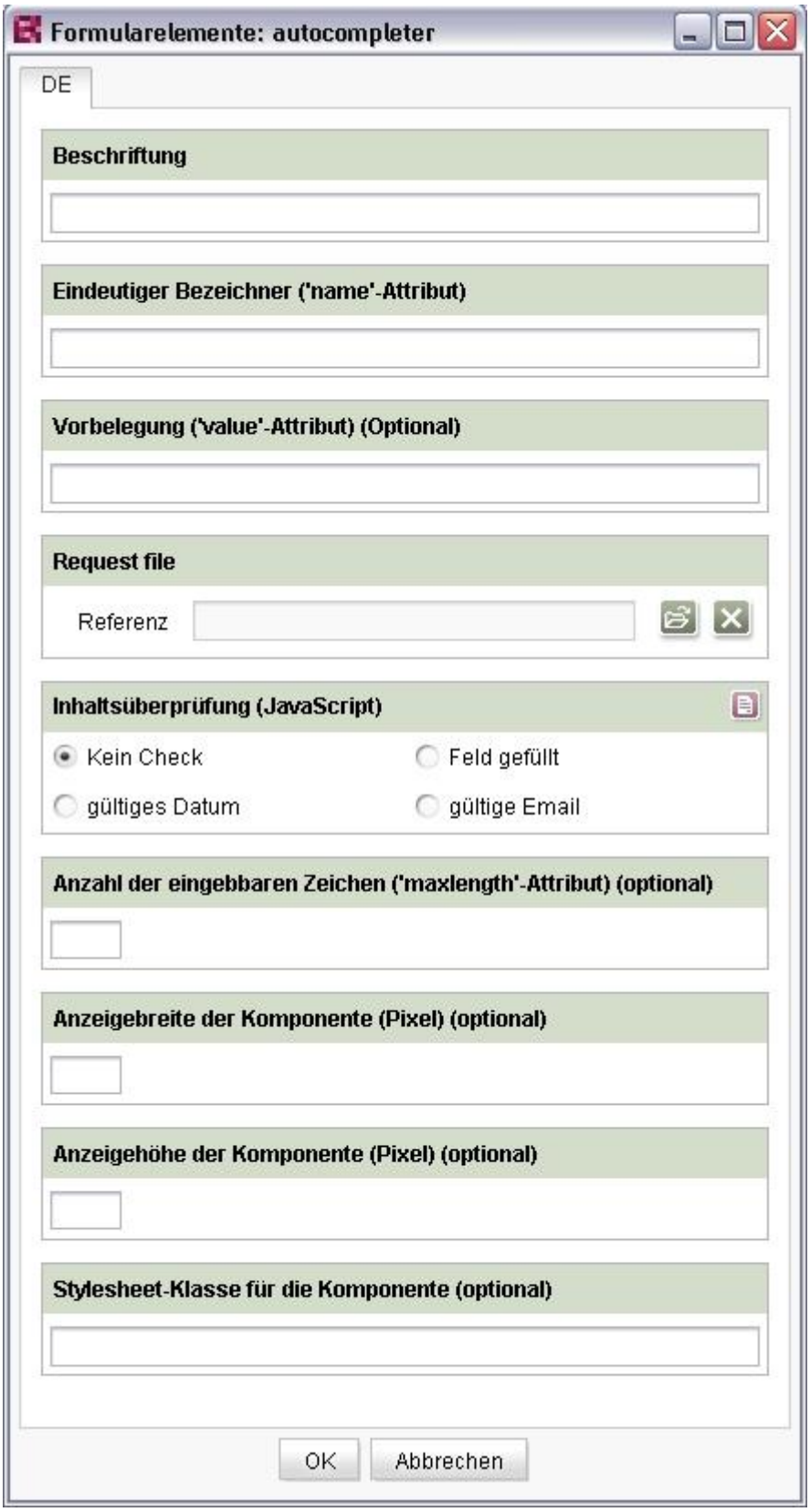

Abbildung 4-14: Formular-Komponente „Autocompleter“

Beschriftung: In diesem Textfeld wird der Titel des Formularfeldes eingegeben.

Eindeutiger Bezeichner (‚name'-Attribut): In diesem Textfeld wird der eindeutige Bezeichner eingetragen.

Vorbelegung (‚value'-Attribut): In diesem Textfeld kann Text eingegeben werden, mit dem das Formularfeld gefüllt ist.

Request-File: Über diese Komponente kann eine Seite aus der Struktur gewählt werden, die die Eingabe entgegennimmt und die zutreffenden Werte zurückgibt.

Inhaltsprüfung (gefüllt bzw. gültiger Wert): Über diese Radiobuttons kann eine Prüfung erstellt werden, die auf Inhalt bzw. auf gültige Daten prüft.

Anzahl Zeichen (‚maxlength'-Attribut): Mithilfe dieses Textfeldes kann die Anzahl der Zeichen definiert werden, die maximal eingegeben werden darf.

Anzeigebreite: In diesem Feld kann die Breite des Formularfeldes in Pixel angegeben werden.

Anzeigehöhe: In diesem Feld kann die Höhe des Formularfeldes in Pixel angegeben werden.

Stylesheet-Klasse (‚class'-Attribut): In diesem Textfeld kann der Name einer CSS-Klasse angegeben werden.

4.3.8 Formular-Komponente „combobox standard"

Die Formular-Komponente „combobox standard" bietet dem Redakteur eine komfortable Möglichkeit, dem Formular-Benutzer eine Auswahlliste zur Verfügung zu stellen. Die Inhalte der Liste sowie die Anzahl und das Layout sind frei formatierbar. Hier muss der Redakteur einen eindeutigen Gruppenbezeichner angeben.

Abbildung 4-15: Formular-Komponente „combobox standard“

Beschriftung: In diesem Textfeld wird der Titel des Formularfeldes eingegeben.

Gruppenbezeichner (‚name'-Attribut): In diesem Textfeld wird der eindeutige Bezeichner eingetragen.

sichtbare Einträge (‚size'-Attribut): In diesem Feld kann die Anzahl der sichtbaren Einträge als Zahl eingegeben werden.

Multiselect (‚multiple'-Attribut): Über diese Radiobuttons wird gesteuert, ob mehrere Einträge in diesem Formularfeld ausgewählt werden können

Inhaltsprüfung (gefüllt bzw. gültiger Wert): Über diese Radiobuttons kann eine Prüfung erstellt werden, die auf Inhalt bzw. auf gültige Daten prüft.

Options: In diese Liste können die Inhalte / Auswahlmöglichkeiten des Formularfeldes eingegeben werden.

Anzeigebreite: In diesem Feld kann die Breite des Formularfeldes in Pixel angegeben werden.

Anzeigehöhe: In diesem Feld kann die Höhe des Formularfeldes in Pixel angegeben werden.

Stylesheet-Klasse (‚class'-Attribut): In diesem Textfeld kann der Name einer CSS-Klasse eingegeben werden.

4.3.9 Formular-Komponente „combobox query"

Die Formular-Komponente „combobox query" bietet dem Redakteur eine komfortable Möglichkeit, dem Formular-Benutzer eine Auswahlliste zur Verfügung zu stellen. Die Inhalte der Liste sowie die Anzahl und das Layout sind frei formatierbar. Bei dieser speziellen SelectBox werden die Auswahloptionen aus der Datenbank über eine „Query" gefiltert und ausgegeben. Hier muss der Redakteur einen eindeutigen Gruppenbezeichner angeben.

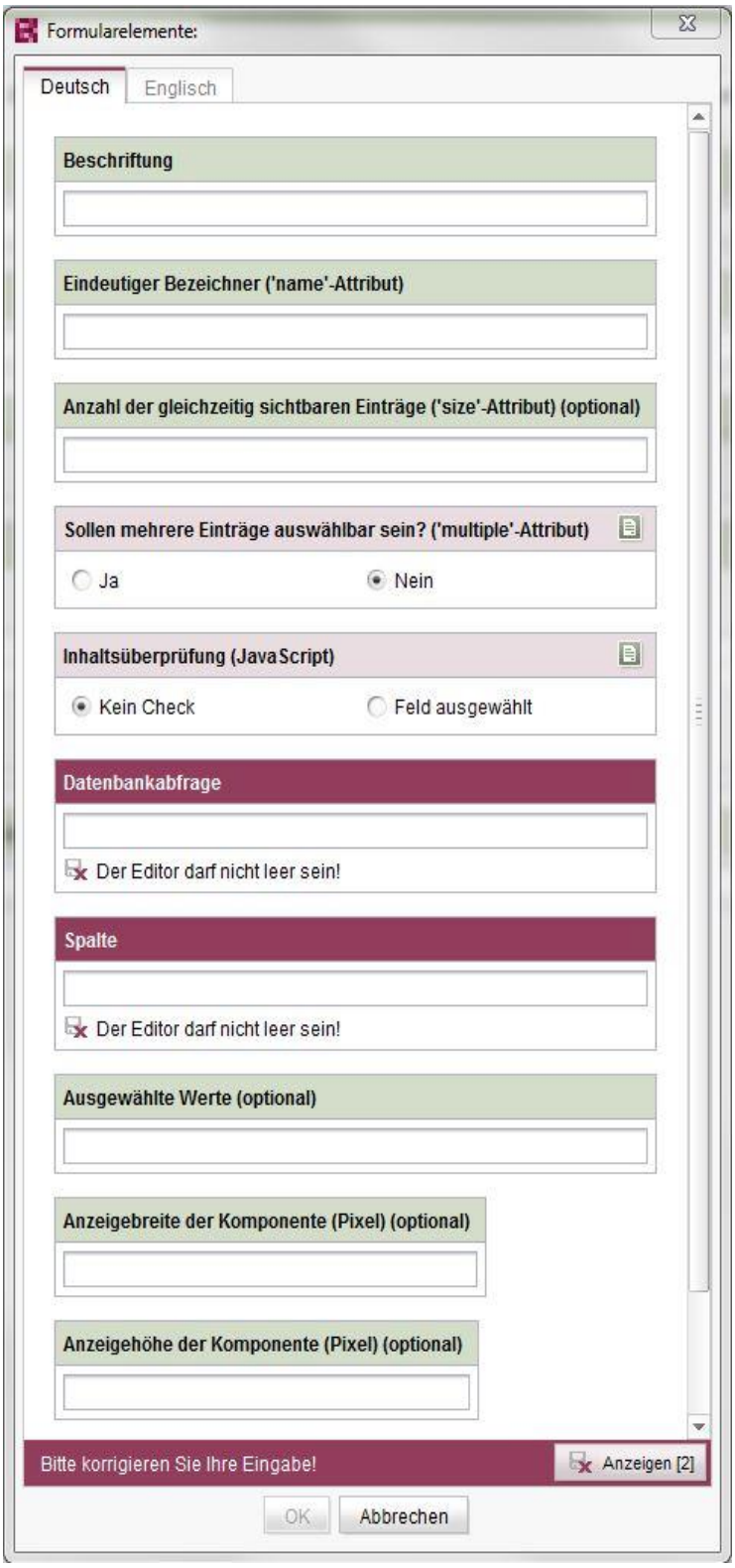

Abbildung 4-16: Formular-Komponente „combobox query“

Beschriftung: In diesem Textfeld wird der Titel des Formularfeldes eingegeben.

Gruppenbezeichner (‚name'-Attribut): In diesem Textfeld wird der eindeutige Bezeichner eingetragen.

sichtbare Einträge (‚size'-Attribut): In diesem Feld kann die Anzahl der sichtbaren Einträge als Zahl eingegeben werden.

Multiselect (‚multiple'-Attribut): Über diese Radiobuttons wird gesteuert, ob mehrere Einträge in diesem Formularfeld ausgewählt werden können

Inhaltsprüfung (gefüllt bzw. gültiger Wert): Über diese Radiobuttons kann eine Prüfung erstellt werden, die auf Inhalt bzw. auf gültige Daten prüft.

Datenbankabfrage: In dieses Textfeld kann der Name (UID) einer Datenbankabfrage (Query) eingegeben werden, die in der Vorlagen-Verwaltung vorhanden ist.

Spalte: In dieses Textfeld kann der Name einer Datenbankspalte bzw. eines Datenbankfeldes eingegeben werden. Die Werte dieser Spalte stehen in diesem Formularfeld zur Auswahl.

Ausgewählte(r) Wert(e) (‚selected'-Attribut): In diesem Textfeld können ein oder mehrere Werte angegeben werden, die in dem Formularfeld vorausgewählt sind. Bei mehreren Werten sind diese durch ein „," (Komma) voneinander zu trennen.

Anzeigebreite: In diesem Feld kann die Breite des Formularfeldes in Pixel angegeben werden.

Anzeigehöhe: In diesem Feld kann die Höhe des Formularfeldes in Pixel angegeben werden.

Stylesheet-Klasse (‚class'-Attribut): In diesem Textfeld kann der Name einer CSS-Klasse eingegeben werden.

4.3.10 Formular-Komponente „combobox date“

Die Formular-Komponente „combobox date“ bietet dem Redakteur eine komfortable Möglichkeit, dem Formular-Benutzer eine Auswahlliste zur Verfügung zu stellen. Die Inhalte der Liste sowie die Anzahl der Einträge und das Layout sind frei formatierbar. In dieser speziellen SelectBox werden automatisch drei Boxen erzeugt, über die komfortable ein Datum ausgewählt werden kann. Hier muss der Redakteur einen eindeutigen Gruppenbezeichner angeben.

Zur Umwandlung in ein richtiges Datumsformat innerhalb des Servlets wird dem eindeutigen Bezeichner ein „toDate_“! vorangestellt, um das Feld für das Servlet identifizierbar zu machen. Wurde als eindeutiger Bezeichner „datum“ eingeben, hat das Feld den Bezeichner „toDate_datum“ und muss auch so z.B. in der Mail-Vorlage verwendet werden.

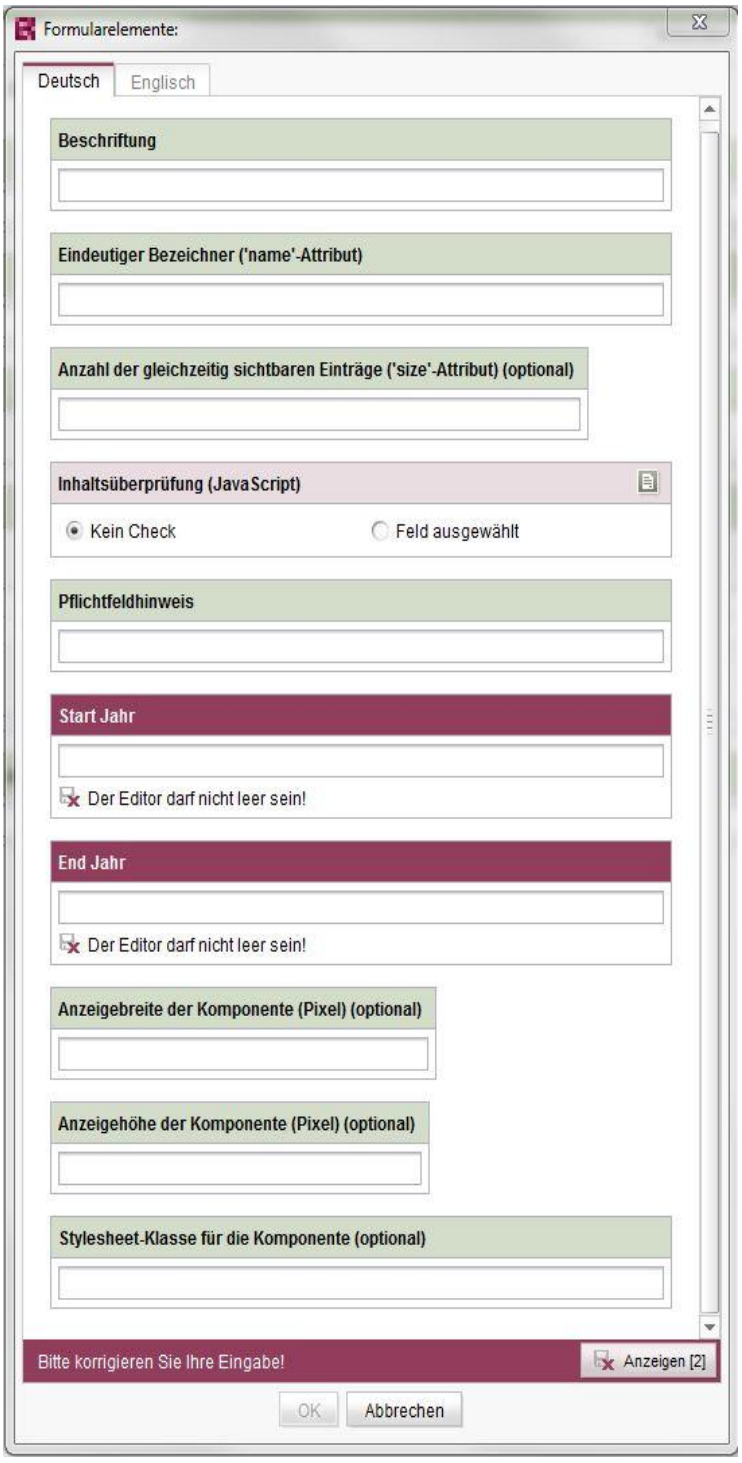

Abbildung 4-17: Formular-Komponente „combobox date“

Beschriftung: In diesem Textfeld wird der Titel des Formularfeldes eingegeben.

Gruppenbezeichner (‚name’-Attribut): In diesem Textfeld wird der eindeutige Bezeichner eingetragen.

sichtbare Einträge (‚size'-Attribut): In diesem Feld kann die Anzahl der sichtbaren Einträge als Zahl eingegeben werden.

Inhaltsprüfung (gefüllt bzw. gültiger Wert): Über diese Radiobuttons kann eine Prüfung erstellt werden, die auf Inhalt bzw. auf gültige Daten prüft.

Start Jahr: In diesem Textfeld kann eine Jahreszahl eingegeben werden. Von diesem Jahr bis zum Wert des Feldes „End Jahr" ist eine Datumsauswahl möglich.

End Jahr: In diesem Textfeld kann eine Jahreszahl eingegeben werden. Von dem Wert des Feldes „Start Jahr" bis zu diesem ist eine Datumsauswahl möglich.

Anzeigebreite: In diesem Feld kann die Breite des Formularfeldes in Pixel angegeben werden.

Anzeigehöhe: In diesem Feld kann die Höhe des Formularfeldes in Pixel angegeben werden.

Stylesheet-Klasse (‚class'-Attribut): In diesem Textfeld kann der Name einer CSS-Klasse eingegeben werden.

4.3.11 Formular-Komponente „fileupload“

Mit dieser Formular-Komponente ist es dem Benutzer möglich, Dateien über das Formular mitzusenden. Zudem ist es möglich, die Dateiformate einzuschränken. Auch hier muss der Redakteur einen eindeutigen Bezeichner für die Komponente vergeben.

Abbildung 4-18: Formular-Komponente „fileupload“

Beschriftung: In diesem Textfeld wird der Titel des Formularfeldes eingegeben.

Eindeutiger Bezeichner (‚name'-Attribut): In diesem Textfeld wird der eindeutige Bezeichner eingetragen.

Inhaltsprüfung (gefüllt bzw. gültiger Wert): Über diese Radiobuttons kann eine Prüfung erstellt werden, die auf Inhalt bzw. auf gültige Dateitypen prüft.

Akzeptierte Dateiformate: In diesem Textfeld kann eine kommaseparierte Liste von Datei-Endungen eingegeben werden. Eine Auswertung findet nur statt, wenn bei der Inhaltsprüfung „gültiger Dateityp" aktiviert ist. Mehrere Datei-Endungen sind durch ein „|" von einander zutrennen.

Anzeigebreite: In diesem Feld kann die Breite des Formularfeldes in Pixel angegeben werden.

Anzeigehöhe: In diesem Feld kann die Höhe des Formularfeldes in Pixel angegeben werden.

Stylesheet-Klasse (‚class'-Attribut): In diesem Textfeld kann der Name einer CSS-Klasse angegeben werden.

4.3.12 Formular-Komponente „captcha"

Mit dieser Formular-Komponente wird in dem Formular eine Captcha-Grafik, ein Link, der eine neue Grafik erzeugt, und ein Textfeld zur Eingabe des Captcha-Codes erzeugt. Diese Komponente soll dazu dienen, nicht menschlichen Anwendern die Nutzung des Formulars zu verweigern.

Aktivieren Sie im Absatz „form-start" die Checkbox „Captcha Validation", damit die Eingabe vom Servlet überprüft wird.

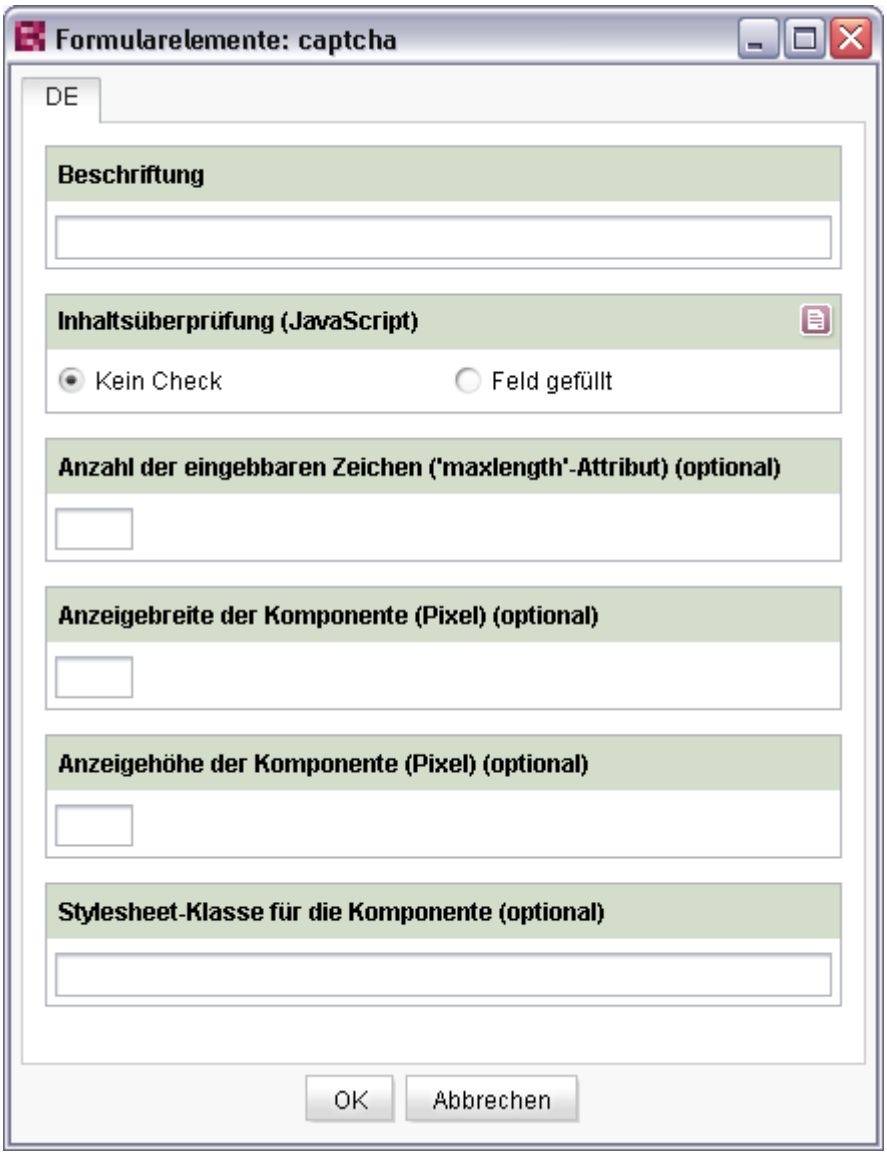

Abbildung 4-19: Formular-Komponente „captcha“

Beschriftung: In diesem Textfeld wird der Titel des Formularfeldes eingegeben.

Anzahl Zeichen (‚maxlength’-Attribut): Mithilfe dieses Textfeldes kann die Anzahl der Zeichen definiert werden, die maximal eingegeben werden darf.

Anzeigebreite: In diesem Feld kann die Breite des Formularfeldes in Pixeln angegeben werden.

Anzeigehöhe: In diesem Feld kann die Höhe des Formularfeldes in Pixeln angegeben werden.

Stylesheet-Klasse (‚class’-Attribut): In diesem Textfeld kann der Name einer CSS-Klasse angegeben werden.

Die Überlappung der Zeichen wird durch die Höhe, Breite und die Anzahl der zu rendernden Zeichen beeinflusst (siehe Kapitel 2.3.1 Seite 12). Es sollte immer darauf geachtet werden, dass die Zeichen nicht zu einfach lesbar sind, da diese sonst von Spam-Robotern gelesen werden können. Bei der Standardkonfiguration von 100 x 100 Pixel mit 6 Zeichen kann ein Captcha so aussehen:

Abbildung 4-20: Formular-Komponente „captcha“-Grafik

5 Benötigte Medien und deren Funktion

5.1 Stylesheet-Datei

Um die Formular-Komponenten schnell und einfach anzupassen, sind für nahezu jede Komponente Konfigurationsfelder für Stylesheet-Klassen vorgesehen. Durch Angabe einer Stylesheet-Klasse kann somit jedes Feld einen individuellen Style erhalten. Eine beispielhafte Stylesheet-Datei „formedit_css“ wird dem Projekt mit der Installation der Projektkomponente hinzugefügt. Die Klassennamen sollten in den Konfigurationsfeldern der Komponenten entsprechend angegeben werden. Die Anpassung der Stylesheet-Datei an individuelle Designvorgaben wird dringend empfohlen.

```
<script type="text/javascript"
src="$CMS_REF(media:"formedit_js")$"></script>
```

Um eigene Stylesheet-Klassen verwenden zu können, müssen diese entweder im HTML-Header der (Seiten-)Vorlagen mittels <style>-Tag deklariert werden, oder einer bestehenden bzw. neuen Stylesheet-Datei im Projekt hinzugefügt werden.

5.2 Javascript-Datei

Innerhalb der mitgelieferten Vorlagen werden Javascript-Funktionen verwendet, um z.B. das Ein- und Ausblenden von einzelnen Formularblöcken zu realisieren. Diese Funktionen sind in dem Medium „formedit_js“ hinterlegt. Das Medium muss in allen (Seiten-)Vorlagen, in denen Javascript-Funktionen benutzt werden können, innerhalb des HTML-Headers referenziert werden:

```
<script type="text/javascript"
src="$CMS_REF(media:"formedit_js")$"></script>
```

5.3 jQuery

Dieses freie Javascript-Framework stellt verschiedene Funktion zur Verfügung, die von den mitgelieferten Vorlagen verwendet werden. Außerdem ist diese Library Voraussetzung für die Nutzung der Formular-Komponente „Autocompleter“ (siehe Kapitel 5.5 Seite 68) und die Formularfeld-Validierung (siehe Kapitel 5.4 Seite 67). Die mitgelieferte Version ist kompatibel zu den mitgelieferten Plugins. Sollte ein Update notwendig sein, ist darauf zu achten, dass dieses zu den Plugins kompatibel

ist. jQuery kann parallel zu anderen Frameworks, z.B. MooTools, eingesetzt werden. Das Framework muss in allen Vorlagen, die die Funktionen von jQuery oder dessen Plugins nutzen, innerhalb des HTML-Headers referenziert werden:

```
<script type="text/javascript"
src="$CMS_REF(media:"jquery")$"></script>
```

Weitere Informationen und Updates: http://www.jquery.com

5.4 Formularvalidierung

Das jQuery-Plugin „validate“ bietet eine komfortable Möglichkeit, die verwendeten Formularfelder bei Eingabe auf inhaltliche Korrektheit zu überprüfen. In den Formular-Komponenten wird auf die Basisfunktionalitäten dieses Plugins zurückgegriffen. Mithilfe dieses Plugins können aber auch Abhängigkeiten zwischen Formularfeldern und weitere Einschränkungen geprüft werden. Um diese Funktionalität im Projekt nutzen zu können, muss die Datei „jquery_validate.js“ in allen (Seiten-)Vorlagen, in denen Formular-Komponenten benutzt werden können, innerhalb des HTML-Headers referenziert werden:

```
<script type="text/javascript"
src="$CMS_REF(media:"jquery_validate")$"></script>
```

Alternativ ist es möglich, die Funktion in eine bereits bestehende Javascript-Datei einzufügen.

Von diesem Plugin werden auch die (Fehler-)Meldungen, die bei fehlender oder falscher Eingabekomponente – inline – erscheinen, in 19 Sprachen mitgeliefert. Diese Fehlermeldungen sind in dem sprachabhängigen Medium „jquery_validate_messages“ hinterlegt und können bei Bedarf geändert und erweitert werden. Damit diese (Fehler-)Meldungen erscheinen, muss das Medium in allen (Seiten-)Vorlagen, in denen Formular-Komponenten benutzt werden können, innerhalb des HTML-Headers referenziert werden:

```
<script type="text/javascript"
src="$CMS_REF(media:"jquery_validate_messages")$"></script>
```

Weitere Informationen und Updates: http://plugins.jquery.com/project/validate

5.5 Autocompleter

Für die Komponente „Autocompleter“ wird ebenfalls ein jQuery-Plugin eingesetzt. Dieses stellt allerdings nur die Funktionalität bereit, eine Anfrage (Request) mit dem eingegebenen Zeichen an eine andere Seite abzusetzen und die Antwort (Response) dem Benutzer anzuzeigen. Die Datei „jquery_autocomplete“ muss in allen (Seiten-)Vorlagen, in denen die Formular-Komponente „Autocompleter“ benutzt werden kann, innerhalb des HTML-Headers referenziert werden:

```
<script type="text/javascript"
src="$CMS_REF(media:"jquery_autocomplete")$"></script>
```

Weitere Informationen zur vollständigen Integration finden Sie im Kapitel 6 Seite 69.

Weitere Informationen und Updates: http://plugins.jquery.com/project/autocompletex

6 Konzept „Autovervollständigung“

Die Formular-Komponente „Autocompleter“ stellt im eigentlichen Sinne nur eine ganz normale Texteingabe-Komponente zur Verfügung. Die Logik, die dieser Komponente die Funktionalität der Autovervollständigung bietet, besteht aus zwei Teilen:

- Der **erste Teil** ist das jQuery-Plugin „autocomplete“ (siehe Kapitel 5.5 Seite 68). Dieses Plugin ermöglicht es, die eingebenen Zeichen im Hintergrund an eine andere dynamische Seite oder ein Servlet zu senden. Weiterhin nimmt diese die Antwort wieder entgegen und blendet sie dem Benutzer ein.
- Der **zweite Teil** muss fallspezifisch implementiert werden: Dazu muss eine dynamische Seite (z.B. jsp oder php) erzeugt werden, die einen Request des Autocompleters entgegennimmt und dann damit eine Quelle (z.B. XML, Datenbank oder CSV) durchsucht. In einem weiteren Schritt müssen dann die gefundenen Daten wieder per Response an den Autocompleter zurückgegeben werden.

Dieses Konzept wird in den mitgelieferten Vorlagen anhand einer Beispiel-XML-Datei umgesetzt. Die Logik ist jsp-seitig implementiert und erfordert zusätzlich die Java-Bibliothek „jdom“.

Die Java Bibliothek „jdom“ ist kein Bestandteil des Moduls „FirstSpirit FormEdit“, sondern muss separat auf dem ApplicationServer bzw. FirstSpirit Server kopiert bzw. als Modul installiert werden. Weitere Informationen: http://www.jdom.org.

Beispiel:
Diesem Beispiel liegt folgende XML-Datei (Seitenvorlage „xmlDataBase“) zugrunde:

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

<jobs>

  <job jobnr="Projektleiter/in (jb-1742-a01)"
jobdescription="Projektleiter/in"/>

  <job jobnr="Azubi (jb-1743-a02)" jobdescription="Azubi"/>

  <job jobnr="Template-Entwickler/in (jb-1744-a03)"
jobdescription="Template-Entwickler/in"/>
```

```
  <job jobnr="Entwicklungsleiter/in (jb-1745-a04)"
jobdescription="Entwicklungsleiter/in"/>

</jobs>
```

Bei Verwendung des Formularfeldes „Autocompleter“ wird bei einer Eingabe nach z.B. 3 Zeichen eine Anfrage (Request) auf die in der Formular-Komponente referenzierten Seite abgesetzt. Diese Seite beruht auf der Seitenvorlage „autocompleterequest“. In dieser Seitenvorlage ist eine Logik implementiert, die diese Anfrage und die übermittelte Zeichenkette entgegennimmt.

```
incomingValue = (String) request.getParameter("q");
```

Innerhalb der referenzierten **XML-Quelle** wird in dem **Vergleichsattribut** nach Vorkommen der angefragten Zeichenkette gesucht.

```
for (Element child : children) {

   if (incomingValue == null ||

child.getAttribute(compareAttr).getValue().toLowerCase().contains(
incomingValue.toLowerCase())) {

buffer.append(child.getAttribute(resultAttr).getValue() + "\n");

}

}
```

Wenn ein zutreffender Eintrag gefunden wurde, wird eine Antwort (Response) an die Formular-Eingabekomponente gesendet:

```
Projektleiter/in
Entwicklungsleiter/in
```

Dem Formular-Anwender werden nun die gefundenen Daten angezeigt, hier z.B. der Wert des Attributs „jobdescription“ (**Rückgabetext**). Wählt der Anwender jetzt eine Möglichkeit aus, wird dieser als Wert des Formularfelds gespeichert.

7 Fallbeispiel: „Gewinnspiel“

In diesem Beispiel werden alle vom Redakteur durchzuführenden Aktionen beschrieben, die erforderlich sind, um ein Formular zu erstellen, mit dem sich ein Benutzer zu einem Gewinnspiel anmelden kann.

Die vom Benutzer eingegebenen Daten sollen zum einen einer personalisierten Teilnahmebestätigung dienen und zum anderen in einer Datenbank gespeichert werden, die alle Teilnehmer enthält.

Um dieses Beispiel durchführen zu können, muss das Modul „FirstSpirit FormEdit“ wie in Kapitel 2 ab Seite 7 beschrieben für ein FirstSpirit-Projekt installiert werden. Des Weiteren werden eine Datenbank und ein zur Verfügung stehender E-Mail-Server vorausgesetzt.

7.1 Erstellen des Formulars

Legen Sie eine neue Seite in der Inhalte-Verwaltung an und wählen Sie als Vorlage „formular“. Fügen Sie der Seite in einem Absatzbereich nacheinander die Absätze „form start“, „form block“ und „form end“ hinzu:

Abbildung 7-1: Seite mit Formular-Absätzen anlegen

Geben Sie nun im „form start“-Absatz eine Überschrift und einen Formularnamen an. Als Sendemethode wählen Sie „Post“ und aktivieren die clientseitige Inhaltsprüfung. Sollten Sie bereits eine Bestätigungsseite und eine Fehlerseite in Ihrer Struktur zur Verfügung haben, können Sie diese hier referenzieren.

Wechseln Sie nun in den Absatz „form block“ und legen Sie je ein Textfeld für Name, Vorname, Straße, Hausnummer, PLZ, Ort und E-Mail an. Verwenden Sie jeweils die Inhaltsprüfung „Feld gefüllt“. Um die Daten einfach in der Datenbank ablegen zu können, ist es ratsam, als eindeutigen Bezeichner den Spaltennamen in der Datenbank zu wählen. Für dieses Beispiel verwenden wir „nachname“, „vorname“, „strasse“, „hausnummer“, „plz“, „ort“ und „email“ als eindeutige Bezeichner:

Abbildung 7-2: Formularelemente

Im Absatz „form end“ sind alle erforderlichen Felder bereits vorausgefüllt, Sie können diese aber auch nach Ihren Vorstellungen anpassen.

7.2 Anlegen der Mail-Vorlage

Erstellen Sie in der Inhalte-Verwaltung eine neue Seite auf Basis der Vorlage „mailtemplate“. Als Empfänger verwenden wir das Feld aus dem Formular, da die E-Mail an den Formularanwender versendet werden soll. In diesem Beispiel also **%email%**.

Den Betreff können Sie ebenso wie den Nachrichtentext und die Absender-Adresse selbst definieren. Im Nachrichtentext können Sie mittels %eindeutigerBezeichner% auf alle Formularfelder zugreifen:

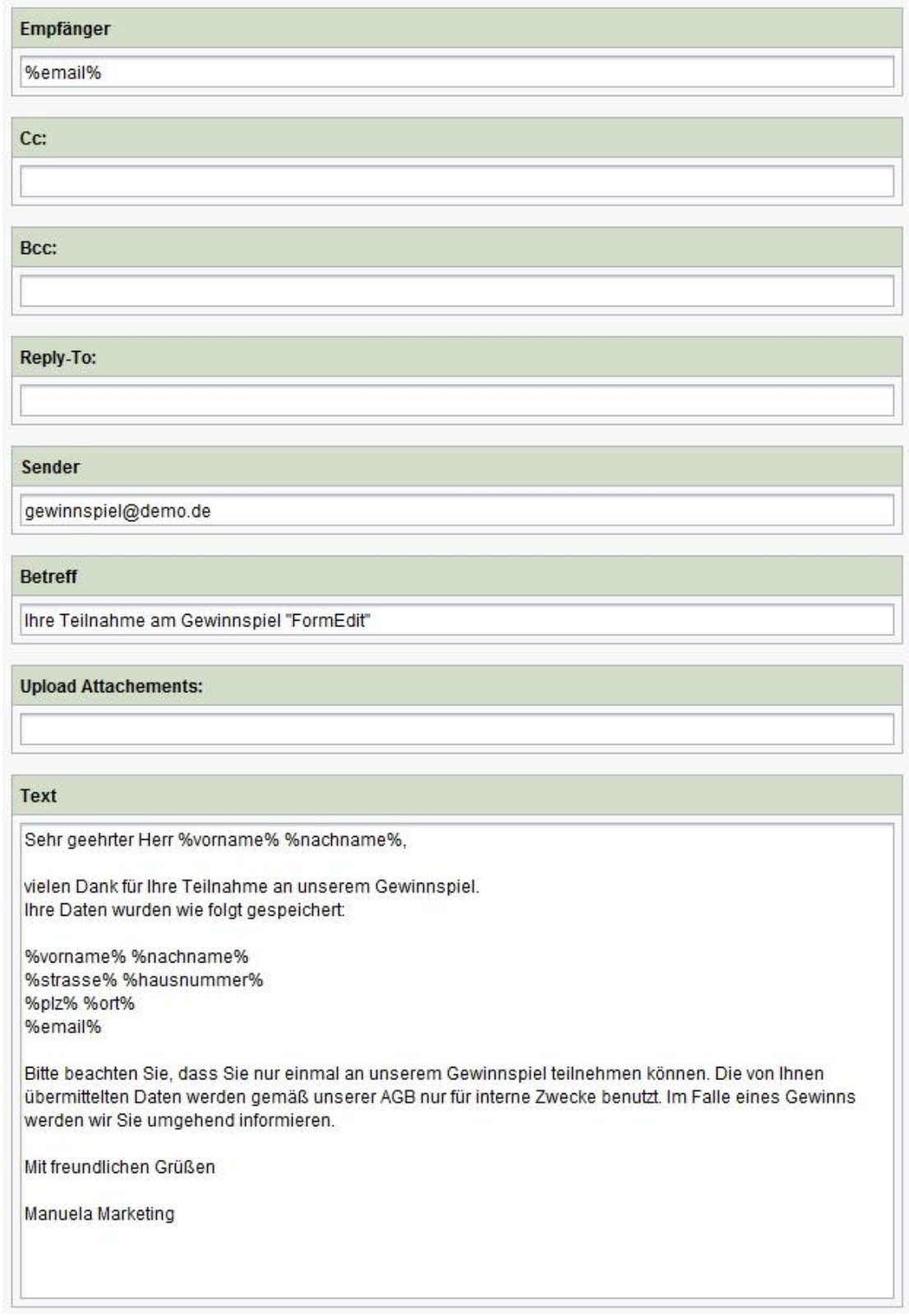

Abbildung 7-3: Mail-Vorlage

Ziehen Sie nun Ihre erstellte Mail-Vorlage per Drag&Drop in die Struktur-Verwaltung

oder referenzieren sie über das Kontextmenü aus der Struktur-Verwaltung.

7.3 Anlegen der Loggerkonfiguration (E-Mail)

Wechseln Sie nun wieder in Ihren „form start“ Absatz und fügen Sie in der Komponente „Weiterverarbeitung“ einen neuen Datensatz ein.

Abbildung 7-4: Datensatz hinzufügen

Im folgenden Dialog geben Sie einen Namen für die E-Mail-Weiterverarbeitung an und wählen den LoggerTyp „MailLogger“. Eine Beschreibung können Sie optional hinzufügen.

Fügen Sie nun in der Komponente „Logger Parameter“ einen neuen Datensatz mit einem Klick auf „Absatz hinzufügen“ hinzu und wählen Sie den Eintrag „logger-text-value“ und klicken Sie „OK“. Im folgenden Dialog geben Sie als Parameter-Name „**smtpHost**“ an und als Parameter-Wert die Adresse Ihres Postausgangs-Servers, z.B. smtp.mustermann.de. Selbiges wiederholen Sie für den Parameter „**encoding**“ mit dem Namen des von Ihnen gewünschten Encodings (hier: „UTF-8“).

Nun legen Sie noch einen neuen Absatz an, diesmal allerdings mit der Vorlage „logger-template-ref“. Als Parameter-Name geben Sie nun „**mailTemplatePath**“ ein und wählen in der Komponente „Mailtemplate“ die von Ihnen erstellte Mail-Vorlage. Weitere Parameter sind in Kapitel 3.2.4 Seite 22 erläutert.

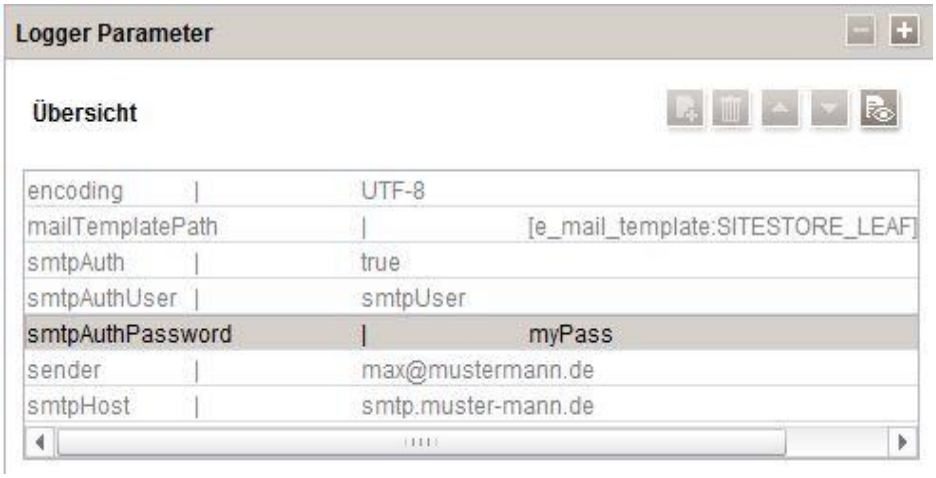

Abbildung 7-5: Parameter MailLogger

Abschließend wählen Sie in der Komponente den Status „Aktiviert“ (siehe Kapitel 3.1 Seite 15) und klicken in der linken oberen Ecke auf das „**Speichern**“-Symbol. Der

von Ihnen angelegte Logger wird nun automatisch in der Komponente „Weiterverarbeitung“ in Ihrem „form start“-Absatz aufgelistet.

7.4 Anlegen der Logger-Konfiguration (Datenbank)

Das Anlegen der zweiten Konfiguration erfolgt genauso wie der in Kapitel 7.3 Seite 74 beschrieben. Hier wählen Sie allerdings den LoggerTyp „**JdbcLogger**“ und legen alle Parameter auf Basis der „logger-text-value“ Vorlage an. Legen Sie bitte als erstes den Parameter „**driver**“ mit dem Namen Ihres Datenbanktreibers an, z.B. „com.mysql.jdbc.Driver“ für eine MySQL-Datenbank.

Bitte beachten Sie, dass der Datenbank-Treiber bereits dem FirstSpirit-Server hinzugefügt wurde.

Mit dem Parameter „**url**“ übergeben Sie die Adresse zu Ihrer Datenbank, z.B. „jdbc:mysql://muster:3306/formlogger“ und mit dem Parameter „**table**“ den Namen Ihrer Tabelle (hier: „demo“).

Nun legen Sie die Parameter „user“ und „password“ an und befüllen diese mit den Anmeldedaten für die Datenbank, die Ihnen üblicherweise Ihr Datenbank-Administrator zur Verfügung stellt.

Abbildung 7-6: Parameter JdbcLogger

Weitere Parameter sind in Kapitel 3.2.3 Seite 20 erläutert.

7.5 Anlegen der Konfigurationsdatei „fs-formlogger.ini“

Bitte lesen Sie hierzu das Kapitel 3.5 ab Seite 30.

7.6 Referenzieren und veröffentlichen

Abschließend referenzieren Sie alle erstellten Seiten in der Struktur und geben diese ggf. frei. Sollten Sie bzw. der Projektadministrator das Modul „FirstSpirit FormEdit“ für das Staging eingerichtet haben, sollte Ihr Formular nach der Generierung im generierten Stand (http://www.ihrredaktionsserver.de:8000/fs4staging_<project_id>/...) sichtbar sein und verwendet werden können.

8 Rechtliche Hinweise

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

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

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