FirstSpirit™
Your Content Integration Platform

FirstSpirit™ Security

FirstSpirit™ Version 4.x

Version	**1.2**
Status	**RELEASED**
Datum	**2011-04-14**
Abteilung	Techn. Documentation
Autor/ Autoren	B. Ehle
Copyright	2011 e-Spirit AG
Dateiname	SECU40DE_FirstSpirit_Modules_Security

e-Spirit AG

Barcelonaweg 14
44269 Dortmund | Germany

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

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

e-Spirit AG

Inhaltsverzeichnis

1 Einleitung

Bei einer Generierung werden Informationen – für jede Seitenreferenz in der Struktur-Verwaltung und für jedes Medium in der Medien-Verwaltung – in einer lokalen Datenbank hinterlegt, der sogenannten FirstSpirit™ Access-Control-Datenbank. Diese Datenbank dient zur Bereitstellungen von Informationen zu FirstSpirit™-Objekten, beispielsweise zur Bereitstellung von Zugriffsrechten, die für ein Objekt gespeichert wurden. Die Synchronisation der Access-Control-Datenbank mit dem aktuell freigegebenen Projektstand erfolgt automatisch bei der Generierung der Inhalte.

Für die Überprüfung von Zugriffsrechten im Live-System ist das Modul FirstSpirit™ Security eng mit der Personalisierung von Inhalten verknüpft (sogenannte "Benutzerrechte" – vgl. FirstSpirit™ Handbuch für Administratoren). Soll beispielsweise ein Zugriffsschutz für bestimmte Objekte im Live-Stand realisiert werden, ist neben dem FirstSpirit™ Modul Security, auch das (kostenpflichtige) Modul FirstSpirit™ PERSONALISATION erforderlich[1].

Neben den Informationen aus der Access-Control-Datenbank stellt das Modul FirstSpirit™ Security ein Servlet zur Veröffentlichung von FirstSpirit™-Inhalten zur Verfügung ("CRC-Transfer-Servlet"). Aufgabe des Servlets ist der Abgleich von generierten Projektdateien zwischen dem FirstSpirit™-Server und dem Live-System. Anhand der CRC- Prüfsummenberechnung können geänderte Dateien ermittelt und nur diese Dateien aktualisiert werden. Dieser differentielle Upload beschleunigt den Aktualisierungsvorgang im Live-System.

[1] Siehe Modul-Dokumentation FirstSpirit™ PERSONALISATION

1.1 Übersicht über die Funktionen

Das Modul FirstSpirit Security unterstützt die folgenden Aspekte:

- Bereitstellung von Informationen zu einzelnen FirstSpirit-Objekten, bspw.:
  - den vollständigen Pfad zum Objekt
  - die CRC-32-Prüfsumme des Objektes
  - das Datum der letzten Änderung des Objektes
  - das Datum der letzten Aktualisierung der gespeicherten Daten eines Objektes
  - den eindeutigen Bezeichner des Objektes
  - die ID des Objektes

  Der Zugriff auf diese Informationen kann über ein Skript oder eine JSP-Seite erfolgen.
- Bereitstellung von Zugriffsrechten auf Objektebene:
  Berechtigungsinformationen (Benutzerrechte bzw. Zugriffsrechte) für den Besucher der generierten Site (sogenannte "Benutzerrechte") können in FirstSpirit auf vielen Knoten im Projekt hinterlegt werden und sind Teil der Meta-Informationen. Die Pflege der Benutzerrechte kann mithilfe der Rechtedefinitions-Komponente vorgenommen werden[2]. Der Zugriff auf die Informationen kann über ein Skript oder eine JSP-Seite erfolgen.
- Abgleich der Dateien bei der Veröffentlichung (über ein FirstSpirit-Servlet): Wird für die Veröffentlichung ein FirstSpirit-Servlet verwendet, so können die bereitgestellten Informationen zusätzlich für den Abgleich der Dateien verwendet werden (auf Basis der Access-Control-Datenbank). Bei der Übertragung werden nur veränderte, neue und gelöschte Dateien berücksichtigt. Unveränderte Dateien werden nicht erneut übertragen.

[2] Zu Benutzerrechten siehe Handbuch für Redakteure JAVAclient Kapitel 13

1.2 Thema dieser Dokumentation

Kapitel 1: Eine kurze Einführung in Konzept und Funktionsumfang der FirstSpirit Access-Control-Datenbank (ab Seite 3).

Kapitel 2: Beschreibt die Installation des Moduls "FirstSpirit Security" auf dem Server und die Installation der Webkomponente in einem Projekt (ab Seite 7).

Kapitel 3: In diesem Kapitel wird die Konfiguration des Moduls "FirstSpirit Security" und der Webkomponente erläutert (ab Seite 10).

Kapitel 4: Eine ausführliche Beschreibung aller Klassen und Methoden, die Zugriff auf die Access-Control-Datenbank bieten (ab Seite 23).

Kapitel 5: Beschreibt De-/Registrierung sowie die Implementierung eines Listeners über das CRC-Transfer-Servlet (ab Seite 76).

1.3 Begriffe und Konzepte

1.3.1 Access-Control-Datenbank und Zugriffsrechte

Der Abgleich der Daten, zwischen der lokalen Access-Control-Datenbank und der Access-Control-Datenbank in der Webanwendung, erfolgt auf Basis der gespeicherten CRC-32-Prüfsumme und dem Pfad zu einer generierten Datei.

Eine Access-Control-Datenbank besteht aus mehreren "Access Control Lists" (ACL). Diese "Zugriffskontrolllisten" können aus mehreren Dateien bestehen. Eine Datei in einer ACL entspricht einer bei der Generierung erzeugten Datei.

Der zentrale Bestandteil einer ACL sind die Benutzerrechte. Dateien mit denselben Benutzerrechten werden in einer ACL zusammengefasst.

Benutzerrechte können in FirstSpirit auf jedem Knoten im Projekt hinterlegt werden und sind Teil der Meta-Informationen. Die Pflege der Berechtigungsinformationen kann mit Hilfe der Rechtedefinitions-Komponente vorgenommen werden.

Weitere Informationen zur Rechtedefinitions-Komponente können dem "FirstSpirit Handbuch für Administratoren" und "FirstSpirit Handbuch für Redakteure (JavaClient)" entnommen werden.

1.3.2 Servlet zur Veröffentlichung ("CRC-Transfer-Servlet")

Aufgabe des "FirstSpirit-Servlets zur Veröffentlichung" ist das Abgleichen der generierten Projektdateien zwischen dem FirstSpirit-Server und dem Live-System. Hierbei wird eine CRC-Prüfsummenberechnung der vorhanden und der neu generierten Dateien durchgeführt, um anhand der Prüfsumme entscheiden zu können, welche der Dateien verändert wurden (neu, geändert oder gelöscht). (Der Vergleich des Erstellungsdatums der Datei ist aufgrund von asynchronen Uhren in den verschiedenen Servern nicht ausreichend.) Werden bei dieser Prüfsummen-Berechnung Unterschiede festgestellt, werden anschließend nur die neuen und geänderten Datei hochgeladen bzw. veraltete Dateien gelöscht (falls keine neue mit gleichem Namen vorhanden ist).

Der differentielle Upload beschleunigt damit nicht die Generierungszeit auf dem FirstSpirit Server, sondern nur den Aktualisierungsvorgang auf dem Live-System. Da dieser Aktualisierungsvorgang speziell in Netzwerken mit einer geringen Bandbreite und bei einer umfangreichen Site einige Zeit in Anspruch nehmen kann, ist durch den Einsatz des FirstSpirit-Servlets eine Zeitreduktion für den gesamten Veröffentlichungsprozess möglich[3].

Das Servlet wird über das Modul FirstSpirit Security zur Verfügung gestellt und kann über den Konfigurationsdialog der zugehörigen Webanwendung "FS Security WebApp" projektspezifisch angepasst werden (siehe Kapitel 3.1.3.1 Seite 13).

Zudem stehen einige Klassen bzw. Interfaces zum Registrieren und Deregistrieren von Listenern zur Verfügung (siehe Kapitel 5 Seite 76).

[3] Zur Konfiguration einer Veröffentlichung über ein FirstSpirit™ Servlet siehe Handbuch für Administratoren

2 Installation

2.1.1 Installieren des Moduls auf dem Server

Das Modul FirstSpirit Security muss zunächst innerhalb der Anwendung zur Server- und Projektkonfiguration installiert werden. Im Bereich Servereigenschaften 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. Die erfolgreich installierte Datei wird anschließend im Dialog "Server Eigenschaften" angezeigt:

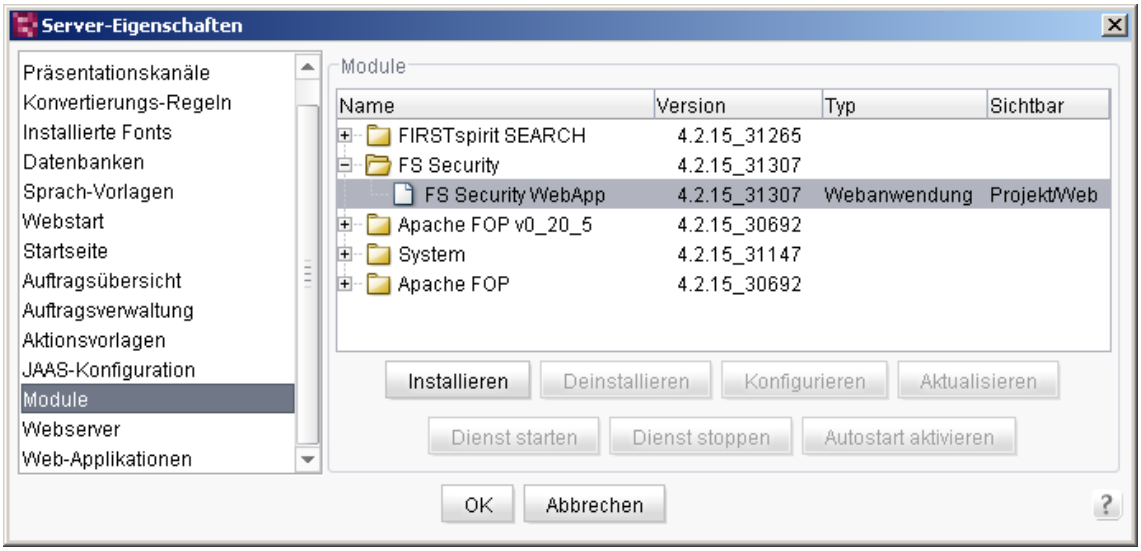

Abbildung 2-1: Installation des Moduls auf dem FirstSpirit-Server

Bestandteil des Moduls FirstSpirit Security ist die Webanwendung "FS Security WebApp". Die Webanwendung stellt das FirstSpirit Veröffentlichungs-Servlet und das Access-Control-Servlet zur Verfügung.

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) innerhalb der gewünschten Projekte zugefügt werden (siehe Kapitel 2.1.2 Seite 8).

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

2.1.2 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 2-2: Installation der Webanwendung innerhalb der Web-Bereiche

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

Abbildung 2-3: 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 von WebEdit

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 Seite 7).

Nach dem Hinzufügen zu einem Webbereich, besteht die Möglichkeit, die Komponente zu konfigurieren, entweder mit einer von der Komponente erzeugten oder einer generischen GUI (siehe Kapitel 3.1.4 Seite 15). Nach der Konfiguration müssen die Komponenten noch aktiviert werden. Dabei kann eine Komponente innerhalb eines Projekts nur für bestimmte Bereiche aktiviert bzw. deaktiviert werden

Installieren: Mit einem Klick auf den Button wird die Web-Komponente im jeweiligen Web-Bereich des Projekts in einer War-Datei zusammengefasst und abhängig vom konfigurierten Webserver installiert. Der Button ist aktiviert, wenn die Web-Komponente noch nicht installiert wurde und zur Installation bereit ist. Ist der Button deaktiviert, wurden die Web-Komponenten bereits installiert.
Handelt es sich beim ausgewählten Webserver, um einen externen Webserver oder um einen generischen Webserver (ohne die erforderliche Skript-Funktionalität), wird stattdessen der **Button "Download"** angezeigt.
Wurde die Web-Anwendung bereits installiert, anschließend aber die Konfiguration geändert, wird statt des Buttons "Installieren", der **Button "Aktualisieren"** angezeigt (siehe Abbildung 3-4).

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

2.1.3 Systemvoraussetzungen

Das Servlets zur Veröffentlichung ist ein vollständig in Java entwickeltes Servlet und benötigt lediglich einen Servlet-Container (Empfehlungen dazu befinden sich im "Technischen Datenblatt" von FirstSpirit).

Hinweis: Die beschriebenen Servlets und Servlet-Filter sind nicht für den Einsatz in einer Cluster-Umgebung geeignet.

3 Konfiguration

Die lokale Access-Control-Datenbank eines Auftrags wird bei der ersten Generierung angelegt und bei jeder weiteren Generierung aktualisiert.

Somit sind die Daten in einem Skript innerhalb eines Auftrages ohne weitere Konfiguration verfügbar und nutzbar (Zur Verwendung von Skripten in Aufträgen siehe "FirstSpirit Handbuch für Administratoren".).

Weitere Konfigurationsschritte sind nur notwendig, wenn:

- die Informationen der (lokalen) Access-Control-Datenbank als Grundlage für die Aktualisierung, das Hinzufügen und das Löschen von Dateien des Webserver dienen sollen (siehe Kapitel 3.1.1 Seite 10).
- die Informationen der lokalen Access-Control-Datenbank auf einen (externen) Webserver übertragen werden sollen (Kapitel 3.1.2 Seite 11).

3.1.1 Abgleich der Dateien über die Access-Control-Datenbank

Basis für den Dateiabgleich beim Aktualisieren, Löschen und Hinzufügen von Dateien sind die CRC-32-Prüfsumme und der Pfad zu einer Datei. Der in der lokalen Access-Control-Datenbank gespeicherte Pfad zu einer Datei ist immer absolut bezogen auf das Generierungsverzeichnis eines Auftrages.

Beispiel: In der Sprache Deutsch (Kürzel: "DE") wird für den ersten Präsentationskanal die Seitenreferenz "index.html" generiert.
Der gespeicherte Pfad lautet:

```
/de/index.html
```

Für den Abgleich der Dateien, ist in einem Auftrag die Veröffentlichung über das "FirstSpirit Veröffentlichungs-Servlet" zu wählen. Dazu muss zunächst in der Auftragsverwaltung des Projekts ein neuer Auftrag hinzugefügt werden. Dem Auftrag wird eine Aktion vom Typ "Veröffentlichung über ein FirstSpirit Veröffentlichungs-Servlet" hinzugefügt (siehe Dokumentation "Handbuch für Administratoren").

Zusätzlich sind Anpassungen im Webserver vorzunehmen. Die Konfiguration der Datei crcTransfer.ini erfolgt projektspezifisch über den Konfigurationsdialog der Webanwendung FS Security WebApp (siehe Kapitel 3.1.4.1 Seite 16). Nach der Konfiguration sollte die Webanwendung auf dem Webserver installiert (siehe Kapitel Seite) und aktualisiert werden (siehe Kapitel 3.1.4 Seite 15).

3.1.2 Informationen zu externem Webserver übertragen

Die Veröffentlichung über ein FirstSpirit-Servlet erfolgt häufig in einem Unterverzeichnis der Webanwendung. Der Pfad zum Wurzelverzeichnis der Webanwendung ist daher unterschiedlich zur Generierung.

Bei der Generierung eines Projekts wird daher eine lokale Access-Control-Datenbank mit allen Dateien erzeugt. Bei der Veröffentlichung wird diese lokale Access-Control-Datenbank mit der Remote-Access-Control-Datenbank abgeglichen. Die Remote-Access-Control-Datenbank kann beliebig viele Projekte verwalten. Damit die einzelnen Dateien in der Remote-Access-Control-Datenbank noch eindeutig zugeordnet werden können, wird der absolute Pfad um ein projektweit freiwählbares Präfix erweitert. Das angegebene Präfix ergänzt die Pfadangabe eines Objektes innerhalb der ACL-Datenbank.

Das Präfix kann in den Projekteinstellungen im Feld "Präfix für die Access-Control-Datenbank" im Unterpunkt "Berechtigungen" definiert werden:

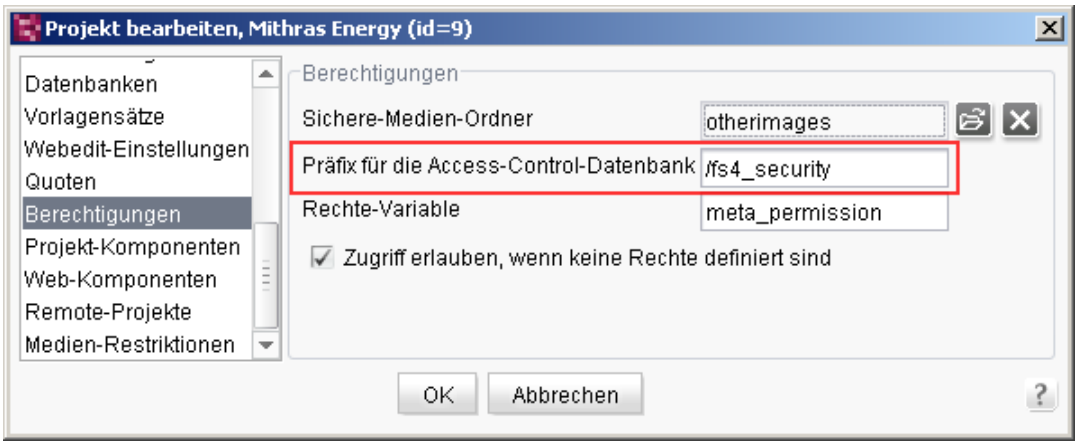

Abbildung 3-1: Eingabe des Präfixes für den Dateiabgleich

Präfix für die Access-Control-Datenbank: Angabe eines Präfixes zur Vervollständigung der ACL-Datenbank-Informationen einer Datei. Der vollständige Pfad zu einer Datei in FirstSpirit besteht immer aus drei Teilen:

- URL der Webanwendung (z.B. http://meinServer.de)
- Präfix für die Access-Control-Datenbank (/fs4_security)
- Pfad zu einer Datei (/de/index.html)

Der vollständige Pfad innerhalb der ACL-Datenbank setzt sich aus dem Präfix und dem Pfad zur Datei zusammen, z.B. `/fs4_security/de/index.html`

In Abbildung 3-1 lautet das Präfix "/fs4_security" und entspricht dem direkten Unterverzeichnis "fs4_security" der Webanwendung, beispielsweise "live":

"~Webserver/webapps/live/fs4_security".

Die Angabe des Präfixes ist absolut zur Webanwendung. Zusätzlich entspricht das Präfix in diesem Fall dem letzten Teil des Feldwertes "Pfad auf Live-Server" im Dialog des Veröffentlichungs-Servlets (siehe Abbildung 3-2).

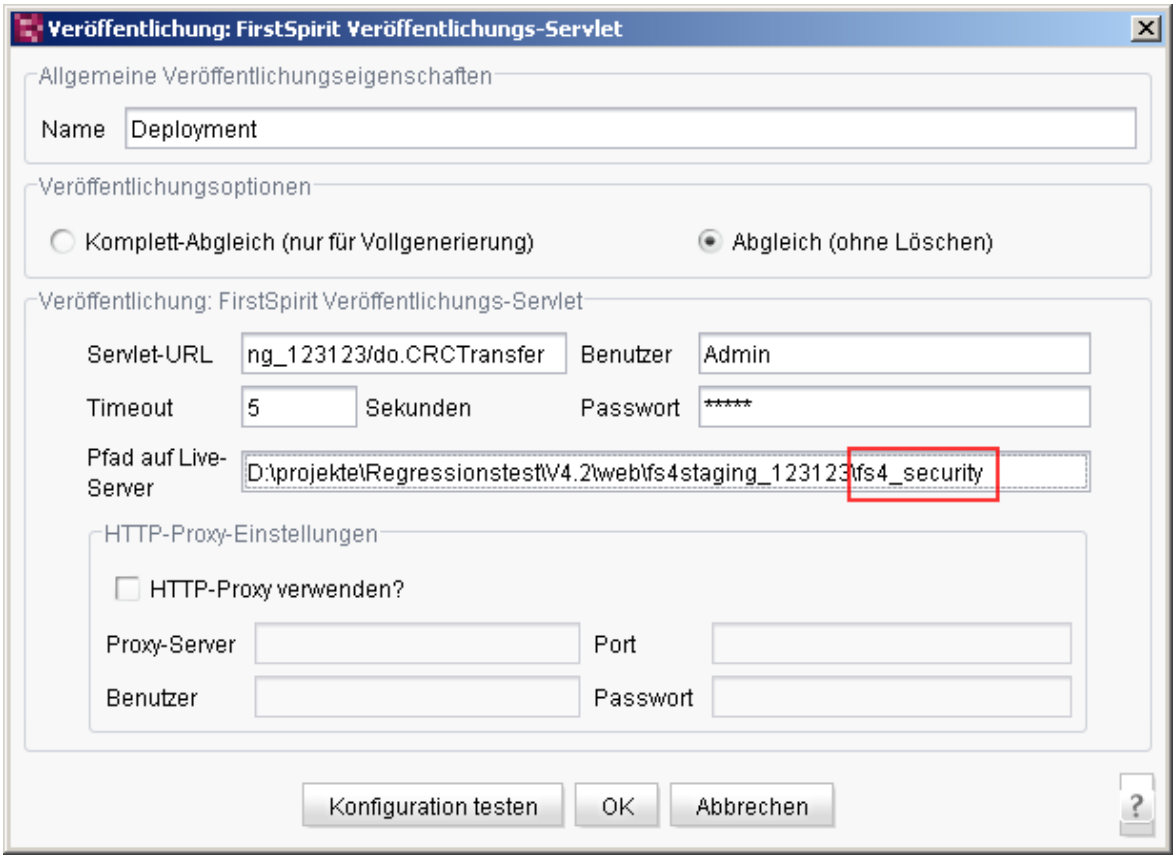

Abbildung 3-2: Dialog FirstSpirit Veröffentlichungs-Servlet

Der Pfad der Datei "index.html" aus dem Beispiel in der Webanwendung lautet:

```
/fs4_security/de/index.html
```

Wird das Präfix geändert, so bleiben die Einträge mit dem ursprünglichen Präfix in der Access-Control-Datenbank in der Webanwendung erhalten.

Ein weiterer wichtiger Punkt ist die Einstellung "Abgleich (ohne Löschen)" im Dialog des Veröffentlichungs-Servlet. Ist diese Option aktiviert, werden die Einträge zu Dateien, die im Projekt gelöscht wurden, nicht aus der Access-Control-Datenbank entfernt.

3.1.3 Konfiguration des Moduls "FirstSpirit Security"

Das Modul FirstSpirit Security umfasst die folgenden Teilbereiche:

- das FirstSpirit Veröffentlichungs-Servlet (siehe Kapitel 3.1.3.1 Seite 13)
- Abgleich der Access-Control-Datenbank (siehe Kapitel 3.1.3.2 Seite 14)

3.1.3.1 FirstSpirit Veröffentlichungs-Servlet

Das Veröffentlichungs-Servlet ist ein fester Bestandteil der Webanwendung im Webserver. Durch dieses Servlet ist es möglich, bei einer Veröffentlichung die lokale Dateiliste mit der Webanwendung abzugleichen.

Durch die CRC-32-Prüfsummenberechnung für jede generierte Datei können:

- nur geänderte und neue Dateien übertragen werden.
- nicht mehr erzeugte Dateien in der Webanwendung gelöscht werden (optional).

So wird sichergestellt, dass alle aktuellen Daten erzeugt, aber nur veränderte Daten auch übertragen werden.

Der Dateiabgleich erfolgt auf Basis der berechneten CRC-32-Prüfsummen. Diese Prüfsumme ist abhängig vom Datei-Inhalt und der Dateigröße. Daher können Informationen, die sich bei jeder Generierung ändern, z.B. die Verwendung des Generierungsstartzeitpunkts ("#global.startDate") innerhalb eines Meta-Tags für Suchmaschinen, sich nachteilig auf die Veröffentlichung auswirken.

Um das FirstSpirit-Servlet zur Veröffentlichung zu verwenden, muss auf dem Zielsystem eine Servlet-Engine installiert sein, die vollen Zugriff auf das Dateiverzeichnis hat, in dem sich die Webseite befindet.

3.1.3.2 Access-Control-Servlet

Über das Access-Control-Servlet wird die Access-Control-Datenbank bei einer Veröffentlichung aktualisiert. Dabei wird der Stand der lokalen Access-Control-Datenbank (über das Servlet) mit dem Stand der Access-Control-Datenbank in der Webanwendung abgeglichen.

Voraussetzung für die Nutzung des Access-Control-Servlets ist die Verwendung des FirstSpirit Veröffentlichungs-Servlets.

Für die Verwendung des Servlets genügt die Installation des Moduls (siehe 2.1.1 Seite 7). Alternativ kann das Servlet aber auch manuell in die Datei "web.xml" der Webanwendung eingefügt werden:

```
<servlet>
  <servlet-name>
    fss-AccessControlServlet
  </servlet-name>

  <servlet-class>
    de.espirit.firstspirit.acl.servlet.AccessControlServlet
  </servlet-class>

  <load-on-startup>
    1
  </load-on-startup>
</servlet>
```

3.1.4 Konfigurieren der Webanwendung

Mit einem Klick auf den Button öffnet sich der Konfigurationsdialog der Webanwendung (vgl. Abbildung 2-2). Im Konfigurationsdialog werden die Inhalte der beiden Dateien "crcTransfer.ini" und "aclFilter.conf" angezeigt (vgl. Abbildung 3-3):

- Die Datei "crcTransfer.ini" enthält die Konfiguration für das FirstSpirit Veröffentlichungs-Servlet (siehe Kapitel 3.1.4.1 Seite 16).
- Die Datei "aclFilter.conf" enthält die Konfiguration für den Multi-Access-Control-Filter. Über diese Datei können unterschiedliche Filter für Projektinhalte definiert werden für die, innerhalb der generierten bzw. veröffentlichten Inhalte, ein spezieller Zugriffsschutz gewünscht ist (siehe Kapitel 3.1.4.2 Seite 18).

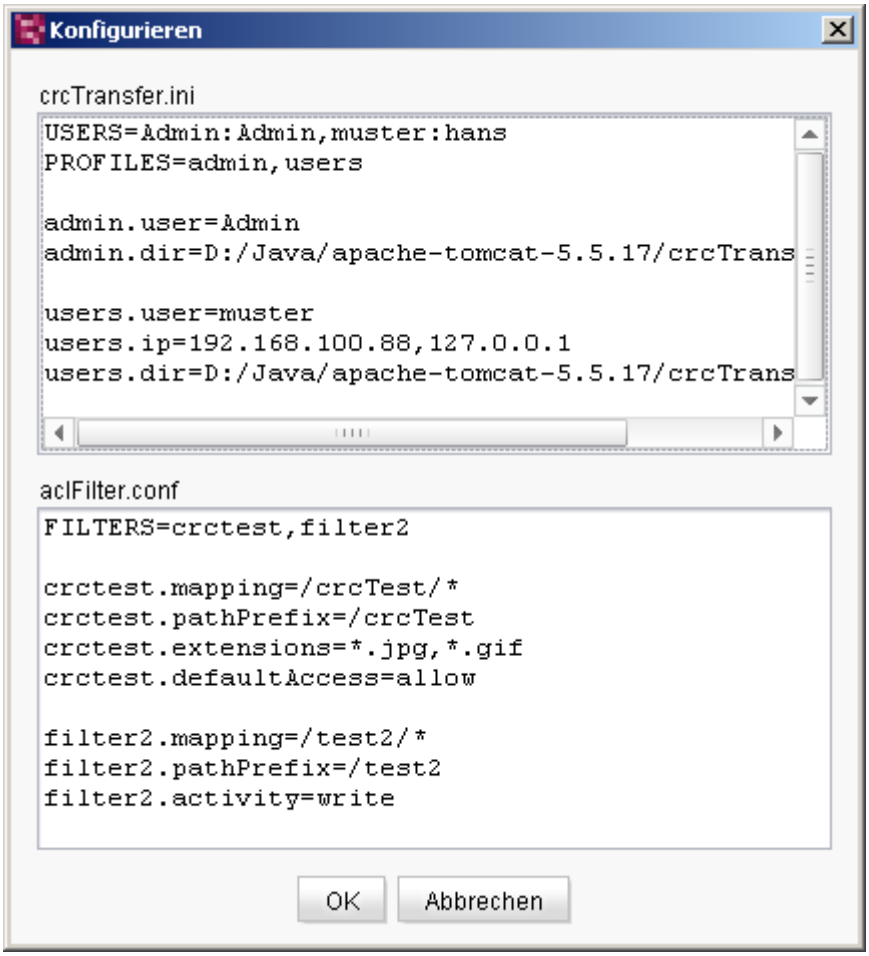

Abbildung 3-3: Konfigurieren der Webanwendung Security

Nach Änderung der Konfiguration muss die Konfigurationsdatei auf dem Webserver veröffentlicht werden. Dazu muss im Bereich Web-Komponenten die Webanwendung "FS Security WebApp" aktualisiert werden:

Abbildung 3-4: Aktualisieren der Webanwendung nach der Konfiguration

3.1.4.1 crcTransfer.ini

Die Konfigurationsdatei crcTransfer.ini dient zur Konfiguration des FirstSpirit-Veröffentlichungs-Servlets (siehe Kapitel 3.1.3.1 Seite 13).

crcTransfer.ini

```
USERS=Admin:Admin,muster:hans
PROFILES=admin,users

admin.user=admin
admin.dir=D:/Java/apache-tomcat-5.5.17/crcTransfer/crcTest

users.user=muster
users.ip=192.168.100.88,127.0.0.1
users.dir=D:/Java/apache-tomcat-5.5.17/crcTransfer/crcTest
```

Abbildung 3-5: Konfiguration crcTransfer.ini

Dort kann das Zielverzeichnis für die Veröffentlichung (DocumentRoot des Webservers) sowie die Anmeldedaten (Benutzername und Passwort) konfiguriert werden.

Dazu stehen die folgenden Parameter zur Verfügung:

Parameter "USERS": Hier werden die Benutzerkonten eingetragen, die in der Server- und Projektkonfiguration im Auftrag zur Veröffentlichung verwendet werden können. Mehrere Einträge erfolgen als kommaseparierte Liste, wobei Benutzername und Passwort eines Benutzers jeweils über einen : getrennt werden:
Benutzername:Passwort,Benutzername2:Passwort2

Parameter "PROFILES": Hier werden kommasepariert Profilnamen eingetragen, die aktiviert werden sollen. Die Profile selbst werden anschließend durch ihre Parameter "profilname.*" definiert.

Parameter "profilname.user": Hier werden kommasepariert die Benutzernamen (siehe USERS) eingetragen, für die dieses Profil gelten soll.

Parameter "profilname.ip": Falls hier kommaseparierte IP-Adressen angegeben werden, ist nur von diesen IP-Adressen aus die Nutzung dieses Profils möglich.

Parameter "profilname.dir": Hier werden kommasepariert absolute Pfade (voll qualifizierter Pfad) angegeben, in die das Servlet schreiben darf. Um beispielsweise eine Veröffentlichung für die Webapplikation "fs4staging_1921116" zu konfigurieren, sollte folgender Pfad eingetragen werden, damit das Verzeichnis `/WEB-INF` der Webapplikation vom Veröffentlichungsauftrag nicht entfernt wird: `D:/Tomcat/webapps/fs4staging_1921116/fs4_security`

 *Bei Angabe von Pfaden in Windows-Dateisystemen ist darauf zu achten, dass keine Backslashs, sondern Slashs verwendet werden:* `D:/Tomcat/webapps/fs4staging_1921116/fs4_security`

 *Die hier angegebenen Verzeichnisse werden bei einer Veröffentlichung vollständig gelöscht!*

Nach der Konfiguration der `crcTransfer.ini` sollte die Webapplikation gestartet werden.

Die Angaben zum Servlet befinden sich anschließend in der Datei web.xml der Webanwendung:

```
<servlet-mapping>
        <servlet-name>fss-CRCTransferServlet</servlet-name>
        <url-pattern>*.CRCTransfer</url-pattern>
</servlet-mapping>
```

Zum Testen der Erreichbarkeit des Servlets, kann das Servlet im Webbrowser aufgerufen werden, z.B. über:
http://www.mydomain.de/ fs4staging_1921116/do.CRCTransfer
Das Servlet wird eine Fehlermeldung anzeigen, da im Browser keine Anmeldedaten übermittelt wurden.

Um den Zugriffsschutz für Medien innerhalb der generierten Projektinhalte zu gewährleisten, muss zusätzlich der Filter für die Auslieferung konfiguriert werden

(siehe Kapitel 3.1.4.2 Seite 18).

3.1.4.2 Konfiguration des Multi-Access-Control-Filters

aclFilter.conf

```
FILTERS=fs4security_secureMedia_DE,fs4security_secureMedia_EN,fs4security_secureMedia_pic

# alle sprachabhängigen Medien im Verzeichnis secure media DE
fs4security_secureMedia_DE.mapping=/fs4staging_1921116/fs4_security/media/de/secure_media/*
fs4security_secureMedia_DE.pathPrefix=/fs4_security
fs4security_secureMedia_DE.defaultAccess=deny

# alle sprachabhängigen Medien im Verzeichnis secure media EN
fs4security_secureMedia__EN.mapping=/fs4staging_1921116/fs4_security/media/en/secure_media/*
fs4security_secureMedia__EN.pathPrefix=/fs4_security
fs4security_secureMedia__EN.defaultAccess=deny

# ein sprachunabhängiges Medium
fs4security.mapping=/fs4staging_1921116/fs4_security/media/secure_media/picture.jpg
fs4security.pathPrefix=/fs4_security
fs4security.defaultAccess=deny
```

Abbildung 3-6: Konfiguration aclFilter.conf

FILTERS: Über den Parameter kann ein Bezeichner für den zu konfigurierenden Filter angegeben werden. Die Konfiguration des Filters folgt dann der Syntax:
Bezeichner.Konfigurationsparameter
Es können beliebig viele Filter als kommaseparierte Liste angegeben werden.

filter.mapping: Zuordnung eines Generierungs-Verzeichnisses zum Filter (oder allgemeiner eine absolute Pfadangabe in der Webanwendung). Die Inhalte des hier angegebenen Verzeichnisses werden durch den Filter überprüft. Grundlage für das Mapping des Filters ist immer:

- Mapping muss den Servlet-Context der Webanwendung enthalten (hier /`fs4staging_1921116`)
- Präfix (zur Vervollständigung der ACL-Information im Live-System) (hier /`fs4_security`)
- Pfad zum zu schützenden Objekt (Verzeichnis und/oder Datei)

Dabei sollte der Filter immer so schmal wie möglich gewählt werden, um während Auslieferung in das Live-System eine hohe Performance zu gewährleisten.

Pro Filter darf immer nur ein Mapping definiert werden. Werden mehrere Mappings benötigt, beispielsweise für sprachabhängige Inhalte, muss für jedes Mapping ein neuer Filter definiert werden.

Wird der Platzhalter * als Abschluss des Mappings angegeben (siehe Abbildung

3-6), wird des gesamte Verzeichnis inklusive aller untergeordneten Elemente überprüft. Wird der Platzhalter nicht angegeben, so wirkt sich der Filter nur auf genau eine Datei aus, z.B.:
`filter.mapping=/fs4staging_1921116/fs4_security/media/de/secure_media/b.jpg`

Das Mapping muss immer individuell angepasst werden und ist stark abhängig von der definierten Pfaderstellung in der Generierung (default, Infix, Multiview):

- Default: Bei dieser Methode wird für jede Sprache des Projekts ein eigener Unterordner auf dem Webserver generiert (de, en, etc.). Für das Filter-Mapping muss beachtet werden, dass sprachabhängige Inhalte in unterschiedlichen Ordnern generiert werden. Für jeden Ordner muss dann ein eigener Filter mit einem angepassten Mapping definiert werden.
- Multiview und Infix: Bei diesen Methoden existieren keine sprachspezifischen Unterordner, stattdessen werden die Dateien für jede Sprache mit dem jeweiligen Sprachkürzel gekennzeichnet, z.B. index.html.de, oder index.de.html. Damit genügt ein Filter um alle gewünschten Inhalte – sprachabhängig oder sprachunabhängig mit einzubeziehen.

Beispiele zum Filter-Mapping (bei Standard-Pfaderzeugung "default"):

Die Beispiele zeigen die Filterkonfiguration für einen bestimmten Medien-Ordner des Projekts. Das Präfix für die Beispielkonfiguration lautet `/fs4_security`, das zu schützende Verzeichnis `secure_media`. Für das Mapping muss der entsprechende Ordner (ausgehend vom Root-Knoten) angegeben werden, also z.B.:
`filter.mapping=/fs4staging_1921116/fs4_security/media/de/secure_media/*`

Für das Filter-Mapping sollte beachtet werden, dass sprachabhängige Medien in anderen Verzeichnissen abgelegt werden, als sprachunabhängige Medien. Das bedeutet, um alle Medien im Verzeichnis `secure_media` zu schützen, sind unterschiedliche Filterkonfigurationen erforderlich:

Beispiel für einen sprachunabhängigen Filter:
Filterkonfiguration für sprachunabhängige Medien im Zielordner `secure_media`:

```
FILTERS=fs4security_1
fs4security_1.mapping=/fs4staging_1921116/fs4_security/media/secure_media/*
fs4security_1.pathPrefix=/fs4_security
fs4security_1.defaultAccess=deny
```

Beispiel für einen sprachabhängigen Filter:
Filterkonfiguration für sprachabhängige Medien in den Sprachen DE und EN im Zielordner `secure_media`:

```
FILTERS=fs4security_DE,fs4security_EN
```

```
fs4security_DE.mapping=/fs4staging_1921116/fs4_security/media/de/secure_media/*
fs4security_DE.pathPrefix=/fs4_security
fs4security_DE.defaultAccess=deny

fs4security_EN.mapping=/fs4staging_1921116/fs4_security/media/en/secure_media/*
fs4security_EN.pathPrefix=/fs4_security
fs4security_EN.defaultAccess=deny
```

Das bedeutet, für jede Sprache muss ein Filtereintrag vorgenommen werden.

Beispiel für einen Filter für ein sprachunabhängiges Medium:
Filterkonfiguration für genau ein sprachunabhängiges Medium

```
FILTERS=fs4security
fs4security.mapping=/fs4staging_1921116/fs4_security/media/secure_media/picture.jpg
fs4security.pathPrefix=/fs4_security
fs4security.defaultAccess=deny
```

filter.pathPrefix: Zuordnung eines Präfixes, der zur Ergänzung des Pfads innerhalb der Access-Control-Datenbank genutzt werden soll.
Hintergrund: Die Datenbank kann mehrere FirstSpirit-Projekte verwalten. Die Identifizierung erfolgt dann eindeutig anhand des vollständigen Pfads inklusive Präfix. Der Parameter `pathPrefix` definiert, ab welchem Teil, der vom Filter berücksichtigten URL, die Berechtigungsprüfung in der Access-Control-Datenbank erfolgen soll.

Beispiel:
http://myServer:8000/fs4staging_1921116/fs4_security/de/homepage/index.jsp
Die Prüfung n der Access-Control-Datenbank erfolgt dann ab dem folgenden Pfad:
/fs4_security/de/homepage/index.jsp

Der hier definierte Pfad muss der Einstellung im Bereich "Projekteigenschaften/Berechtigungen" entsprechen (vgl. "Präfix für die Access-Control-Datenbank" in Kapitel 3.1.1).

filter.extensions: Definition der Datei-Endungen, die durch den Filter überprüft werden sollen. Die Datei-Endungen können als kommaseparierte Liste übergeben werden, z.B.:
`filter.extensions=*.jpg,*.png,*.gif,*.jsp`

Wird der Parameter NICHT angegeben, werden alle Inhalte des unter `filter.mapping` angegebenen Generierungs-Verzeichnisses gefiltert.

Wird die Filterung auf bestimmte Datei-Endungen eingeschränkt (z.B. JSP-Dateien), kann es (durch die Angabe von "Welcome Files") zu einer automatischen Vervollständigung der Servlet-Engine kommen, so dass diese Seiten dennoch angezeigt werden.
Das bedeutet, der folgende Aufruf wird durch den Filter geschützt: http://myServer:8000/fs4staging_1921116/fs4_security/de/homepage/index.jsp
Nicht aber der Aufruf:
http://myServer:8000/fs4staging_1921116/ fs4_security/de/homepage/
der anschließend die eigentliche geschützte JSP-Seite anzeigt.

Dabei handelt es sich nicht um einen Fehler in FirstSpirit, sondern um ein Konfigurationsproblem. In diesem Fall sollten die Einschränkung auf bestimmte Datei-Endungen nicht angegeben werden.

filter.defaultAccess: Über diesen Parameter kann das Standard-Zugriffs-Verhalten des Filters definiert werden, wenn keine Rechte für das Objekt über die Rechte-Variable gespeichert sind oder wenn kein Eintrag in der Access-Control-Datenbank besteht. (Das trifft beispielsweise auf Ordner zu, da die Access-Control-Datenbank nur Informationen zu Dateien verwaltet.)
Ist der Wert `allow` definiert, wird der Zugriff auf die gefilterten Inhalte für alle Benutzer erlaubt (nur wenn keine anderslautenden Rechte über die Rechte-Variable definiert wurden).
Ist der Wert `deny` definiert, wird der Zugriff immer unterbunden, wenn keine Zugriffsrechte für das Objekt definiert wurden.

filter.activity: Über diesen Parameter kann ein bestimmtes Zugriffsrecht definiert werden, das über die Eingabekomponente für Zugriffsrechte (CMS_INPUT_PERMISSION) gepflegt wird. Dabei können nur die Zugriffsrechte berücksichtigt werden, die im Formularbereich der Eingabekomponente innerhalb der <ACTIVITIES>-Tags definiert wurden, z.B.:

```
<CMS_INPUT_PERMISSION ...>

  <ACTIVITIES>

    <ACTIVITY name="read"/>

          ...

    <ACTIVITY name="write"/>

          ...

  </ACTIVITIES>

   ...

</CMS_INPUT_PERMISSION>
```

FirstSpirit entscheidet dann anhand des Wertes der Eingabekomponente im Register Metadaten, ob ein Benutzer Zugriff auf ein Objekt hat oder nicht.

4 Klassen und Methoden

Für den Zugriff auf die Access-Control-Datenbank sind folgende Klassen verfügbar:

- Klasse "AccessControlDb" (Paket "de.espirit.firstspirit.acl.db"):
  Die Klasse enthält Methoden für den Zugriff auf die Access-Control-Datenbank, um z.B. eine oder mehrere ACLs anzufordern
  (vgl. Kapitel 4.1.1 Seite 23).
- Klasse "Acl" (Paket "de.espirit.firstspirit.acl"):
  Die Klasse enthält Methoden, um die gespeicherten Berechtigungsinformationen (Benutzerrechte bzw. Zugriffsrechte) zu einer oder mehreren Dateien zu erhalten
  (vgl. Kapitel 4.1.2 Seite 41).
- Klasse "Activity" (Paket "de.espirit.firstspirit.acl"):
  Die Klasse enthält Methoden für die Ausgabe bzw. Auswertung von Berechtigungen
  (vgl. Kapitel 4.1.3 Seite 49).
- Klasse "File" (Paket "de.espirit.firstspirit.acl"):
  Die Klasse enthält Methoden zur Ausgabe der gespeicherten Informationen zu einer generierten Datei, z.B. die ID, den eindeutigen Bezeichner, usw.
  (vgl. Kapitel 4.1.4 Seite 55).

Die in diesem Kapitel beschriebenen Klassen sind nicht Bestandteil der öffentlichen FirstSpirit API und werden sich in zukünftigen Versionen ändern.

4.1.1 Klasse "AccessControlDb" (Paket "de.espirit.firstspirit.acl.db")

Die Klasse "AccessControlDb" aus dem Paket "de.espirit.firstspirit.acl.db" ermöglicht den Zugriff auf die lokale bzw. die in der Webanwendung enthaltene Access-Control-Datenbank über ein Skript oder eine JSP-Seite.

4.1.1.1 Zugriff auf die Access-Control-Datenbank (JSP-Seite)

Wurde in der Webanwendung das Access-Control-Servlet registriert (vgl. Kapitel 3.1.3.2 Seite 14), so stellt das Servlet die Access-Control-Datenbank innerhalb des Anwendungskontexts (Objekt "application") zur Verfügung. Die Bezeichnung des Attributes im Anwendungskontext ist durch die Konstante ACCESS_CONTROL_DB der Klasse "AccessControlServlet" festgelegt.

Der Aufruf für das Laden der Access-Control-Datenbank innerhalb einer JSP-Seite lautet somit:

```
<%@
  page
  import="de.espirit.firstspirit.acl.db.AccessControlDb"
%>
<%@
  page
  import="de.espirit.firstspirit.acl.servlet.AccessControlServlet"
%>


<%
  final AccessControlDb db =
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);
%>
```

4.1.1.2 Zugriff auf die Access-Control-Datenbank (Skript)

Die Access-Control-Datenbank steht in einem Auftrag (innerhalb eines Skripts im Kontext) bereits zur Verfügung. Um auf die Access-Control-Datenbank zuzugreifen ist die Methode "getAccessControlDb()" auf dem Objekt "context" aufzurufen. Die Angabe einer Importanweisung ist nicht notwendig:

```
db = context.getAccessControlDb();
```

4.1.1.3 Methodenübersicht

Die wichtigsten Methoden der Klasse "AccessControlDb" sind:

- Methode "getAcls()":
  Rückgabe einer Liste aller ACLs
  (vgl. Kapitel 4.1.1.4 Seite 26).
- Methode "getFiles(Acl)":
  Rückgabe einer Liste aller Dateien einer ACL
  (vgl. Kapitel 4.1.1.5 Seite 27).
- Methode "getFile(String)":
  Rückgabe einer bestimmten Datei anhand ihres Pfades
  (vgl. Kapitel 4.1.1.6 Seite 29).
- Methode "getAcl(long)":
  Rückgabe einer bestimmten ACL anhand ihrer ID
  (vgl. Kapitel 4.1.1.7 Seite 30).
- Methode "deleteDir(String)":
  Löschen aller Dateien unter Angabe eines Pfades
  (vgl. Kapitel 4.1.1.8 Seite 32).
- Methode "deleteFile(File)":
  Löschen einer Datei
  (vgl. Kapitel 4.1.1.9 Seite 33)
- Methode "deleteAcl(long)":
  Löschen einer bestimmten ACL unter Angabe ihrer ID
  (vgl. Kapitel 4.1.1.10 Seite 34).
- Methode "deleteOldFiles(String, long)":
  Löschen aller Dateien eines Pfades, die älter als das angegebene Datum sind.
  (vgl. Kapitel 4.1.1.11 Seite 34).
- Methode "getFiles(String, long)":
  Rückgabe aller Dateien eines Pfades, die jünger oder gleich dem angegebenen Datum sind.
  (vgl. Kapitel 4.1.1.12 Seite 36).
- Methode "getFiles(long)":
  Rückgabe aller Dateien anhand der ID
  (vgl. Kapitel 4.1.1.13 Seite 38).

4.1.1.4 Methode "getAcls()"

Die Methode "getAcls()" liefert alle ACLs einer Access-Control-Datenbank als Liste zurück ("List<Acl>"). Übergabeparameter sind nicht anzugeben.

Methodensignatur:

```
public java.util.List<de.espirit.firstspirit.acl.Acl> getAcls() throws
com.sleepycat.je.DatabaseException
```

Methodensignatur (Kurzform):

```
getAcls():List<Acl>
```

Beispiel (Skript):

```
db = context.getAccessControlDb();

acls = db != null ? db.getAcls() : null;
```

Beispiel (JSP-Seite):

```
<%@

  page

  import="de.espirit.firstspirit.acl.db.AccessControlDb,

    de.espirit.firstspirit.acl.servlet.AccessControlServlet,

    java.util.List,

    de.espirit.firstspirit.acl.Acl" %><%



  final AccessControlDb db =

    (AccessControlDb) application

      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);


  final List<Acl> acls = db != null ? db.getAcls() : null;



%>
```

4.1.1.5 Methode "getFiles(Acl)"

Die Methode "getFiles(Acl)" liefert eine Liste aller Dateien zu der übergebenen ACL zurück ("List<File>").

Methodensignatur:

```
public java.util.List<de.espirit.firstspirit.acl.File> getAcls(final
de.espirit.firstspirit.acl.Acl acl) throws
com.sleepycat.je.DatabaseException
```

Methodensignatur (Kurzform):

```
getFiles(Acl):List<File>
```

Beispiel (Skript):

```
db = context.getAccessControlDb();

acls = db != null ? db.getAcls() : null;



if (acls != null) {

  for (acl : acls) {

    files = db.getFiles(acl);

  }

}
```

Beispiel (JSP-Seite):

```
<%@

  page

  import="de.espirit.firstspirit.acl.db.AccessControlDb,

    de.espirit.firstspirit.acl.servlet.AccessControlServlet,

    java.util.List,

    de.espirit.firstspirit.acl.Acl,

    de.espirit.firstspirit.acl.File" %><%




  final AccessControlDb db =
```

```
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);

  final List<Acl> acls = db != null ? db.getAcls() : null;

  if (acls != null) {

    for (final Acl acl: acls) {
      final List<File> files = db.getFiles(acl);

    }

  }

%>
```

4.1.1.6 Methode "getFile(String)"

Die Methode "getFile(String)" liefert eine bestimmte Datei (aller ACLs) zurück. Als Übergabeparameter ist der vollständige Pfad der Datei anzugeben.

Die lokale Access-Control-Datenbank verwendet keinen Präfix, jedoch die Access-Control-Datenbank in der Webanwendung. Bei einer Veröffentlichung ist daher das im Projekt angegebene Präfix dem Pfad voranzustellen.

Methodensignatur:

```
public de.espirit.firstspirit.acl.File getFile(final java.lang.String
path) throws com.sleepycat.je.DatabaseException
```

Methodensignatur (Kurzform):

```
getFile(String):File
```

Beispiel (JSP-Seite):

```
<%@

  page

  import="de.espirit.firstspirit.acl.db.AccessControlDb,

    de.espirit.firstspirit.acl.servlet.AccessControlServlet,

    de.espirit.firstspirit.acl.File" %><%




  final AccessControlDb db =

    (AccessControlDb) application

      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);


  final File file =

    db

      .getFile("$CMS_VALUE(deploymentPrefix)$$CMS_REF(#global.ref,
abs:2)$");
```

```
%>
```

Im Beispiel ist "deploymentPrefix" eine Struktur-Variable, die in der Struktur-Verwaltung definiert wurde. Die Variable wird im Auftrag mit dem entsprechenden Präfix überschrieben.

4.1.1.7 Methode "getAcl(long)"

Die Methode "getAcl(long)" liefert eine bestimmte ACL zurück. Als Parameter ist die eindeutige ID einer ACL anzugeben.

Da die Vergabe der IDs für die einzelnen ACLs fortlaufend ist und die IDs bei jeder Generierung vergeben werden, ist die Methode für die Verwendung über mehrere Generierungen hinweg nicht geeignet. Es wird daher empfohlen, in Skripten und JSP-Seiten die Methoden "getAcls()" (vgl. Kapitel 4.1.1.4 Seite 26) bzw. "getFile(String)" (vgl. Kapitel 4.1.1.6 Seite 29) zu verwenden.
Eine Ausnahme von dieser Regel ist die Ermittlung der ACL, wenn die Datei mit "getFile(String)" geholt und die Methode "getAcl()" aufgerufen wurde.

Methodensignatur:

```
public de.espirit.firstspirit.acl.Acl getAcl(final long id) throws
com.sleepycat.je.DatabaseException
```

Methodensignatur (Kurzform):

```
getAcl(long):Acl
```

Beispiel (JSP-Seite):

```
<%@

  page

  import="de.espirit.firstspirit.acl.db.AccessControlDb,

    de.espirit.firstspirit.acl.servlet.AccessControlServlet,

    de.espirit.firstspirit.acl.Acl" %><%



  final AccessControlDb db =

    (AccessControlDb) application
```

```
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);


  final Acl acl = db.getAcl(1L);
%>
```

4.1.1.8 Methode "deleteDir(String)"

Mit der Methode "deleteDir(String)" ist es möglich, z.B. nach dem Wechsel des Präfixes, alle Dateien aus der Access-Control-Datenbank zu löschen, deren Pfad mit dem übergebenen Pfad beginnt.

Die lokale Access-Control-Datenbank verwendet keinen Präfix, jedoch die Access-Control-Datenbank in der Webanwendung. Bei einer Veröffentlichung ist daher das im Projekt angegebene Präfix dem Pfad voranzustellen.

Methodensignatur:

```
public void deleteDir(final java.lang.String path) throws
com.sleepycat.je.DatabaseException
```

Methodensignatur (Kurzform):

```
deleteDir(String):void
```

Beispiel (JSP-Seite):

```
<%@

  page

  import="de.espirit.firstspirit.acl.db.AccessControlDb,

    de.espirit.firstspirit.acl.servlet.AccessControlServlet" %><%



  final AccessControlDb db =

    (AccessControlDb) application

      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);


  db.deleteDir("/fs4_security");



%>
```

4.1.1.9 Methode "deleteFile(File)"

Mit der Methode "deleteFile(File)" ist es möglich, die angegebene Datei aus der Access-Control-Datenbank zu löschen.

Methodensignatur:

```
public void deleteFile(final de.espirit.firstspirit.acl.File file) throws
com.sleepycat.je.DatabaseException
```

Methodensignatur (Kurzform):

```
deleteFile(File):void
```

Beispiel (JSP-Seite):

```
<%@

  page

  import="de.espirit.firstspirit.acl.db.AccessControlDb,

    de.espirit.firstspirit.acl.servlet.AccessControlServlet,

    de.espirit.firstspirit.acl.File" %><%




  final AccessControlDb db =

    (AccessControlDb) application

      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);


  final File file =

    db

      .getFile("/fs4_security/de/index.html");


  db.deleteFile(file);




%>
```

4.1.1.10 Methode "deleteAcl(long)"

Mit der Methode "deleteAcl(long)" kann eine bestimmte ACL, unter Angabe der ID, aus Access-Control-Datenbank gelöscht werden.

Methodensignatur:

```
public void deleteAcl(final long id) throws
com.sleepycat.je.DatabaseException
```

Methodensignatur (Kurzform):

```
deleteAcl(long):void
```

Beispiel (JSP-Seite):

```
<%@

  page

  import="de.espirit.firstspirit.acl.db.AccessControlDb,

    de.espirit.firstspirit.acl.servlet.AccessControlServlet" %><%




  final AccessControlDb db =

    (AccessControlDb) application

      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);



  db.deleteAcl(1L);

%>
```

4.1.1.11 Methode "deleteOldFiles(String, long)"

Mit der Methode "deleteOldFiles(String, long)" werden alle Dateien, die älter als das übergebene Datum sind und deren Pfad mit dem übergebenen Pfad beginnen, aus der Access-Control-Datenbank entfernt.

Die Methode erwartet als ersten Übergabeparameter den Pfad, der zu löschenden Dateien. Die Angabe "null" bedeutet, das alle Dateien (ohne Beachtung eines Präfixes) berücksichtigt werden sollen. Als zweiter Parameter ist das Datum als Zeitstempel (Datentyp: "long") anzugeben. Ein lesbares Datumsformat kann durch

Java-Methoden in einen Zeitstempel überführt werden.

Für das Entfernen der Dateien wird der übergebene Zeitstempel mit dem Zeitstempel der letzten Aktualisierung der gespeicherten Daten eines Objektes verglichen (vgl. Kapitel 4.1.4.7 Seite 67). Ist der Zeitstempel der letzten Aktualisierung der gespeicherten Daten kleiner als der übergebene Zeitstempel und beginnt der Pfad des Objektes mit dem übergebenen Pfad, so wird das Objekt bzw. die Datei gelöscht.

Die lokale Access-Control-Datenbank verwendet keinen Präfix, jedoch die Access-Control-Datenbank in der Webanwendung. Bei einer Veröffentlichung ist daher das im Projekt angegebene Präfix dem Pfad voranzustellen.

Methodensignatur:

```
public void deleteOldFiles(final java.lang.String pathPrefix, final long
timestamp) throws com.sleepycat.je.DatabaseException
```

Methodensignatur (Kurzform):

```
deleteOldFiles(String, long):void
```

Beispiel (JSP-Seite):

```
<%@

  page

  import="de.espirit.firstspirit.acl.db.AccessControlDb,

    de.espirit.firstspirit.acl.servlet.AccessControlServlet" %><%



  final AccessControlDb db =

    (AccessControlDb) application

      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);


  db.deleteOldFiles("/fs4_security", 1196674444102L);
```

```
%>
```

4.1.1.12 Methode "getFiles(String, long)"

Die Methode "getFiles(String, long)" liefert eine Liste aller Dateien zurück, deren Pfade mit dem übergebenen Pfad beginnen und die genauso alt oder jünger sind als das übergebene Datum.

Die Methode erwartet als ersten Übergabeparameter den Pfad der Dateien. Die Angabe "null" bedeutet, dass alle Dateien (ohne Beachtung eines Präfixes) berücksichtigt werden sollen. Als zweiter Parameter ist das Datum als Zeitstempel (Datentyp: "long") anzugeben. Ein lesbares Datumsformat kann durch Java-Methoden in einen Zeitstempel überführt werden.

Für die Ermittlung der Dateien wird der übergebene Zeitstempel mit dem Zeitstempel der letzten Aktualisierung der gespeicherten Daten eines Objektes verglichen (vgl. Kapitel 4.1.4.7 Seite 67). Ist der Zeitstempel der letzten Aktualisierung der gespeicherten Daten gleich oder größer als der übergebene Zeitstempel und beginnt der Pfad des Objektes mit dem übergebenen Pfad, so wird das Objekt bzw. die Datei zurückgeliefert.

Die lokale Access-Control-Datenbank verwendet keinen Präfix, jedoch die Access-Control-Datenbank in der Webanwendung. Bei einer Veröffentlichung ist daher das im Projekt angegebene Präfix dem Pfad voranzustellen.

Die Methode "getFiles(String, long)" kann in einem Skript eines Auftrags verwendet werden, um Daten zu allen generierten Dateien zu ermitteln. Hierfür ist der Startzeitpunkt der Generierung zu verwenden. Der Startzeitpunkt der Generierung steht mit Hilfe der Methode "getStartTime()" im Skript-Kontext eines Auftrags zur Verfügung. Die Methode "getStartTime()" liefert ein "java.util.Date"-Objekt zurück. Ein Beispiel für die Verwendung von "getStartTime()" ist unten aufgeführt.

Methodensignatur:

```
public java.util.List<de.espirit.firstspirit.acl.File> getFiles(final
java.lang.String pathPrefix, final long timestamp) throws
com.sleepycat.je.DatabaseException
```

Methodensignatur (Kurzform):

```
getFiles(String, long):List<File>
```

Beispiel (Skript):

```
import java.text.SimpleDateFormat;

import java.util.Locale;


db = context.getAccessControlDb();

files =

  db != null

  ? db.getFiles(null, context.getStartTime().getTime())

  : null;


if (files != null) {

  df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.GERMANY);

  for (file : files) {

    print(

         "   path=" + file.getPath()

      + ", lastModified=" + df.format(file.getLastModified())

      + ", lastUpdate=" + df.format(file.getLastUpdate())

      + ", crc=" + file.getCrc32()

    );

  }

}
```

Beispiel (JSP-Seite):

```
<%@
  page
  import="de.espirit.firstspirit.acl.db.AccessControlDb,
    de.espirit.firstspirit.acl.servlet.AccessControlServlet,
    java.util.List,
    de.espirit.firstspirit.acl.File" %><%



  final AccessControlDb db =
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);



  final List<File> files =
    db
      .getFiles("/fs4_security", 1196675825992L);



%>
```

4.1.1.13 Methode "getFiles(long)"

Die Methode "getFiles(long)" liefert eine Liste aller Dateien zurück, die der übergebenen ID entsprechen.

Mehrere Dateien bzw. Objekte haben die gleiche ID, wenn bei der Generierung (von einem Element ausgehend) mehrere Dateien erzeugt werden. Dies ist z.B. bei einem Datenquellen-Absatz der Fall. Hier werden bei der Generierung der Seitenreferenz in der Struktur-Verwaltung für die Datensätze u. U. mehrere Dateien erzeugt. Die ID der Dateien ist daher gleich (der Präfix und das Erstellungsdatum der Dateien können jedoch unterschiedlich sein). Um eine Liste aller Dateien zu erhalten, ist in diesem Fall die ID der Seitenreferenz anzugeben.

Methodensignatur:

```
public java.util.List< de.espirit.firstspirit.acl.File File>
getFiles(final long elementId) throws com.sleepycat.je.DatabaseException
```

Methodensignatur (Kurzform):

```
getFiles(long):List<File>
```

Beispiel (Skript):

```
import java.text.SimpleDateFormat;

import java.util.Locale;



db = context.getAccessControlDb();

files =

  db != null

  ? db.getFiles(123456L)

  : null;



if (files != null) {

  df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.GERMANY);

  for (file : files) {

    print(

         "   path=" + file.getPath()

      + ", lastModified=" + df.format(file.getLastModified())

      + ", lastUpdate=" + df.format(file.getLastUpdate())

      + ", crc=" + file.getCrc32()

    );

  }

}
```

Beispiel (JSP-Seite):

```
<%@
  page
  import="de.espirit.firstspirit.acl.db.AccessControlDb,
    de.espirit.firstspirit.acl.servlet.AccessControlServlet,
    java.util.List,
    de.espirit.firstspirit.acl.File" %><%

  final AccessControlDb db =
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);

  final List<File> files =
    db
      .getFiles(123456L);
%>
```

4.1.2 Klasse "Acl" (Paket "de.espirit.firstspirit.acl")

Die Klasse "Acl" aus dem Paket "de.espirit.firstspirit.acl" ermöglicht den Zugriff auf die gespeicherten Berechtigungsinformationen (Benutzerrechte bzw. Zugriffsrechte) zu einer oder mehreren Dateien.

Font-Images verfügen über keine Berechtigungsinformationen.

4.1.2.1 Methodenübersicht

Die wichtigsten Methoden der Klasse "Acl" sind:

- Methode "getId()":
  Rückgabe der ID einer ACL
  (vgl. Kapitel 4.1.2.2 Seite 42).
- Methode "getDocument()":
  Rückgabe des symbolischen Namens der verwendeten Gruppenhierarchie (z.B. in der Rechtedefinitions-Eingabekomponente)
  (vgl. Kapitel 4.1.2.3 Seite 43).
- Methode "getActivities()":
  Rückgabe der konfigurierten und gespeicherten Aktivitäten
  (vgl. Kapitel 4.1.2.4 Seite 44).
- Methode "getPermissions()":
  Rückgabe der gespeicherten Rechtedefinitionen für die einzelnen Aktivitäten
  (vgl. Kapitel 4.1.2.5 Seite 46).
- Methode "getPriority()":
  Rückgabe der konfigurierten und gespeicherten Standardpriorität bei Überschneidungen der Rechte.
  (vgl. Kapitel 4.1.2.6 Seite 48).

4.1.2.2 Methode "getId()"

Die Methode "getId()" liefert die ID einer ACL zurück.

Methodensignatur:

```
public long getId()
```

Methodensignatur (Kurzform):

```
getId():long
```

Beispiel (JSP-Seite):

```
<%@
  page
  import="de.espirit.firstspirit.acl.db.AccessControlDb,
    de.espirit.firstspirit.acl.servlet.AccessControlServlet,
    java.util.List,
    de.espirit.firstspirit.acl.Acl " %><%

  final AccessControlDb db =
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);

  final List<Acl> acls = db != null ? db.getAcls() : null;

  if (acls != null) {

    for (final Acl acl: acls) {
%><%= acl.getId() %><br /><%
    }
  }
%>
```

4.1.2.3 Methode "getDocument()"

Die Methode "getDocument()" liefert den symbolischen Namen der verwendeten Gruppenhierarchie zurück. Der symbolische Name der verwendeten Gruppenhierarchie wird u.a. in der Rechtedefinitions-Eingabekomponente mit dem Parameter "group" festgelegt:

```
<CMS_INPUT_PERMISSION name="st_permission" group="GruppenFile">
  <ACTIVITIES>
    <ACTIVITY name="read"/>
    <ACTIVITY name="write"/>
  </ACTIVITIES>
  <LANGINFOS>
    <LANGINFO label="CMS_INPUT_PERMISSION" lang="*"/>
  </LANGINFOS>
</CMS_INPUT_PERMISSION>
```

Methodensignatur:

```
public java.lang.String getDocument()
```

Methodensignatur (Kurzform):

```
getDocument():String
```

Beispiel (JSP-Seite):

```
<%@
  page
  import="de.espirit.firstspirit.acl.db.AccessControlDb,
    de.espirit.firstspirit.acl.servlet.AccessControlServlet,
    java.util.List,
    de.espirit.firstspirit.acl.Acl" %><%



  final AccessControlDb db =
    (AccessControlDb) application
```

```
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);


  final List<Acl> acls = db != null ? db.getAcls() : null;


  if (acls != null) {


    for (final Acl acl: acls) {
%><%= acl.getDocument() %><br /><%
    }
  }



%>
```

4.1.2.4 Methode "getActivities()"

Die Methode "getActivities()" liefern die festgelegten und gespeicherten Aktivitätsbezeichner (Teil der Zugriffsrechte) als Liste zurück. Aktivitäten werden u.a. in der Rechtedefinitions-Eingabekomponente mit dem Tag "<ACTIVITIES>" festgelegt:

```
<CMS_INPUT_PERMISSION name="st_permission" group="GruppenFile">
  <ACTIVITIES>
    <ACTIVITY name="read"/>
    <ACTIVITY name="write"/>
  </ACTIVITIES>
  <LANGINFOS>
    <LANGINFO label="CMS_INPUT_PERMISSION" lang="*"/>
  </LANGINFOS>
</CMS_INPUT_PERMISSION>
```

Jede Aktivität kann über abweichende Rechte verfügen.

Methodensignatur:

```
public List<java.lang.String> getActivities()
```

Methodensignatur (Kurzform):

```
getActivities():List<String>
```

Beispiel (JSP-Seite):

```
<%@
  page
  import="de.espirit.firstspirit.acl.db.AccessControlDb,
    de.espirit.firstspirit.acl.servlet.AccessControlServlet,
    java.util.List,
    de.espirit.firstspirit.acl.Acl" %><%



  final AccessControlDb db =
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);


  final List<Acl> acls = db != null ? db.getAcls() : null;


  if (acls != null) {


    for (final Acl acl: acls) {
%><%= acl.getActivities() %><br /><%
    }
  }
```

```
%>
```

4.1.2.5 Methode "getPermissions()"

Die Methode "getPermissions()" liefert eine Map zurück. Die Map enthält sämtliche Berechtigungsinformationen zu den zugehörigen Dateien. Der Schlüssel eines Eintrages der Map ist ein Aktivitätsbezeichner (vgl. auch Kapitel 4.1.2.4 Seite 44) und der Wert ist ein Berechtigungsobjekt ("Activity"). Mit dem Berechtigungsobjekt können dann die Berechtigungen einer Aktivität ausgewertet werden.

Methodensignatur:

```
public java.util.Map<java.lang.String,
de.espirit.firstspirit.acl.Activity> getPermissions()
```

Methodensignatur (Kurzform):

```
getPermissions():Map<String, Activity>
```

Beispiel (JSP-Seite):

```
<%@
  page
  import="de.espirit.firstspirit.acl.db.AccessControlDb,
    de.espirit.firstspirit.acl.servlet.AccessControlServlet,
    java.util.List,
    de.espirit.firstspirit.acl.Acl" %><%

  final AccessControlDb db =
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);

  final List<Acl> acls = db != null ? db.getAcls() : null;
  if (acls != null) {
    for (final Acl acl: acls) {
%><%= acl.getPermissions() %><br /><%
    }
```

```
  }

%>
```

4.1.2.6 Methode "getPriority()"

Die Methode "getPriority()" liefert den festgelegten und gespeicherten Wert für das Konfliktverhalten bei sich überschneidenden Rechtekonfigurationen für ein Berechtigungsobjekt. Den Standardwert für das Konfliktverhalten wird u.a. in der Rechtedefinitions-Eingabekomponente mit dem Parameter "priority" festgelegt:

```
<CMS_INPUT_PERMISSION name="st_permission" priority="deny"
group="GruppenFile">

  <ACTIVITIES>

    <ACTIVITY name="read"/>

    <ACTIVITY name="write"/>

  </ACTIVITIES>

  <LANGINFOS>

    <LANGINFO label="CMS_INPUT_PERMISSION" lang="*"/>

  </LANGINFOS>

</CMS_INPUT_PERMISSION>
```

Methodensignatur:

```
public de.espirit.firstspirit.service.permission.Priority getPriority()
```

Methodensignatur (Kurzform):

```
getPriority():Priority
```

Beispiel (JSP-Seite):

```
<%@

  page

  import="de.espirit.firstspirit.acl.db.AccessControlDb,

    de.espirit.firstspirit.acl.servlet.AccessControlServlet,

    java.util.List,

    de.espirit.firstspirit.acl.Acl" %><%




  final AccessControlDb db =
```

```
    (AccessControlDb) application

      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);


  final List<Acl> acls = db != null ? db.getAcls() : null;


  if (acls != null) {


    for (final Acl acl: acls) {
%><%= acl.getPriority() %><br /><%
    }
  }
%>
```

4.1.3 Klasse "Activity" (Paket "de.espirit.firstspirit.acl")

Mit der Klasse "Activity" aus dem Paket "de.espirit.firstspirit.acl" können gespeicherte Berechtigungsinformationen einer Aktivität zu einer oder mehreren Dateien ausgewertet und weiter verarbeitet werden.

Aktivitäten werden in u.a. in der Rechtedefinitions-Eingabekomponente definiert. Die Berechtigungen bzw. Benutzerrechte werden dann mit der Rechtedefinitions-Eingabekomponente auf den jeweiligen Objekten in den einzelnen Verwaltungen festgelegt (Meta-Informationen).

4.1.3.1 Methodenübersicht

Die wichtigsten Methoden der Klasse "Activity" sind:

- Methode "getName()":
  Rückgabe des gespeicherten Aktivitätsbezeichners
  (vgl. Kapitel 4.1.3.2 Seite 50).
- Methode "getAllowed()":
  Rückgabe der Benutzer/Gruppen für die der Zugriff auf die Dateien erlaubt ist
  (vgl. Kapitel 4.1.3.3 Seite 51).
- Methode "getForbidden()":
  Rückgabe der Benutzer/Gruppen für die der Zugriff auf die Dateien verboten ist
  (vgl. Kapitel 4.1.3.4 Seite 53).

4.1.3.2 Methode "getName()"

Die Methode "getName()" liefert den Bezeichner der Aktivität zurück.

Methodensignatur:

```
public java.lang.String getName()
```

Methodensignatur (Kurzform):

```
getName():String
```

Beispiel (JSP-Seite):

```
<%@
  page
  import="de.espirit.firstspirit.acl.db.AccessControlDb,
    de.espirit.firstspirit.acl.servlet.AccessControlServlet,
    java.util.List,
    de.espirit.firstspirit.acl.Acl,
    de.espirit.firstspirit.acl.Activity,
    java.util.Map" %><%



  final AccessControlDb db =
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);


  final List<Acl> acls = db != null ? db.getAcls() : null;


  if (acls != null) {


    for (final Acl acl: acls) {
      final Map<String, Activity> permissions =
```

```
        acl.getPermissions();


        if (permissions != null) {
          for(final Activity activity : permissions.values()) {
%><%= activity.getName() %><br /><%
          }
        }

    }
  }


%>
```

4.1.3.3 Methode "getAllowed()"

Mit der Methode "getAllowed()" werden die Gruppen bzw. Benutzer, für die der Zugriff auf einer oder mehreren Dateien erlaubt ist, als Liste zurückgeliefert.

Methodensignatur:

```
public java.util.List<java.lang.String> getAllowed()
```

Methodensignatur (Kurzform):

```
getAllowed():List<String>
```

Beispiel (JSP-Seite):

```
<%@
  page
  import="de.espirit.firstspirit.acl.db.AccessControlDb,
    de.espirit.firstspirit.acl.servlet.AccessControlServlet,
    java.util.List,
    de.espirit.firstspirit.acl.Acl,
    de.espirit.firstspirit.acl.Activity,
```

```
    java.util.Map" %><%


  final AccessControlDb db =
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);

  final List<Acl> acls = db != null ? db.getAcls() : null;

  if (acls != null) {

    for (final Acl acl: acls) {
      final Map<String, Activity> permissions =
        acl.getPermissions();

        if (permissions != null) {
          for(final Activity activity : permissions.values()) {
%><%= activity.getAllowed() %><br /><%
          }
        }

    }
  }


%>
```

4.1.3.4 Methode "getForbidden()"

Mit der Methode "getForbidden()" werden die Gruppen bzw. Benutzer, für die der Zugriff auf einer oder mehreren Dateien verboten ist, als Liste zurückgeliefert.

Methodensignatur:

```
public java.util.List<java.lang.String> getForbidden()
```

Methodensignatur (Kurzform):

```
getForbidden():List<String>
```

Beispiel (JSP-Seite):

```
<%@
  page
  import="de.espirit.firstspirit.acl.db.AccessControlDb,
    de.espirit.firstspirit.acl.servlet.AccessControlServlet,
    java.util.List,
    de.espirit.firstspirit.acl.Acl,
    de.espirit.firstspirit.acl.Activity,
    java.util.Map" %><%



  final AccessControlDb db =
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);


  final List<Acl> acls = db != null ? db.getAcls() : null;


  if (acls != null) {


    for (final Acl acl: acls) {
      final Map<String, Activity> permissions =
```

```
        acl.getPermissions();


        if (permissions != null) {
          for(final Activity activity : permissions.values()) {
%><%= activity.getForbidden() %><br /><%
          }
        }


    }
  }



%>
```

4.1.4 Klasse "File" (Paket "de.espirit.firstspirit.acl")

Die Klasse "File" aus dem Paket "de.espirit.firstspirit.acl" bietet die Möglichkeit auf Informationen einer generierten Datei zurückzugreifen. Verfügbar sind folgende Informationen:

- der vollständige Pfad zum Objekt
- die CRC-32-Prüfsumme des Objektes
- das Datum der letzten Änderung des Objektes
- das Datum der letzten Aktualisierung der gespeicherten Daten eines Objektes
- der eindeutige Bezeichner des Objektes
- die ID des Objektes

4.1.4.1 Methodenübersicht

Die wichtigsten Methoden der Klasse "File" sind:

- Methode "getPath()":
  Rückgabe des vollständigen Pfades zum Objekt (vgl. Kapitel 4.1.4.2 Seite 56).
- Methode "getElementId()":
  Rückgabe der ID des Objektes (vgl. Kapitel 4.1.4.3 Seite 58).
- Methode "getElementUid()":
  Rückgabe des eindeutigen Bezeichners des Objektes
  (vgl. Kapitel 4.1.4.4 Seite 60).
- Methode "getCrc32()":
  Rückgabe der CRC-32-Prüfsumme des Objektes (vgl. Kapitel 4.1.4.5 Seite 63).
- Methode "getLastModified()":
  Rückgabe des Datums der letzten Änderung des Objektes (vgl. Kapitel 4.1.4.6 Seite 65).
- Methode "getLastUpdate()":
  Rückgabe des Datums der letzten Aktualisierung der gespeicherten Daten eines Objektes (vgl. Kapitel 4.1.4.7 Seite 67).
- Methode "getAcl()":
  Rückgabe der ID der ACL (vgl. Kapitel 4.1.4.8 Seite 70).
- Methode "getLength()":
  Rückgabe der Dateigröße (vgl. Kapitel 4.1.4.9 Seite 72).
- Methode "getElementTag()":
  Rückgabe des Tag-Namens eines Objektes (vgl. Kapitel 4.1.4.10 Seite 74).

4.1.4.2 Methode "getPath()"

Mit der Methode "getPath()" wird der gespeicherte, absolute Pfad bis zum Generierungswurzelknoten eines Objektes zurückgeliefert.

Die lokale Access-Control-Datenbank verwendet keinen Präfix, jedoch die Access-Control-Datenbank in der Webanwendung. Bei einer Veröffentlichung wird daher das im Projekt angegebene Präfix dem Pfad vorangestellt und bei der Speicherung berücksichtigt.

Methodensignatur:

```
public String getPath()
```

Methodensignatur (Kurzform):

```
getPath():String
```

Beispiel (Skript):

```
db = context.getAccessControlDb();

acls = db != null ? db.getAcls() : null;


if (acls != null) {
  for (acl: acls) {
    files = db.getFiles(acl);
    if (files != null) {
      for (file : files) {
        print(file.getPath());
      }
    }
  }
}
```

Beispiel (JSP-Seite):

```
<%@
```

```
  page
  import="de.espirit.firstspirit.acl.db.AccessControlDb,
    de.espirit.firstspirit.acl.servlet.AccessControlServlet,
    java.util.List,
    de.espirit.firstspirit.acl.Acl,
    de.espirit.firstspirit.acl.File" %><%



  final AccessControlDb db =
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);


  final List<Acl> acls = db != null ? db.getAcls() : null;


  if (acls != null) {


    for (final Acl acl: acls) {


      final List<File> files = db.getFiles(acl);
      if (files != null) {
        for (File file : files) {
%><%= file.getPath() %><br /><%
        }
      }


    }
  }
```

```
%>
```

4.1.4.3 Methode "getElementId()"

Durch die Methode "getElementId()" wird die ID zurückgeliefert, die bei der Generierung des Objektes gültig war.

Font-Images haben keine ID. Die Rückgabe in diesem Fall ist "-1".

Methodensignatur:

```
public long getElementId()
```

Methodensignatur (Kurzform):

```
getElementId():long
```

Beispiel (Skript):

```
db = context.getAccessControlDb();

acls = db != null ? db.getAcls() : null;



if (acls != null) {

  for (acl: acls) {

    files = db.getFiles(acl);

    if (files != null) {

      for (file : files) {

        print(file.getElementId());

      }

    }

  }

}
```

Beispiel (JSP-Seite):

```
<%@
  page
  import="de.espirit.firstspirit.acl.db.AccessControlDb,
    de.espirit.firstspirit.acl.servlet.AccessControlServlet,
    java.util.List,
    de.espirit.firstspirit.acl.Acl,
    de.espirit.firstspirit.acl.File" %><%

  final AccessControlDb db =
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);

  final List<Acl> acls = db != null ? db.getAcls() : null;

  if (acls != null) {

    for (final Acl acl: acls) {

      final List<File> files = db.getFiles(acl);
      if (files != null) {
        for (File file : files) {
%><%= file.getElementId() %><br /><%
        }
      }

    }
  }
%>
```

4.1.4.4 Methode "getElementUid()"

Durch die Methode "getElementUid()" wird der eindeutige Bezeichner zurückgeliefert, der bei der Generierung des Objektes gültig war.

Eindeutige Bezeichner sind objektabhängig und nicht projektweit eindeutig, d.h. z.B. Medien und Seitenreferenzen können den gleichen eindeutigen Bezeichner haben.

Font-Images haben keinen eindeutigen Bezeichner. Die Rückgabe in diesem Fall ist "null".

Methodensignatur:

```
public java.lang.String getElementUid()
```

Methodensignatur (Kurzform):

```
getElementUid():String
```

Beispiel (Skript):

```
db = context.getAccessControlDb();

acls = db != null ? db.getAcls() : null;



if (acls != null) {

  for (acl: acls) {

    files = db.getFiles(acl);

    if (files != null) {

      for (file : files) {

        print(file.getElementUid());

      }

    }

  }
```

```
}
```

Beispiel (JSP-Seite):

```
<%@
  page
  import="de.espirit.firstspirit.acl.db.AccessControlDb,
    de.espirit.firstspirit.acl.servlet.AccessControlServlet,
    java.util.List,
    de.espirit.firstspirit.acl.Acl,
    de.espirit.firstspirit.acl.File" %><%

  final AccessControlDb db =
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);

  final List<Acl> acls = db != null ? db.getAcls() : null;

  if (acls != null) {

    for (final Acl acl: acls) {

      final List<File> files = db.getFiles(acl);
      if (files != null) {
        for (File file : files) {
%><%= file.getElementUid() %><br /><%
        }
      }
    }
  }
%>
```

4.1.4.5 Methode "getCrc32()"

Die Methode "getCrc32()" liefert die berechnete CRC-32-Prüfsumme zu einer generierten Datei zurück.

CRC-32-Prüfsummen werden anhand des Dateiinhaltes ermittelt.

Methodensignatur:

```
public long getCrc32()
```

Methodensignatur (Kurzform):

```
getCrc32():long
```

Beispiel (Skript):

```
db = context.getAccessControlDb();

acls = db != null ? db.getAcls() : null;


if (acls != null) {
  for (acl: acls) {
    files = db.getFiles(acl);
    if (files != null) {
      for (file : files) {
        print(file.getCrc32());
      }
    }
  }
}
```

Beispiel (JSP-Seite):

```
<%@
  page
  import="de.espirit.firstspirit.acl.db.AccessControlDb,
    de.espirit.firstspirit.acl.servlet.AccessControlServlet,
    java.util.List,
    de.espirit.firstspirit.acl.Acl,
    de.espirit.firstspirit.acl.File" %><%



  final AccessControlDb db =
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);


  final List<Acl> acls = db != null ? db.getAcls() : null;


  if (acls != null) {


    for (final Acl acl: acls) {


      final List<File> files = db.getFiles(acl);
      if (files != null) {
        for (File file : files) {
%><%= file.getCrc32() %><br /><%
        }
      }
    }
  }
%>
```

4.1.4.6 Methode "getLastModified()"

Die Methode "getLastModified()" liefert das Datum bzw. den Zeitpunkt der letzten Änderung des Objektes (als Zeitstempel) zurück. Die Rückgabe erfolgt als "long" und kann durch Java-Methoden in ein lesbares Datumsformat überführt werden.

Methodensignatur:

```
public long getLastModified()
```

Methodensignatur (Kurzform):

```
getLastModified():long
```

Beispiel (Skript):

```
import java.text.SimpleDateFormat;

import java.util.Locale;


df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.GERMANY);

db = context.getAccessControlDb();


db = context.getAccessControlDb();

acls = db != null ? db.getAcls() : null;


if (acls != null) {

  for (acl: acls) {

    files = db.getFiles(acl);

    if (files != null) {

      for (file : files) {

        print(

          file.getLastModified()

          + " ("

          + df.format(file.getLastModified())

          + ")"

        );
```

```
      }
    }
  }
}
```

Beispiel (JSP-Seite):

```
<%@
  page
  import="java.text.SimpleDateFormat,
    java.util.Locale,
    de.espirit.firstspirit.acl.db.AccessControlDb,
    de.espirit.firstspirit.acl.servlet.AccessControlServlet,
    java.util.List,
    de.espirit.firstspirit.acl.Acl,
    de.espirit.firstspirit.acl.File" %><%

  final SimpleDateFormat df =
    new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.GERMANY);

  final AccessControlDb db =
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);

  final List<Acl> acls = db != null ? db.getAcls() : null;

  if (acls != null) {

    for (final Acl acl: acls) {

      final List<File> files = db.getFiles(acl);
```

```
      if (files != null) {

        for (File file : files) {

%>

  <%= file.getLastModified() %>

  (<%= df.format(file.getLastModified()) %>)

  <br />

<%

        }

      }


    }

  }

%>
```

4.1.4.7 Methode "getLastUpdate()"

Die Methode "getLastUpdate()" liefert das Datum bzw. den Zeitpunkt der letzten Aktualisierung der gespeicherten Daten des Objektes (als Zeitstempel) zurück. Die Rückgabe erfolgt als "long" und kann durch Java-Methoden in ein lesbares Datumsformat überführt werden.

Methodensignatur:

```
public long getLastUpdate()
```

Methodensignatur (Kurzform):

```
getLastUpdate():long
```

Beispiel (Skript):

```
import java.text.SimpleDateFormat;

import java.util.Locale;


df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.GERMANY);

db = context.getAccessControlDb();
```

```
db = context.getAccessControlDb();

acls = db != null ? db.getAcls() : null;


if (acls != null) {

  for (acl: acls) {

    files = db.getFiles(acl);

    if (files != null) {

      for (file : files) {

        print(

          file.getLastUpdate()

          + " ("

          + df.format(file.getLastUpdate())

          + ")"

        );

      }

    }

  }

}
```

Beispiel (JSP-Seite):

```
<%@

  page

  import="java.text.SimpleDateFormat,

    java.util.Locale,

    de.espirit.firstspirit.acl.db.AccessControlDb,

    de.espirit.firstspirit.acl.servlet.AccessControlServlet,

    java.util.List,

    de.espirit.firstspirit.acl.Acl,

    de.espirit.firstspirit.acl.File" %><%
```

```
  final SimpleDateFormat df =
    new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.GERMANY);

  final AccessControlDb db =
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);

  final List<Acl> acls = db != null ? db.getAcls() : null;

  if (acls != null) {

    for (final Acl acl: acls) {

      final List<File> files = db.getFiles(acl);
      if (files != null) {
        for (File file : files) {
%>
  <%= file.getLastUpdate() %>
  (<%= df.format(file.getLastUpdate()) %>)
  <br />
<%
        }
      }
    }
  }

%>
```

4.1.4.8 Methode "getAcl()"

Mit der Methode "getAcl()" kann die ID der ACL ermittelt werden, zu der die Datei bzw. das Objekt gehört.

Die ID kann dann genutzt werden, um die Berechtigungsinformationen zur Datei zu ermitteln.

Methodensignatur:

```
public long getAcl()
```

Methodensignatur (Kurzform):

```
getAcl():long
```

Beispiel (Skript):

```
db = context.getAccessControlDb();

acls = db != null ? db.getAcls() : null;


if (acls != null) {

  for (acl: acls) {

    files = db.getFiles(acl);

    if (files != null) {

      for (file : files) {

        print(

          file.getPath()

          + ": "

          + db.getAcl(file.getAcl()).getActivities()

        );

      }

    }

  }

}
```

Beispiel (JSP-Seite):

```
<%@
  page
  import=" de.espirit.firstspirit.acl.db.AccessControlDb,
    de.espirit.firstspirit.acl.servlet.AccessControlServlet,
    java.util.List,
    de.espirit.firstspirit.acl.Acl,
    de.espirit.firstspirit.acl.File" %><%

  final AccessControlDb db =
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);

  final List<Acl> acls = db != null ? db.getAcls() : null;

  if (acls != null) {

    for (final Acl acl: acls) {

      final List<File> files = db.getFiles(acl);
      if (files != null) {
        for (File file : files) {
%>
  <%= file.getPath() %>:
  <%= db.getAcl(file.getAcl()).getActivities() %>
  <br />
<%
        }
      }
```

```
    }

  }

%>
```

4.1.4.9 Methode "getLength()"

Die Methode "getLength()" liefert die Größe einer Datei in Bytes zurück.

Methodensignatur:

```
public long getLength()
```

Methodensignatur (Kurzform):

```
getLength():long
```

Beispiel (Skript):

```
db = context.getAccessControlDb();

acls = db != null ? db.getAcls() : null;


if (acls != null) {

  for (acl: acls) {

    files = db.getFiles(acl);

    if (files != null) {

      for (file : files) {

        print(file.getLength());

      }

    }

  }

}
```

Beispiel (JSP-Seite):

```
<%@
  page
  import="de.espirit.firstspirit.acl.db.AccessControlDb,
    de.espirit.firstspirit.acl.servlet.AccessControlServlet,
    java.util.List,
    de.espirit.firstspirit.acl.Acl,
    de.espirit.firstspirit.acl.File" %><%

  final AccessControlDb db =
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);

  final List<Acl> acls = db != null ? db.getAcls() : null;

  if (acls != null) {

    for (final Acl acl: acls) {
      final List<File> files = db.getFiles(acl);
      if (files != null) {
        for (File file : files) {
%><%= file.getLength() %><br /><%
        }
      }
    }
  }
%>
```

4.1.4.10 Methode "getElementTag()"

Mit der Methode "getElementTag()" kann der Tag-Name einer Datei bzw. eines Objektes ermittelt werden. Diese Methode ist wichtig, um Objekte voneinander unterscheiden zu können, da z.B. die UID einer Seitenreferenz und eines Medium identisch sein können.

Methodensignatur:

```
public String getElementTag()
```

Methodensignatur (Kurzform):

```
getElementTag():String
```

Beispiel (Skript):

```
db = context.getAccessControlDb();

acls = db != null ? db.getAcls() : null;



if (acls != null) {

  for (acl: acls) {

    files = db.getFiles(acl);

    if (files != null) {

      for (file : files) {

        print(file.getElementTag());

      }

    }

  }

}
```

Beispiel (JSP-Seite):

```
<%@
  page
  import="de.espirit.firstspirit.acl.db.AccessControlDb,
    de.espirit.firstspirit.acl.servlet.AccessControlServlet,
    java.util.List,
    de.espirit.firstspirit.acl.Acl,
    de.espirit.firstspirit.acl.File" %><%

  final AccessControlDb db =
    (AccessControlDb) application
      .getAttribute(AccessControlServlet.ACCESS_CONTROL_DB);

  final List<Acl> acls = db != null ? db.getAcls() : null;

  if (acls != null) {

    for (final Acl acl: acls) {
      final List<File> files = db.getFiles(acl);
      if (files != null) {
        for (File file : files) {
%><%= file.getElementTag() %><br /><%
        }
      }
    }
  }
%>
```

5 CRC-Transfer-Servlet

5.1 Listener registrieren bzw. deregistrieren

Das CRC-Transfer-Servlet bietet zwei öffentliche Methoden an, um Listener zu registrieren bzw. zu deregistrieren:

```
public void addTransferListener(final TransferListener transferListener);
public void removeTransferListener(final TransferListener transferListener);
```

Das CRC-Transfer-Servlet registriert sich im ServletContext und ist folgendermaßen zu erreichen:

```
final CRCTransferServlet crcServlet = (CRCTransferServlet)
servletContext.getAttribute(CRCTransferServlet.CRC_TRANSFER_SERVLET);
```

Ein TransferListener wird vor und nach jeder Dateioperation, die das CrcServlet durchführt informiert. Dabei blockieren die Listener-Aufrufe den aktuellen Servlet-Aufruf. Langandauernde Operationen sollten also nebenläufig in einem Thread ausgeführt werden, da es zu einem Timeout auf Clientseite kommt, wenn der Request zu lange blockiert wird.

Im nachfolgenden Kapitel werden die Klassen bzw. Interfaces vorgestellt, die für die Implementierung eines Listeners von Interesse sind (siehe Kapitel 5.2 Seite 76).

5.2 Implementierung eines Listeners

Für die Implementierung von Listenern sind folgende Klassen/Interfaces verfügbar:

- Interface "de.espirit.firstspirit.client.io.crc.TransferListener":
- Klasse "de.espirit.firstspirit.client.io.crc.AbstractTransferListener"
- Interface "de.espirit.firstspirit.client.io.crc.CRCSession"

5.3 Beispiel

Die Beispiel-JSP-Seite muss in derselben Web-Applikation wie das CrcServlet laufen und implementiert einen einfachen Listener, der alle Calls auf System.out ausgibt:

Parameter der JSP-Seite:

- TransferListenerTest.jsp: CrcServlet und Listener ausgeben
- TransferListenerTest.jsp?action=add Listener registrieren
- TransferListenerTest.jsp?action=remove Listener deregistrieren.

```
<%@ page import="de.espirit.firstspirit.client.io.crc.*, java.util.Date"



%><pre><%



final CRCTransferServlet crcServlet = (CRCTransferServlet)
pageContext.getServletContext().getAttribute(CRCTransferServlet.CRC_TRANS
FER_SERVLET);

   final String action = request.getParameter("action");

   if ("add".equals(action)) {

          final TransferListener listener = new
AbstractTransferListener() {



public void beforeListFiles(final CRCSession session, final String path)
{

                        System.out.println("JspClass.beforeListFiles");

                        System.out.println("  path=" + path);

                        dumpCRCSession(session);

                }



public void beforeGetFile(final CRCSession session, final String path) {

                        System.out.println("JspClass.beforeGetFile");

                        System.out.println("  path=" + path);

                        dumpCRCSession(session);

                }



public void beforeUpload(final CRCSession session, final String path,
final boolean append) {

                        System.out.println("JspClass.beforeUpload");
```

```
                        System.out.println("  path=" + path);
                        System.out.println("  append=" + append);
                        dumpCRCSession(session);
                }

public void beforeDownload(final CRCSession session, final String path) {
                        System.out.println("JspClass.beforeDownload");
                        System.out.println("  path=" + path);
                        dumpCRCSession(session);
                }

public void beforeDelete(final CRCSession session, final String path) {
                        System.out.println("JspClass.beforeDelete");
                        System.out.println("  path=" + path);
                        dumpCRCSession(session);
                }

public void beforeExists(final CRCSession session, final String path) {
                        System.out.println("JspClass.beforeExists");
                        System.out.println("  path=" + path);
                        dumpCRCSession(session);
                }

public void beforeRename(final CRCSession session, final String oldPath,
final String newPath) {
                        System.out.println("JspClass.beforeRename");
                        System.out.println("  oldPath=" + oldPath);
                        System.out.println("  newPath=" + newPath);
                        dumpCRCSession(session);
                }
```

```
public void beforeTouch(final CRCSession session, final String path,
final long time) {

                        System.out.println("JspClass.beforeTouch");

                        System.out.println("  path=" + path);

                        System.out.println("  time=" + time);

                        dumpCRCSession(session);

                }


public void beforeSwitch(final CRCSession session, final String oldPath,
final String newPath) {

                        System.out.println("JspClass.beforeSwitch");

                        System.out.println("  oldPath=" + oldPath);

                        System.out.println("  newPath=" + newPath);

                        dumpCRCSession(session);

                }


public void beforeMkdir(final CRCSession session, final String path) {

                        System.out.println("JspClass.beforeMkdir");

                        System.out.println("  path=" + path);

                        dumpCRCSession(session);

                }


public void afterListFiles(final CRCSession session, final String path) {

                        System.out.println("JspClass.afterListFiles");

                        System.out.println("  path=" + path);

                        dumpCRCSession(session);

                }


public void afterGetFile(final CRCSession session, final String path) {
```

```
                        System.out.println("JspClass.afterGetFile");

                        System.out.println("  path=" + path);

                        dumpCRCSession(session);

                }


public void afterUpload(final CRCSession session, final String path,
final boolean append) {

                        System.out.println("JspClass.afterUpload");

                        System.out.println("  path=" + path);

                        System.out.println("  append=" + append);

                        dumpCRCSession(session);

                }


public void afterDownload(final CRCSession session, final String path) {

                        System.out.println("JspClass.afterDownload");

                        System.out.println("  path=" + path);

                        dumpCRCSession(session);

                }


public void afterDelete(final CRCSession session, final String path) {

                        System.out.println("JspClass.afterDelete");

                        System.out.println("  path=" + path);

                        dumpCRCSession(session);

                }


public void afterExists(final CRCSession session, final String path) {

                        System.out.println("JspClass.afterExists");

                        System.out.println("  path=" + path);

                        dumpCRCSession(session);

                }
```

```
public void afterRename(final CRCSession session, final String oldPath,
final String newPath) {

                        System.out.println("JspClass.afterRename");

                        System.out.println("  oldPath=" + oldPath);

                        System.out.println("  newPath=" + newPath);

                        dumpCRCSession(session);

                }


public void afterTouch(final CRCSession session, final String path, final
long time) {

                        System.out.println("JspClass.afterTouch");

                        System.out.println("  path=" + path);

                        System.out.println("  time=" + time);

                        dumpCRCSession(session);

                }


public void afterSwitch(final CRCSession session, final String oldPath,
final String newPath) {

                        System.out.println("JspClass.afterSwitch");

                        System.out.println("  oldPath=" + oldPath);

                        System.out.println("  newPath=" + newPath);

                        dumpCRCSession(session);

                }


public void afterMkdir(final CRCSession session, final String path) {

                        System.out.println("JspClass.afterMkdir");

                        System.out.println("  path=" + path);

                        dumpCRCSession(session);

                }
```

```
private void dumpCRCSession(final CRCSession session) {

System.out.println("  user=" + session.getUser());

System.out.println("  remoteAddr=" + session.getLastRemoteAddr());

System.out.println("  creationTime=" + new
Date(session.getCreationTime()));

System.out.println("  lastUsageTime=" + new
Date(session.getLastUsageTime()));

                }

   };

crcServlet.addTransferListener(listener);

final TransferListener oldListener = (TransferListener)
session.getAttribute("TransferListener");



if (oldListener != null) {

   crcServlet.removeTransferListener(oldListener);

   }

   session.setAttribute("TransferListener", listener);

   } else if ("remove".equals(action)) {

   final TransferListener listener = (TransferListener)
   session.getAttribute("TransferListener");



if (listener != null) {

   crcServlet.removeTransferListener(listener);

   session.removeAttribute("TransferListener");

   }

}



out.println("CRCTransferServlet: " + crcServlet);

out.println("TransferListener: " +
session.getAttribute("TransferListener"));

%></pre>
```

6 Rechtliche Hinweise

Das Modul "FirstSpirit™ Security" 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".