Benutzerhandbuch

„Die Menschen heute
sind nicht mehr verwurzelt,
sondern vernetzt.“
Anna Heine

Inhaltsverzeichnis

Kapitel 1

Einführung

JUpload ist eine Anwendung (genauer: Java Applet) für den Upload von lokalen Dateien vom Benutzer (Client) zu einem Webserver (Server) aus Basis des Standardprotokolls HTTP.

Die Applikation ersetzt eine von den heutigen Browser nur in geringem Maße umgesetzte Funktion und erweitert dadurch die Funktionsvielfalt der unterstützten Browser. Hauptsächlich handelt es sich dabei um die Möglichkeit, mehrere ausgewählte Resourcen innerhalb eines Arbeitsschrittes hochzuladen. Nebenbei wird die Benutzerfreundlichkeit von Webanwendungen erhöht, so z.B. durch die automatische Bildvorschau oder die Statusanzeige während der Datenübertragung (die nur in sehr wenigen speziellen Browser zum Lieferumfang gehört).

Desweiteren ist JUpload so gestaltet, dass auch die sichere Datenübertragung durch Verschlüsselung auf Basis von SSL möglich ist.

Eine für viele Benutzer sehr angenehme Funktion könnte die Resume-Funktion sein, allerdings muss diese auch serverseitig unterstützt werden. Eine Neuimplementierung von Serveranwendungen ist dabei nicht notwendig, jedoch eine Anpassung der eigentlichen Teilprogramme, die serverseitig für die entgegennahme der Daten zuständig ist.

Dieses Handbuch gibt Ihnen einen Einblick in die Installation und Konfiguration (für Administratoren und Webentwickler siehe Administratorenhandbuch), die Benutzung des Applets (Benutzerhandbuch). Entwickler finden im Entwicklerhandbuch alle notwendigen Informationen in Bezug auf die Sourcen um grundlegende Veränderungen am Applet vornehmen zu können.

Kapitel 2

Benutzerhandbuch

Dieser Abschnitt erklärt die Bedienung von JUpload. Lesen Sie hier über die Funktionsweise von JUpload, über dessen Bedienung und welche Reaktionen Sie von der Anwendung erwarten können.

Durch die große Vielfalt an Einstellungsmöglichkeiten kann allerdings nicht auf jede Eventualität eingegangen werden - bei Problem kontaktieren Sie bitte zuerst Ihren Systemadministrator bzw. den Anbieter der JUpload-Anwendung.

2.1 Programmbedienung

2.1.1 Starten

Ist das Produkt auf dem Webserver korrekt installiert, wird es automatisch beim Aufrufen der entsprechenden URL vom installierten Java Plug-In aufgerufen. Während dem Laden erscheint eine Statusanzeige.

Ist das Applet vollständig geladen, erscheint ein Dialog über das verwendete Zertifikat. Bitte prüfen Sie die Angaben im Zertifikat auf Ihre Richtigkeit und akzeptieren Sie es (Klicken Sie auf Ja).

2.1.2 Übersicht

Ist die Anwendung vollständig geladen und das Zertifikat akzeptiert, erscheint die grafische Benutzeroberfläche von JUpload. Auf der linken Seite in der oberen Hälfte sehen Sie die **Dateiliste**. Im unteren Bereich sehen Sie evtl. den Anzeigebereich für die **Echtzeitantwort** des Servers. Diese ist nur Sichtbar, wenn die entsprechende Funktion freigeschalten[1] ist.

Auf der rechten Seite ist der Bereich für die Funktionsknöpfe und Statusanzeigen angeordnet. Die drei Funktionsknöpfe **Hinzufügen**, **Entfernen** und **Hochladen** dienen zur Steuerung der Anwendung.

2.1.3 Dateien auswählen

Um Dateien für das Hochladen vorzubereiten, müssen Sie ausgewählt werden. Ausgewählte Dateien erscheinen in der grafischen Oberfläche von JUpload auf der linken

[1] siehe Administratorenhandbuch: showRealTimeResponse

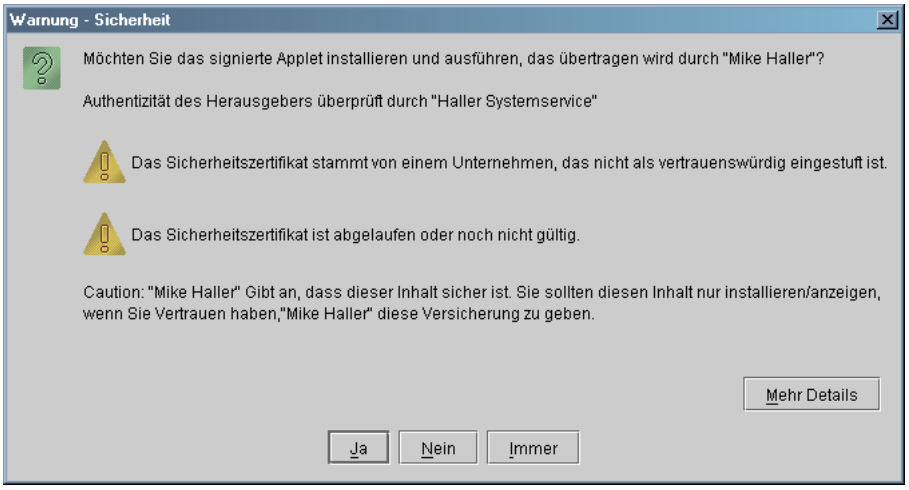

Abbildung 2.1: Zertifikatsdetails

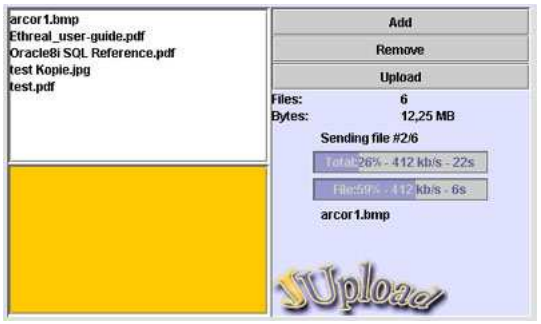

Abbildung 2.2: JUpload Hauptoberfläche

Seite in der **Dateiliste**. Dateien werden dort nur mit ihrem kurzen Dateinamen angezeigt, nicht mit der vollständigen Dateipfadangabe. Dies ist jedoch über das jeweilige Tooltip zugänglich. Fahren Sie mit der Maus hierzu auf eine Datei und warten Sie kurz - es erscheint ein Schwebekasten mit der vollständigen Pfadangabe derjenigen Datei, die sich unter dem Mauscursor befindet. Zum Hinzufügen von Dateien in die Dateiliste klicken Sie bitte auf den Knopf mit der Beschriftung **Hinzufügen**. Es wird ein vom Betriebssystem größtenteils vorgegebener Dateiauswahldialog angezeigt. Für Details zur Bedienung des Dateiauswahldialogs siehe Handbuch Ihres Betriebssystems.

JUpload bietet eine Bildvorschaufunktions an. Wird im Dateiauswahldialog eine Bilddatei ausgewählt, deren Bildformat unterstützt wird, kann das Vorschaubild auf der rechten Seite betrachtet werden. Ist Java > 1.4 installiert, kann mit Hilfe des Mausrads in das Bild hinein- und herausgezoomt werden. Wählen Sie eine oder mehrere Dateien mit einem Klick der linken Maustaste aus und drücken Sie den Knopf **Hinzufügen** des Dateiauswahldialogs. Durch gleichzeitiges gedrückt halten der Umschalttaste kann ein Dateibereich ausgewählt werden. Durch gleichzeitig gedrückte Steuerungstaste können mehrere, nicht zwingend aufeinanderfolgende Dateien, ausgewählt werden. Um alle Dateien und Ordner (Achtung!) auszuwählen, drücken Sie **STRG + A** (Alles auswählen). Ausgewählte bzw. selektierte Dateien werden farblich hervorgehoben. Wird

der Ausnahmedialog mit **Abbrechen** beendet oder über Schließen geschlossen, werden keine Dateien in die Dateiliste übernommen.

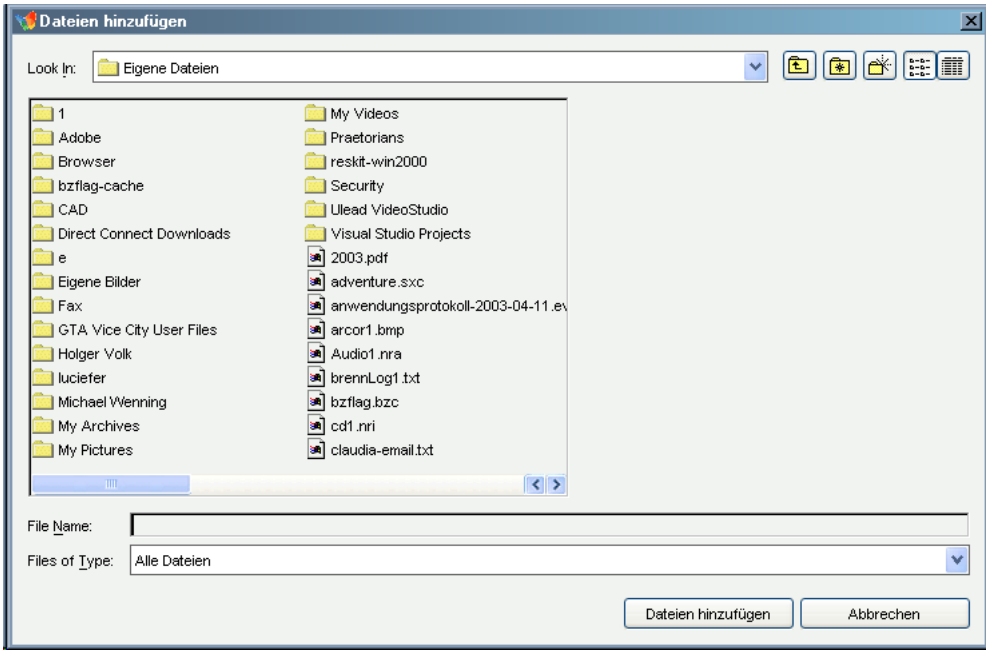

Abbildung 2.3: Dateiauswahldialog mit benutzerdefiniertem SkinLF Look And Feel

Drag'n'Drop

Mit Drag'n'Drop können auch Dateien vom Dateisystem-Explorer direkt in die Warteschlange von JUpload gezogen werden.

Klicken Sie hierzu auf die hochzuladende Datei im Explorer. Bei gedrückter Maustaste ziehen Sie nun die Datei in den weissen Bereich links oben im Applet. Dort angekommen lassen Sie die Maustaste los, die Datei sollte nun in die List aufgenommen sein und der Dateiname angezeigt werden.

Ein Herausziehen der Dateien aus der Liste (zum Entfernen) wird nicht unterstützt.

Dateiliste

Die Liste der Dateien wird verkürzt dargestellt, d.h. ohne die vollständigen Pfadangaben. Sollten Sie Informationen über die Pfadangaben in der Liste benötigen, fahren Sie mir der maus über die entsprechende Datei.

Nach ein oder zwei Sekunden erscheint ein sog. Tooltip, der Ihnen den vollständigen Pfad zu dieser Datei auf Ihrer Festplatte anzeigt.

2.1.4 Dateien entfernen

Befinden Sich in der Dateiliste Einträge, die Sie nicht hochladen möchten, selektieren Sie diese mit der linken Maustaste. Auch hier gilt: gedrückt halten der Umschalttaste ermöglicht die Selektierung mehrerer aufeinanderfolgender Dateien, gedrückt halten

der Steuerungstaste ermöglicht die Selektierung einzelner, nicht zwingend aufeinanderfolgender Dateien. Alle Einträge werden durch drücken von STRG-A ausgewählt. Haben Sie die ungewünschten Dateien ausgewählt, drücken Sie auf den Knopf **Entfernen**, und die Einträge werden entfernt. Die Dateien im Dateisystem werden nicht gelöscht!

Abbildung 2.4: Dateiwarteschlange

Ein Herausziehen der Dateien per Drag'n'Drop ist nicht möglich, jedoch das Hereinziehen aus dem Windows Explorer (oder andere Drag'n'Drop-unterstützende Programme).

2.1.5 Hochladen

Wenn die Auswahl der Dateien abgeschlossen ist, können Sie nun die Dateiübertragung beginnen lassen. Drücken Sie hierzu auf den Knopf **Hochladen**. Der Upload beginnt nun und kann nicht mehr gestoppt werden.

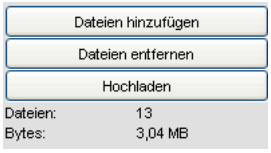

Abbildung 2.5: Buttons und Anzeige der Dateigrößen in der Warteschlange

Ein abgebrochener Upload kann nicht an gleicher Stelle fortgeführt werden, wenn mit POST gearbeitet wird (default). Ist JUpload und der Server allerdings für PUT eingerichtet, so ist ein Uploadresuming möglich.

Um einen abgebrochenen Upload forzuführen, starten Sie JUpload nochmals wie gewohnt und selektieren einfach die gleichen Dateien nochmals. Der Upload fährt automatisch dort fort, wo er abgebrochen worden ist.

Dieses Feature muss vom Server unterstützt werden. Wenn Sie Fragen dazu haben, kontaktieren Sie bitte Ihren Systemadministrator.

2.1.6 Statusanzeige

Während des Hochladens ist eine Statusanzeige auf der rechten Seite zu sehen. Diese zeigt den aktuellen Protokollstatus an („Server suchen", „Verbindung aufbauen" etc.), sie zeigt die aktuelle Datei und den Verlauf in Prozent an. Innerhalb des Prozentbalkens kann die geschätzte Übertragungsgeschwindigkeit und zu verbleibende Zeit abgelesen

werden. Es wird noch der Dateiname der aktuellen Datei angezeigt und ein zweiter Prozentbalken, der den Gesamtprozess betrifft (alle Dateien).

Abbildung 2.6: Buttons, Statusmeldung und Prozentanzeige der laufenden Übertragung

Die Statusanzeige zeigt eine Statusnachricht an, die den Zustand der Anwendung ausgibt. Diese Zustände zeigen an, ob die Datenübertragung noch ausgehandelt wird, oder ob bereits Daten übertragen werden.

Außerdem zeigt Sie die Übertragunsgeschwindigkeit und die geschätzte Fertigstellungszeit an, sowie natürlich einen prozentualen Fortschrittsbalken.

Der obere Balken zeigt den Fortschritt für die Datei an, die gerade gesendet wird - der untere zeigt den Gesamtfortschritt an. Unterhalb der Balken wird der Dateiname der aktuellen Datei und ihre Warteschlangennummer angezeigt.

Sollte der Dateiname zu gross für die Darstellung werden, wird er abgeschnitten. Möchten Sie dennoch die Informationen sehen, können Sie den vertikalen Begrenzungsbalken bei gedrückter Maustaste weiter nach links verschieben.

2.1.7 Bildvorschau

Die Bildvorschau ermöglicht das komfortable Auswählen der Hochzuladenden Dateien. Als unterstützte Bilddatentypen gelten die von Java unterstützten Formate: GIF, JPEG, PNG.

Ausserdem lässt sich mit der Bildvorschau in das Bild hinein- und herauszoomen. Nutzen Sie dafür das Mausrad. Diese Funktion steht erst ab Java 1.4 zur Verfügung.

Die Bildvorschau zeigt außerdem die Dimension der Bilddatei (Höhe und Breite) in Pixel an, sowie deren Dateigröße in Bytes.

2.1.8 Rekursive Auswahl

Werden im Dateiauswahldialog Verzeichnisse (Ordner) ausgewählt, werden diese rekursiv bearbeitet. Das bedeutet, dass alle unterhalb dieses Ordners zu findenen Dateien und Ordner durchsucht werden. Alle gefundenen Dateien werden in die Dateiliste übernommen. Diese Funktion ist als standardmäßíg aktiviert und kann vom Administrator deaktiviert werden.

Außerdem kann vom Administrator eingestellt werden, ob die Pfadangaben mit übertragen werden oder nicht. Viele Server erkennen allerdings Pfadangaben und entfernen diese automatisch aus Sicherheitsgründen. Sollten die Pfadangaben für Ihre Anwendung jedoch benötigt werden, ist es JUpload möglich die Pfadangaben zu verschleiern, sodass der Webserver diese nicht entfernen kann.

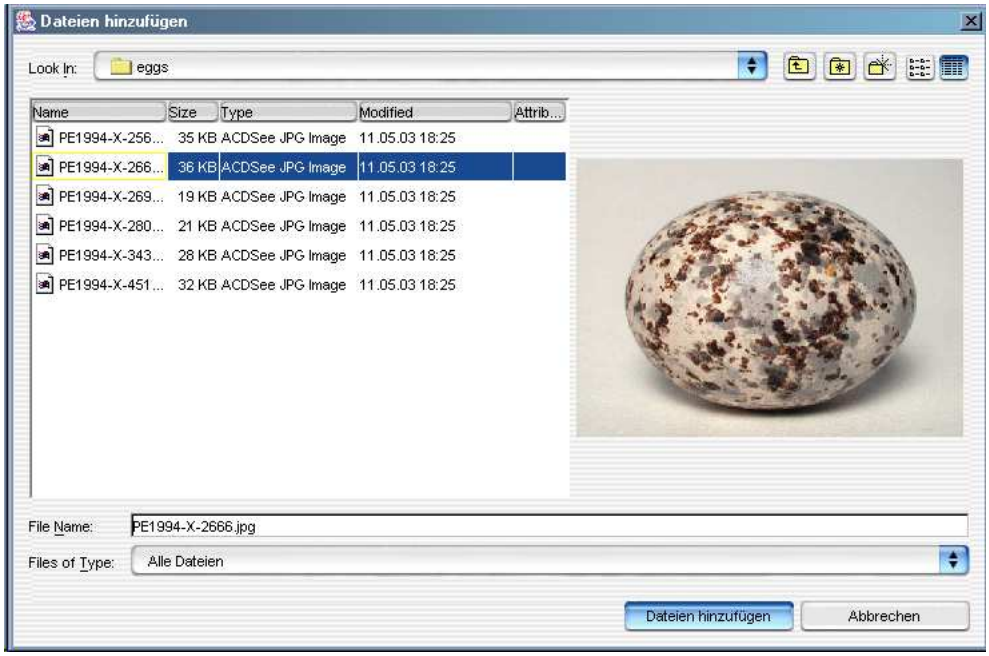

Abbildung 2.7: Dateiauswahldialog im XP-Look, rechts die Bildvorschau

2.1.9 Dateifilter

Die Dateifilter des Dateiauswahldialogs dienen zum verstecken von nicht gewünschten Dateien. Ist ein bestimmter Dateifilter, z.B. ein Bilddateifilter voreingestellt, werden beim Aufrufen des Dateiauswahldialogs nur noch Dateien angezeigt, die dem Filter entsprechen. In diesem Fall also nur noch GIF, JPEG etc. Bilddateien.

Der Dateifilter kann vom Administrator vorgegeben werden und eigene Filter definieren.

2.1.10 Fehlermeldungen

Fehlermeldungen seitens JUpload werden in Dialogfenstern angezeigt. Es gibt verschiedene Arten von Fehlermeldungen. Die Fehlermeldungen vor dem Upload beziehen sich meist auf das Überschreiten von Grenzen oder auf Bedienfehler. Fehlermeldungen nach dem Upload beziehen sich meist auf Serverseitige Fehlermeldungen, d.h. die Verbindung zum Server konnte nicht aufgebaut werden, oder der Server meldete einen Verarbeitungsfehler etc.

Das Anzeigen der Fehlermeldungen kann vom Administrator unterdrückt werden.

2.2 Hilfe

Zusätzlich zu diesem vollständigen Handbuch erhalten Sie Support zu diesem Produkt auf der Produktseite des Herstellers.

http://www.haller-systemservice.net/jupload/

2.2.1 Online-Hilfe

Aus Platzgründen ist keine Online-Hilfe im Applet integriert.

Im Programm selbst gibt es eine dynamisch generierte Liste aller unterstützter Konfigurationsparameter. Klicken Sie hierzu mit der rechten Maustaste im rechten Panel auf den Hintergrund. Der About-Eintrag zeigt die Lizenzbestimmungen an, der Eintrag zum Applet-Tag zeigt ein komplettes Applet-Tag-Beispiel mit allen verwendbaren Parametereinstellungen samt ihrer aktuellen Default-Einstellungen an.

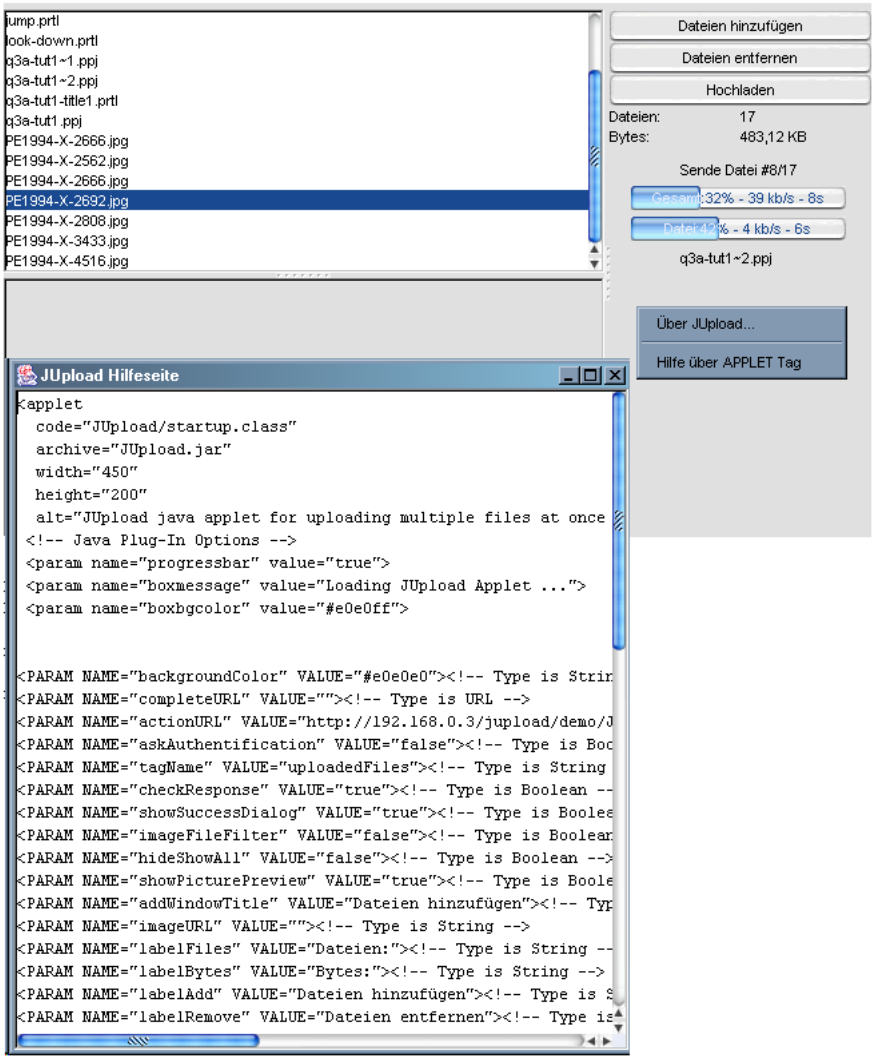

Abbildung 2.8: Applet während des Uploads mit geöffnetem Hilfetext

Kapitel 3

Administratorenhandbuch

Das Administratorhandbuch behandelt Themen zur einmaligen Installation und Konfiguration des Applets auf dem Server.

Die Benutzung des Applets ist im Abschnitt Benutzerhandbuch beschrieben.

3.1 Installation

In den folgenden Abschnitten werden die Systemvoraussetzungen geklärt und die Vorgehensweise zur Installation und Konfiguration abgehandelt.

3.1.1 Systemvoraussetzungen

Serverseitig: JUpload hat serverseitig kaum Voraussetzungen. Diese beziehen sich lediglich darauf, dass Binärdaten vom Server an den Client übertragen werden können - dies wird von jedem Webserver unterstützt.

Jedoch ist zu beachten, dass einige Servererweiterungen nicht oder nur schlecht mit den eingesetzten Standardprotokollen umgehen können (siehe ältere ASP und PHP Versionen). Werden aktuelle Servererweiterungen (z.B. mindestens PHP 4.2) eingesetzt, sollte es keine Probleme geben.

Clientseitig: Auf der Clientseite wird *Java 1.4* vorausgesetzt. Bei dieser Version wird das benutzte Java Plug-In 1.4 bereits mitinstalliert - die Browser verwendet das Plug-In für das Ausführen von Java Applets. Das Produkt kann auch unter Java 1.3 eingesetzt werden, dies ist aber nicht getestet.

Eine Nutzung der Anwendung mit *Microsoft Virtual Machine ist nicht möglich*. Der Grund hierfür liegt in der veralteten Technologie der MS-JVM (diese basiert auf Java 1.1). Es wird ausserdem ein Browser benötigt, der mit signierten Applets umgehen kann. Getestet wurde mit Internet Explorer, Mozilla, Opera. Konqueror sollte nicht eingesetzt werden.

3.1.2 Installation

Entpacken Sie das Archiv (`jupload.zip`) in ein Verzeichnis unterhalb der `documentroot` des Webservers.

Wichtig ist die Datei `JUpload.jar`, welche die eigentliche, signierte Applikation enthält.

3.2 Konfiguration

Das Applet muss nun an die Anforderungen Ihrer Webanwendung angepasst werden. Zum leichten Einstieg ist im Archiv eine Beispielanwendung JUpload.php enthalten. Stellen Sie bitte vorher sicher, dass PHP richtig installiert ist und funktionstüchtig ist.

Stellen Sie ausserdem sicher, dass der normale Dateiupload funktioniert. Erzeugen Sie dazu eine einfache HTML-Datei mit folgendem Inhalt:

Beispiel:

```
<html>
 <form method="POST"
       action="JUpload.php"
       enctype="application/form-data">

  <input type=file>

  <input type=submit>

 </form>
</html>
```

Ergebnis auf der HTML-Seite:

Abbildung 3.1: Ein Standard-Dateielement für Uploads

Starten Sie den Browser und öffnen Sie die HTML-Datei über die Adresse Ihres Webservers. öffnen Sie die Datei nicht über das Dateisystem. Um sicherzugehen sollten Sie die Seite von einem anderen Computer aus aufrufen.

Wählen Sie eine lesbare Datei von Ihrer lokalen Festplatte aus und klicken Sie auf submit. Wird die Datei hochgeladen und auf dem Server im Upload-Verzeichnis (siehe `JUpload.php`) auffindbar, hat der Upload funktioniet und sie können JUpload konfigurieren.

Die wichtigste und gleichzeitig als einzige notwendige Einstellung müssen Sie nun den Parameter actionURL eintragen. Lassen Sie diesen auf die gleiche URL zeigen, wie sie in der HTML Datei als action eingetragen haben.

Beispiel:

```
<applet ...>

 <param name="actionURL"
        value="http://localhost/JUpload.php">

</applet>
```

Achten Sie bei den Adressen auf die richtige Gross- und Kleinschreibung!

Das Applet-Tag kann aus den Beispieldateien aus dem Archiv übernommen werden. Im Produktiveinsatz sollte es allerdings durch eine Kombination aus `OBJECT` und `EMBED` Tags ersetzt werden - das APPLET-Tag ist inzwischen veraltet. Mit Hilfe der `OBJECT` und `EMBED` Tags kann ausserdem eine automatische Installation des notwendigen Java Plug-Ins vorgenommen werden.

Das folgende `APPLET`-Tag dient zur Orientierung:

```
<applet
  code="JUpload/startup.class"
  archive="JUpload.jar"
  width="650"
  height="400">

  <param name="actionURL"
         value="JUpload.php">

</applet>
```

applet Das Applet-Tag teilt dem Browser mit, das an dieser Stelle im HTML-Dokument ein Java Applet eingebunden werden soll. Als Attribute werden hier alle notwendigen Informationen eingetragen, um das Applet für den Browser auffindbar zu machen.

code Dieses Attribut gibt den Namen der Startklasse an. Die Startklasse befindet sich im Package JUpload und muss mitangegeben werden. Die *startup*-Klasse prüft zunächst die im Browser installierte Java-Version und erzeugt dann das eigentliche JUpload-Objekt.

archive Aus Gründen der übersichtlichkeit ist die komplette Anwendung in ein Archiv eingepackt (JUpload.jar). Dieses Attribut muss angegeben werden (ansonsten würde der Browser die Datei startup.class direkt anfordern). Ausserdem wird das JAR-Archiv für die Signierung benötigt.

width Diese Eigenschaft gibt die Breite des Applets in Pixel an. Ein Applet kann seine Grösse später während der Laufzeit nicht mehr verändern - sie ist fest vorgegeben.

height Gibt die Höhe des Applets in Pixel an.

param Das Param-Tag dient zur Übergabe von Parametern (Argumenten) an die JUpload Anwendung. Jedem Parameter wird ein Name (Schlüsselname, Feldname) und ein Wert zugewiesen. Beide sollten mit Quotes angegeben werden.

Parameternamen dürfen keine Leerzeichen enthalten und müssen den entsprechenden Konventionen genügen. Es ist nicht möglich, mehrere Parameter mit dem gleichen Namen zu übergeben.

actionURL Gibt die Adresse der serverseitigen Anwendung an, welche Uploaddaten entgegennimmt. Adressen können meist absolut oder relativ zur sog. Codebase angegeben werden. Die Codebase ist die Adresse des aktuellen HTML-Dokument welches im Browser dargestellt wird und das Applet beinhaltet.

3.2.1 Allgemeines zu Parametern

Eine Liste aller unterstützen Parameter finden Sie in einem späteren Abschnitt.

Für alle Parameter ist in JUpload ein Standardwert (Default) vorgegeben. Wird ein Parameter nicht explizit durch den Administrator im Applet-Tag angegeben, werden intern die vorgegebenen Standardwerte benutzt. Diese Werte können je nach Versionsstand unterschiedlich sein.

Es gibt verschiedene Typen von Parametern:

URL Eine URL gibt eine Webadresse an. Sie kann meist relativ oder absolut angegeben werden. Beispiele:

```
http://localhost/jupload.php
jupload.php
/myapplication/upload.asp
http://other.server:88/apps/upload.php
```

Boolean Parameter des Typ Boolean sind an/aus Werte. Sie können mit folgenden Werten definiert werden: **true, false, on, off, 1, 0, yes, no**

String Parameter des Typ String gibt eine beliebige Zeichenkette an. Wird meist für Meldungen oder Buttontexte genutzt.

Integer Parameter des Typ Integer sind Zahlangaben. Diese werden häufig für Grössenangaben in Byte genutzt.

Oft kann durch die Angabe von -1 das entsprechende Feature deaktiviert werden, ohne eine Limitangabe aufgehoben werden.

3.2.2 Erweiterte Konfiguration

Eine erweiterte Konfiguration wird über verschiedene Parameter erreicht. Die Auflistung und Beschreibung der einzelnen Parameter finden Sie in einem anderen Abschnitt.

Internationalisierung

JUpload kann durch die Verwendung von java properties-Dateien sehr einfach an verschiedene Sprachen angepasst werden. Die dazu notwendigen Resourcendateien befinden sich im jar-Archive im Verzeichnis `JUpload/`. Der Dateiname folgt der Formatierung

```
jupload_xx_YY.properties
```

Sprache (xx) und Land (YY) folgen den international standardisierten Codes, zu finden unter folgenden Adressen:

Sprachcodes: http://ftp.ics.uci.edu/pub/ietf/http/related/iso639.txt

Ländercodes: http://userpage.chemie.fu-berlin.de/diverse/doc/ISO_3166.html bzw. http://www.iso.org/iso/en/prods-services/iso3166ma/index.html

Die Dateien sind einfache ASCII-Dateien, jede Zeile besteht aus einem Schlüsselwert (Zusammengesetzt aus Klassenname und fortlaufender Nummer), einem Zuweisungsoperator und dem übersetzten Wert. Meistens können Formatierungssequenzen (n etc.) verwendet werden. Das Applet versucht beim Starten automatisch die clientseitige Locale ausfindig zu machen und dann das entsprechende Sprachfile zu laden. Wird kein passendes Sprachfile gefunden, lädt es automatisch die Defaulteinstellungen in *jupload.properties*

Ausschnitt eines übersetzten Beispiels:

```
...
HTTPPostRequest.6=Anfrage vorbereiten...
HTTPPostRequest.8=SecureHTTP vorbereiten...
HTTPPostRequest.13=Öffne Verbindung...
HTTPPostRequest.16=Lese Dateiinformation...
HTTPPostRequest.21=Aushandeln...
...
```

Rekursive Auswahl

Werden im Dateiauswahldialog Verzeichnisse (Ordner) ausgewählt, werden diese rekursiv bearbeitet. Das bedeutet, dass alle unterhalb dieses Ordners zu findenen Dateien und Ordner durchsucht werden. Alle gefundenen Dateien werden in die Dateiliste übernommen.

Diese Funktion ist als standardmäßíg aktiviert und kann vom Administrator deaktiviert werden. Außerdem kann vom Administrator eingestellt werden, ob die Pfadangaben mit übertragen werden oder nicht. Viele Server erkennen allerdings Pfadangaben und entfernen diese automatisch aus Sicherheitsgründen. Sollten die Pfadangaben für Ihre Anwendung jedoch benötigt werden, ist es JUpload möglich die Pfadangaben zu verschleiern, sodass der Webserver diese nicht entfernen kann.

Um die rekursiven Pfadangaben zu übertragen, werden Sie von JUpload in die Typangabe eincodiert. Die Typangabe (Content-Type) wird von den Webservern durchgeschleust. Eingeschalten wird die Funktion durch folgenden Parameter:

```
<param name="useRecursivePaths" value="true">
```

Die relative Pfadangabe wird nun in den Content-Type eincodiert, dies sieht z. B. folgendermaßen aus:

```
Content-Type: JUpload/Fgjfg73dfhdlkj3jfdklsdfu4=
```

Dabei ist **JUpload** ein Identifizierungsstring, der Rest nach dem Slash der codiert Pfad. Dieser wird später in der serverseitigen Anwendung einfach durch base64_decode($string)[1] dekodiert und kann sofort genutzt werden.

Hier einige Beispiele:

Angenommen, auf dem lokalen Dateisystem liegen folgende Dateien:

```
/home/mike/work/work10.txt
/home/mike/work/old/work05.txt
/home/mike/work/backup/work02.txt
```

[1] http://www.php.net/base64_decode

Beim Upload ohne das useRecursivePaths-Feature, kommen die Dateien auf dem Server wie folgt an:

```
work10.txt
work05.txt
work02.txt
```

Es lässt sich dabei nicht mehr rekonstruieren, in welchen relativen Pfaden (old, backup) die Dateien vorher platziert waren. Mit Hilfe der Funktion, kommen beim Webserver nun folgende Daten an:

```
work/work10.txt
work/old/work05.txt
work/backup/work02.txt
```

Das basiert auf der Annahme, dass sich der Benutzer in seinem home-Verzeichnis befunden hat, und nur das Verzeichnis work ausgewählt und hochgeladen hat. Bitte beachten Sie, dass es für den Content-Type verschiedene Begrenzungen hinsichtlich der maximalen Länge gibt, und dass dadurch die Funktionsweise dieser JUpload-Sonderfunktion nicht gewährleistet werdenkann.

3.2.3 Debugging

In diesem Abschnitt wird die Vorgehensweise bei Programmfehlern oder Ablauffehlern erläutert. Treten Probleme bei der Datenübertragung oder bei der Funktionsweise von JUpload auf, sollte mit Hilfe des Debugmodus eine Untersuchung und Lokalisierung der Problemstelle stattfinden.

Das Umschalten in den Debugmodus erfolgt durch die Angabe des Parameters debug:

```
<param name="debug" value="true">
```

Die entsprechenden Debugmeldungen werden in diesem Modus in die Java Console geschrieben. Diese kann im Java Plug-In automatisch geöffnet werden[2]. Das *Java Console Log Window* ist bei den Browser unter verschiedenen Menüpunkten zu finden. Zusätzlich kann sie über das Systemtraysymbol des Java Plug-Ins geöffnet werden.

Die Debugmeldungen geben detaillierte Informationen über den Ablauf der Anwendung und den aktuellen Status aus. über einen zweiten Parameter kann eine zusätzliche Dateiausgabe eingeschalten werden:

```
<param name="debugLogfile" value="c:/jupload.log">
```

oder für Linuxbenutzer

```
<param name="debugLogfile" value="/home/username/jupload.log">
```

Dabei ist wichtig, dass der Benutzer Schreibrechte auf diese Datei besitzt.

[2] Einstellung im Plugin Control Panel

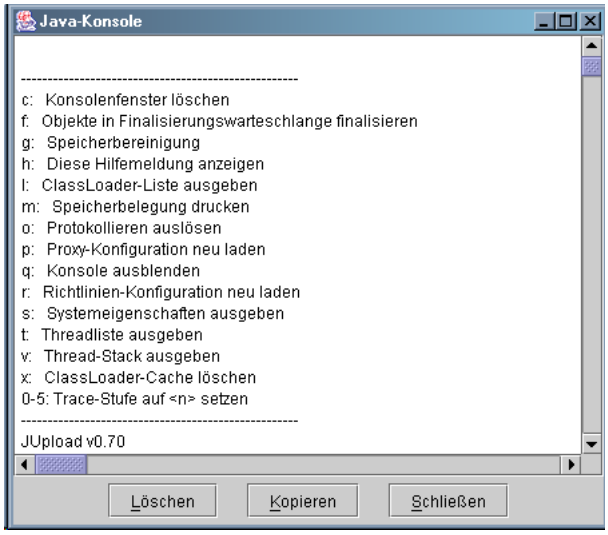

Abbildung 3.2: Java Console Log Window

3.3 Programmierschnittstelle

3.3.1 JavaScript API

Die Schnittstelle zwischen Java und JavaScript wurde von Netscape auf den Namen *LiveConnect*[3] getauft. Mit Hilfe dieses Pakets kann Java auf JavaScript Funktionen (und auch auf das DOM) zugreifen und vice versa. Beides nutzt JUpload bzw. JUpload unterstützt diese Erweiterungen.

Um von JavaScript aus auf JUpload-Funktionen zugreifen zu können, muss das Applet mit einem Namen versehen werden:

```
<applet width=123
  height=456 name="JUpload" alt="Mein Applet">
</applet>
```

Durch diesn Tag ist das Applet nun unter document.JUpload verwendbar bzw. können von JavaScript Funktionen aus JUpload-Methoden aufgerufen werden. Im folgenden Abschnitt werden diese Funktionen erläutert.

3.3.2 Funktionen

jsIsReady() Diese Funktion gibt true zurück, wenn JUpload vollständig geladen worden ist, ansonsten false.

jsRegisterAddListener(String) Mit dieser Funktion kann eine Callback Funktion für den Hinzufügen-Button registriert werden. Siehe Details im nächsten Abschnitt.

jsRegisterRemoveListener(String) Mit dieser Funktion kann eine Callback Funktion für den Entfernen-Button registriert werden. Siehe Details im nächsten Abschnitt.

[3] http://wp.netscape.com/eng/mozilla/3.0/handbook/javascript/livecon.htm

jsRegisterUploadListener(String) Mit dieser Funktion kann eine Callback Funktion für den Hochladen-Button registriert werden. Siehe Details im nächsten Abschnitt.

jsRegisterUploaded(String) Mit dieser Funktion kann eine Callback Funktion für den erfolgten Upload registriert werden. Siehe Details im nächsten Abschnitt.

jsGetFileNumber() Liefert die Anzahl der Dateien zurück, die sich in der Warteschlange befinden.

jsGetFileAt(Index) Liefer den (vollständigen) Dateinamen einer Datei in der Warteschlange mit dem angegebenen Index.

jsGetFileSizeAt(Index) Liefert die Dateigröße in Bytes.

jsClickAdd() Simuliert einen Click auf den Hinzufügen-Button: Öffnet den Dateiauswahldialog.

jsClickRemove() Simuliert einen Click auf den Entfernen-Button: Entfernt selektierte Dateien aus der Warteschlange. Sind keine Dateien selektiert, erscheint ein Hinweis.

jsClickUpload() Simuliert einen Click auf den Hochladen-Button: Startet den Datenübertragunsprozess.

3.3.3 Callback Funktionen

Für die Verwendung der Callback Funktionen muss JUpload die Namen der JavaScript Handler bekannt sein und es muss das Schlüsselwort *mayscript* im Applet-Tag stehen. Um die JavaScript-Funktionen beim Applet zu registrieren, dienen die Methoden *jsRegister**.

jsRegisterAddListener(Funktionsname) Mit dieser Funktion kann eine Callback Funktion für den Hinzufügen-Button registriert werden. Diese benutzerdefinierte JavaScript-Funktion wird aufgerufen, bevor der Benutzer eine Datei zur Warteschlange hinzufügt.

jsRegisterRemoveListener(Funktionsname) Mit dieser Funktion kann eine Callback Funktion für den Entfernen-Button registriert werden. Diese benutzerdefinierte JavaScript-Funktion wird aufgerufen, bevor der Benutzer eine Datei aus der Warteschlange entfernt.

jsRegisterUploadListener(Funktionsname) Mit dieser Funktion kann eine Callback Funktion für den Hochladen-Button registriert werden. Diese benutzerdefinierte JavaScript-Funktion wird aufgerufen, nachdem der Benutzer auf den Hochladen-Button geklickt hat und bevor der eigentliche Upload startet. Achtung: die JavaScript Funktionen blockieren die Ablaufsteuerung des Applets, d.h. solange die JavaScript-Funktionen kein *return* durchführen, pausiert das Applet.

jsRegisterUploaded(Funktionsname) Mit dieser Funktion kann eine Callback Funktion für den erfolgten Upload registriert werden. Diese benutzerdefinierte JavaScript-Funktion wird aufgerufen, nachdem der Upload durchgeführt wurde und bevor evtl. auf eine neue Webseite umgeleitet wird.

3.3.4 Beispiel

```
<html><head><title>JUpload JavaScript API Example</title>
<SCRIPT language="JavaScript">
<!--
function myOnLoad()
{
  if (!document.JUpload) return;

  // check if jupload is up and running
  if (document.JUpload.jsIsReady())
  {
    // Register the listeners
    document.JUpload.jsRegisterAddListener("myAddListener");
    document.JUpload.jsRegisterRemoveListener("myRemoveListener");
    document.JUpload.jsRegisterUploadListener("myUploadListener");
    document.JUpload.jsRegisterUploaded("myUploaded");
  }
  else
  {
    // wait and try again until JUpload is ready
    window.setTimeout('myOnLoad()',400);
  }
}

function myAddListener()
{
 alert("A file was added");
}

function myRemoveListener()
{
 alert("A file was remove");
}

function myUploadListener()
{
 alert("Upload will be started");
}

function myUploaded()
{
 alert("Upload is finished");
}


function displayFileQueue()
{
 // if there is no JUpload, go back
 if(!document.JUpload) return;
 if(!document.JUpload.jsIsReady()) return;

 // open a new empty document
 F1=window.open("about:blank","queueWindow","width=300, height=400");
 F1.document.writeln("<html><body><h1>file list:</h1>there are");
 var number = document.JUpload.jsGetFileNumber();
 F1.document.writeln(number);
 F1.document.writeln("files.<ul>");
 // write a file list
 for(i=0;i<number;i++)
  {
   F1.document.writeln("<li>"+document.JUpload.jsGetFileAt(i));
   F1.document.writeln("<br><font color=green>Size: "+document.JUpload.jsGetFileSizeAt(i)+"</font>");
  }
 F1.document.writeln("</ul><body></html>");
}

//-->
</SCRIPT></HEAD>
<BODY onLoad="myOnLoad();">

<form name="JUploadForm">
 <input type=button value="Add files" onClick="document.JUpload.jsClickAdd();">
 <input type=button value="Remove files" onClick="document.JUpload.jsClickRemove();">
 <input type=button value="Start Upload" onClick="document.JUpload.jsClickUpload();">
 <input type=button value="Display File Queue" onClick="displayFileQueue();">
</form>

<applet
  code="JUpload/startup.class"
  archive="JUpload.jar"
  name="JUpload"
  mayscript
  width="650"
  height="400"
  alt="JUpload uploads files to webserver">
 <PARAM NAME="actionURL" VALUE="JUpload.php">
</applet>
</body>
</html>
```

Kapitel 4

Entwicklerhandbuch

Dieser Abschnitt des Handbuchs ist für Java-Entwickler, welche Erklärungen zu der Funktionsweise von JUpload benötigen. Dieser Teil des Handbuchs handelt vom Quelltext des Produkts, dessen Strukturierung und von den eingesetzten Protokollen.

4.1 Programmierschnittstelle

4.1.1 API JavaDoc

Die JavaDoc findet sie im JUpload-Entwickler Paket im Unterverzeichnis *javadoc*. Sie benötigen lediglich einen Browser, um die Dokumentation betrachten zu können. Die JavaScript API ist im Administratorenhandbuch näher erläutert.

4.2 Hinweise

4.2.1 Anhang

Im Anhang finden Sie detaillierte Klassendiagramme und den gedruckten und kommentierten Source Code.

4.2.2 Protokolle

Die verwendeten Methoden beruhen zum großteil auf dem Protokoll HTTP. Siehe Details zu HTTP/MIME unter RFC1867 (http://www.faqs.org/rfcs/rfc1867.html und ftp://ftp.rfc-editor.org/in-notes/rfc2387.txt).

Für Details über das HTTP Protokoll siehe (ftp://ftp.rfc-editor.org/in-notes/rfc2616.txt)

4.2.3 JavaScript

JavaScript bzw. das benötigte LiveConnect Paket [1] ist nicht in allen Browser implementiert.

[1] Für Entwickler: im Netscape Navigator 3.0 enthalten: java_30.jar

Folglich kann auch JUpload nicht mit jedem Browser auf das DOM [2] zugreifen. Getestet und als funktionstüchtig erkannt wurden:

Browser	**Windows**[3]	**Linux**[4]	**Macintosh**
Internet Explorer 6	Ja	-	-
Netscape Navigator	-	-	-
Opera 7	-	-	-
Mozilla 1.5	Ja	Ja	-
Konqueror	-	Nein	-

Ein „-“ bedeutet, dass noch keine Test durchgeführt wurden.

Zugriff auf HTML Formfelder

Um evtl. Benutzereingaben zusammen mit Dateien verschicken zu können, werden oft HTML Formulare benutzt. Diese können Texteinträge, Passworteinaben, Checkboxen, Radioboxen und Selektionen beinhalten. Um diese Felder zu übertragen, nutzt JUpload den Zugriff auf den DOM-Objektbaum. Über das LiveConnect-Paket erhält man das *document-Objekt*, unter dem alle folgenden Objekte angeordnet sind. Die Struktur kann anhand dieses einfachen Modells nachvollzogen werden:

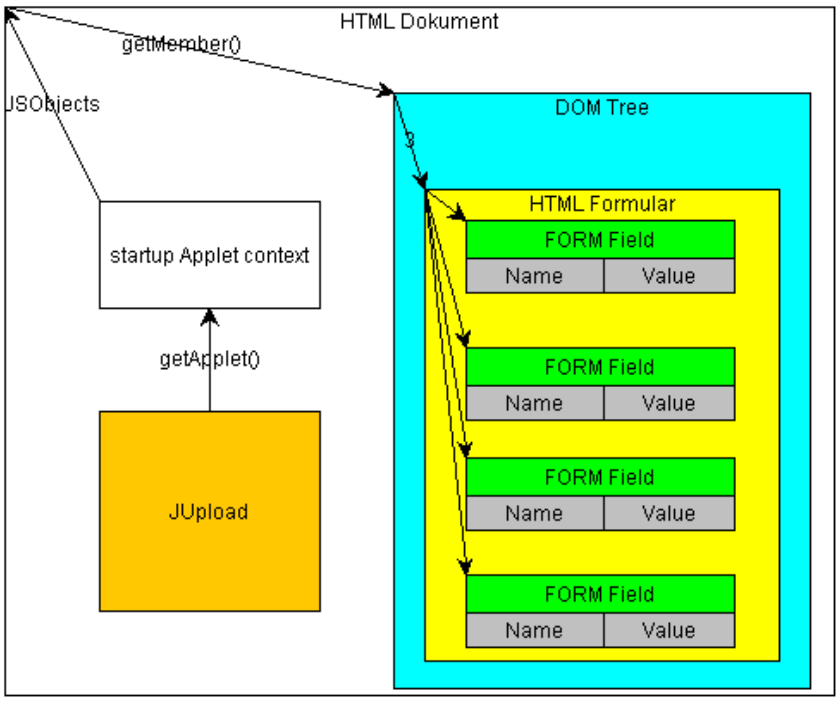

Abbildung 4.1: Zugriffshierarchie auf Formularfelder

Unter bestimmten Voraussetzungen ist es nicht möglich, die einzelnen Formularfelder als Arrayelemente anzusprechen. Eine Lösung dieses Problems ist der Zugriff auf die Formularfelder über ihr *name-Attribut*, welche dann aber fest vorgegeben[5] werden

[2] Data Object Model: http://www.w3.org/DOM/

[5] Siehe hierzu Parametereinstellung *includeFormFields*

müssen.

Für Entwickler sieht der Zugriff auf JavaScript Objekte wie folgt aus:

```
1  // Durchhangeln bis zu den Formularfeldern
2  JSObject win=parent.applet.jsBrowserWindow;
3  JSObject doc=(JSObject) win.getMember("document");
4  JSObject form=(JSObject) doc.getMember("JUploadForm");
5  JSObject elements=(JSObject) form.getMember("elements");
6
7  // Anzahl der Felder ermitteln
8  Double dnum=new Double(elements.getMember("length").toString());
9  int    num=dnum.intValue();
10
11 // Einzeln auslesen
12 for(int i=0;i < num;i++){
13  Object thisSlot=elements.getSlot(i);
14  String fieldName =(String) jsThisSlot.getMember("name");
15  String fieldValue=(String) jsThisSlot.getMember("value");
16 }
```

Kapitel 5

Anhang

5.1 Konfiguration

Das (inzwischen veraltete) Applet-Tag wird wie folgt erzeugt:

```
<applet
  code="JUpload/startup.class"
  archive="JUpload.jar"
  width="500"
  height="300"
  mayscript
  name="JUploadApplet"
  alt="JUpload applet">
 <!-- hier die PARAM-Tags einsetzen -->
</applet>
```

code Gibt den Namen der Startklasse an, falls notwendig mit Angabe des Packages.

archive Gibt den Dateinamen bzw. die URL des jar-Archivs an, welches die Anwendung und benötigte Bibliotheken oder Resourcen-Dateien enthält. Hier können auch mehrere Archive (kommagetrennt) angegeben werden, die dann auch in entsprechender Reihenfolge geladen werden. Besitzen zwei jar-Archive die gleiche Signatur, wird das Zertifikatswindow nur einmal angezeigt. Haben Sie jedoch unterschiedliche Signaturen, wird auch das Zertifikatswindow mehrfach angezeigt und der Benutzer muss für jede Signatur einzeln bestätigen bzw. akzeptieren.

width Gibt die reservierte Breite für das Appletfenster in Pixel an.

height Gibt die reservierte Höhe für das Appletfenster in Pixel an.

alt Gibt einen Beschreibungstext für die Anwendung an. Diese sollte erscheinen, wenn der Benutzer mit der Maus über dem Applet schwebt bzw. wenn der Browser das Applet nicht laden kann.

mayscript Ein Schlüsselwort um die JavaScript-Unterstützung zu aktivieren.

name Setzt den Namen des JUpload-Applets (bzw. dem entsprechenden DOM-Objekt) um von JavaScript auf das Applet zugreifen zu können. Es kann dann mittels document.JUploadApplet angesprochen werden.

Laut Sun's Empfehlung sollte man allerdings auf die neuen Tags *OBJECT* und *EMBED* umsteigen - Object für den Internet Explorer und Embed für den Netscape Navigator.

Mozilla.org gibt diesen Code für die Verwendung an:

```
<OBJECT classid="java:appletname.class" type="application/java"
    codebase="..." archive="..." width="..." height="...">
  <PARAM name="..." value="..." /> [...]
</OBJECT>
```

Für eine genaue Beschreibung des Object-Tags, siehe W3C (http://www.w3.org/TR/html401/struct/objects.html#edef-OBJECT)

Bisheriges APPLET tag:

```
<APPLET code="XYZApp.class" codebase="html/" align="baseline"
    width="200" height="200">
<PARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
    No Java 2 SDK, Standard Edition v 1.3 support for APPLET!!
</APPLET>
```

Der folgende Code ist ein Beispiel für das equivalente Java Plug-in Tag, herausgenommmen aus der Dokumentation von Sun Microsystems:

```
<!-- The following code is specified at the beginning of the <BODY> tag. -->
<SCRIPT LANGUAGE="JavaScript"><!--
    var _info = navigator.userAgent; var _ns = false;
    var _ie = (_info.indexOf("MSIE") > 0 && _info.indexOf("Win") > 0
        && _info.indexOf("Windows 3.1") < 0);
//--></SCRIPT>
<COMMENT><SCRIPT LANGUAGE="JavaScript1.1"><!--
    var _ns = (navigator.appName.indexOf("Netscape") >= 0
        && ((_info.indexOf("Win") > 0 && _info.indexOf("Win16") < 0
        && java.lang.System.getProperty("os.version").indexOf("3.5") < 0)
        || _info.indexOf("Sun") > 0));
//--></SCRIPT></COMMENT>


<!-- The following code is repeated for each APPLET tag -->
<SCRIPT LANGUAGE="JavaScript"><!--
    if (_ie == true) document.writeln('
<OBJECT
    classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
    width="200" height="200" align="baseline"
codebase="http://java.sun.com/products/plugin/1.3/jinstall-13-win32.cab#Version=1,3,0,0">
    <NOEMBED><XMP>');
    else if (_ns == true) document.writeln('
<EMBED
    type="application/x-java-applet;version=1.3" width="200" height="200"
    align="baseline" code="XYZApp.class" codebase="html/"
    model="models/HyaluronicAcid.xyz"
    pluginspage="http://java.sun.com/products/plugin/1.3/plugin-install.html">
    <NOEMBED><XMP>');
//--></SCRIPT>
   <APPLET code="XYZApp.class" codebase="html/" align="baseline"
        width="200" height="200">
</XMP>
    <PARAM NAME="java_code" VALUE="XYZApp.class">
    <PARAM NAME="java_codebase" VALUE="html/">
    <PARAM NAME="java_type" VALUE="application/x-java-applet;version=1.3">
    <PARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
    <PARAM NAME="scriptable" VALUE="true">
    No Java 2 SDK, Standard Edition v 1.3 support for APPLET!!
</APPLET></NOEMBED></EMBED>
</OBJECT>

<!--
    <APPLET code="XYZApp.class" codebase="html/" align="baseline"
        width="200" height="200">
    <PARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
    No Java 2 SDK, Standard Edition v 1.3 support for APPLET!!
    </APPLET>
-->
```

5.1.1 Plug-In Einstellungen

Zur Vollständigkeit werden hier nochmals die möglichen Einstellungen für das Anpassen des Java Plug-Ins erklärt.

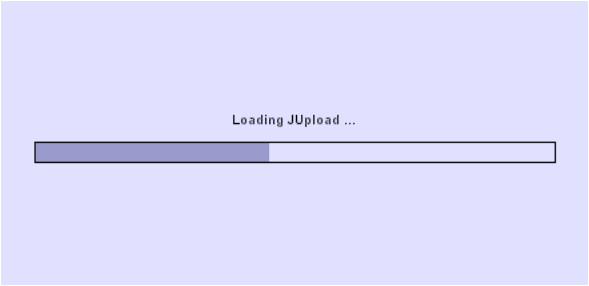

Abbildung 5.1: Ladevorgang mit progressbar und boxmessage

progressBar Schaltet die Anzeige eines Prozentbalkens während des Ladens des Archivs ein bzw. aus. Wert true schaltet ein, Wert false aus.

```
<param name="progressbar" value="true">
```

boxmessage Ändert die angezeigte Nachricht während des Ladens des jar-Archivs.

```
<param name="boxmessage" value="Loading JUpload Applet ...">
```

boxbgcolor Ändert die Farbe der angezeigte Nachricht während des Ladens des jar-Archivs.

```
<param name="boxbgcolor" value="#e0e0ff">
```

5.1.2 JUpload Parametereinstellungen

Im folgenden werden alle unterstützten Parametereinstellungen eklärt.

actionURL Adresse des serverseitigen Scripts, welches den Upload entgegennimmt.

```
<param name="actionURL" value="http://localhost/demo/JUpload.php">
```

completeURL Angabe einer URL, die nach dem erfolgreichen Upload im Browser aufgerufen wird. Dabei wird das aktuelle Dokument ersetzt. Diese wird nur aufgerufen, wenn showRealtimeResponse ausgeschalten ist.

```
<param name="completeURL" value="http://www.example.com/upload-ok.html">
```

errorURL Angabe einer URL, die nach dem missglückten Upload im Browser aufgerufen wird. Dabei wird das aktuelle Dokument ersetzt. Diese wird nur aufgerufen, wenn showRealtimeRespone ausgeschalten ist.

```
<param name="errorURL" value="http://www.example.com/upload-error.html">
```

debug Eine boolean-Variable zum Wechseln in den Debugmodus. Wert true schaltet ein, Wert false schaltet aus.

```
<param name="debug" value="true">
```

askAuthentification Die boolean-Funktion schaltet die Benutzerverifikation ein. Nutz der Webserver die Authentifikationsmethode Basic, um das entgegennehmende Uploadscript zu schützen, kann JUpload dafür die Authentifikation des Benutzers einholen. Dies gilt auch für Benutzergeschützte Proxyserver.

```
<param name="askAuthentification" value="false">
```

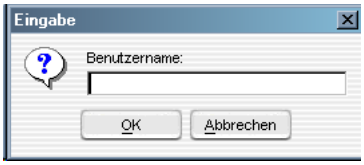

Abbildung 5.2: Eingabedialog für den Benutzernamen bei eingeschaltetem askAuthentification

tagName Dieser String-Parameter gibt den Feldnamen der Dateianhänge für den Upload an. Dieser ist Standardmässig uploadedFiles. JUpload hängt daran eine fortlaufende Nummer an.

```
<param name="tagName" value="uploadedFiles">
```

checkResponse Dieser boolean-Parameter schaltet die Prüfung der Serverantwort durch JUpload ein bzw. aus.

```
<param name="checkResponse" value="true">
```

showSuccessDialog Dieser boolean-Parameter gibt an, ob JUpload die Serverantwort als Meldungsdialog anzeigen soll oder nicht. Diese Funktion setzt voraus, dass checkResponse angeschalten und realTimeResponse ausgeschalten ist.

```
<param name="showSuccessDialog" value="true">
```

Abbildung 5.3: Meldung über erfolgreichen Upload

imageFileFilter Dieser boolean-Parameter schaltet den Bilddateienfilter ein. Im Auswahldialog für die hochzuladenden Dateien erscheint dann in der Liste der Dateitypen zusötzlich ein Dateifilter nur für Bilddateien.

```
<param name="imageFileFilter" value="false">
```

hideShowAll Dieser boolean-Parameter gibt an, dass der Standarddateifilter für alle Dateien im Dateiauswahldialog nicht mehr angezeigt wird.

```
<param name="hideShowAll" value="false">
```

backgroundColor Gibt die Hintergrundfarbe für die meisten der verwendeten Komponenten an. So kann das Applet an die Farbgestaltung der Webseite angepasst werden. Die Angabe erfolgt im bekannten Schema #RRGGBB mit Hexadezimalwerten für Rot, Grün und Blau. Stringnamen wie red, orange etc. werden nicht unterstützt.

```
<param name="backgroundColor" value="#e0e0ff">
```

showPicturePreview Schaltet beim Setzen auf false den Vorschaumodus aus und kann mit Setzen auf true wieder aktiviert werden.

```
<param name="showPicturePreview" value="false">
```

addWindowTitle Ändert den Fenstertitel des Dateiauswahldialogs.

```
<param name="addWindowTitle" value="Dateien auswählen">
```

imageURL Gibt eine URL für ein Logo oder für ein Bild an, welches im rechten unteren bereich des Statusbereichs (unterhalb der Buttons und Prozentbalken) angezeigt wird. Unterstützt JPG und transparente GIF-Bilder, jedoch keine Animationen.

```
<param name="imageURL" value="../images/applet-logo.gif">
```

labelFiles Gibt den Text für die Anzahl der Dateien an („Files:“ bzw. „Dateien:“);

```
<param name="labelFiles" value="Dateien:">
```

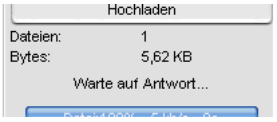

Abbildung 5.4: Die Labels für Dateianzahl und Gesamtgröße

labelBytes Gibt den Text für die Größe der Dateien an.

```
<param name="labelbytes" value="Größe:">
```

labelAdd Gibt den Text auf dem Button für Hinzufügen der Dateien an.

```
<param name="labelAdd" value="Hinzufügen">
```

labelRemove Gibt den Text für den Entfernen-Button an.

```
<param name="labelRemove" value="Entfernen">
```

labelUpload Gibt den Text für den Hochladen-Button an.

```
<param name="labelUpload" value="Hochladen">
```

successDialogMessage Gibt den Text an, welche nach dem erfolgreichen Hochladen in einem Dialog angezeigt wird. Der Benutzer muss diesen Dialog mit OK bestätigen, um fortzufahren.

```
<param name="successDialogMessage" value="Hochladen war erfolgreich">
```

successDialogTitle Gibt den Titel des o.g. Dialogs an.

```
<param name="successDialogTitle" value="Alles Okay!">
```

addToolTip Gibt den Tooltip-Text für den Hinzufügen-Button an.

```
<param name="addToolTip" value="Klicken um Dateien hinzuzufügen">
```

removeToolTip Gibt den Tooltip-Text für den Entfernen-Button an.

```
<param name="removeToolTip" value="Klicken um Dateien zu entfernen">
```

uploadToolTip Gibt den Tooltip-Text für den Hochladen-Button an.

```
<param name="uploadToolTip" value="Klicken um gewählte Dateien zu übertragen">
```

useRecursivePaths Schaltet die Funktion für Unterstützung von rekursiven (und relativen) Pfadangaben an. Die Pfadangaben werden dabei innerhalb des Content-Types codiert und mitübertragen. Die Codierung ist eine einfache base64-Codierung und kann mittels base64_decode($strEncodedPath) in PHP einfach rückgängig gemacht werden.

```
<param name="useRecursivePaths" value="true">
```

checkJavaVersion Schaltet die Prüfung der Java Version auf dem Clientrechner ein bzw. aus.

```
<param name="checkJavaVersion" value="true">
```

checkJavaVersionGotoURL Ist die entdeckte Java-Version zu alt, wird der Benutzer auf diese Seite weitergeleitet.

```
<param name="checkJavaVersionGotoURL" value="http://www.java.com/">
```

browserCookie Enthält einen Cookie, der zusammen mit dem Request an dern Server geschickt werden soll. Wird für Anwendungen benötigt, die mit Sessions arbeiten und die Session-id weder über den Query-String noch über versteckte Formfelder übertragen können. Dieses Feld muss zwingend dynamisch in den Applet-Tag-Source eingebunden werden: mittels JavaScript:

```
document.write('<param name=browserCookie value="');
document.write(document.cookie);
document.writeln('">');
```

JUpload ist es nicht möglich, auf die beim Sessionstart angelegten Cookies im Browser zuzugreifen.

defaultAddDirectory Gibt ein Startverzeichnis für den Dateiauswahldialog an. Achtung: verschiedene Plattformen, verschiedene Pfade. Eine Angabe von c:/dateien würde bei einem Linux-Benutzer nicht funktionieren und vice versa.

```
<param name="defaultAddDirectory" value="c:/dateien">
```

usePutMethod Schaltet die Verwendung der speziellen PUT-Methode ein. Vorteil dieser Methode ist, dass extrem grosse Dateien übertragen werden können, da der Upload hierbei in Fragmenten realisiert ist. Desweiteren ist diese Methode besser, wenn Proxies und Firewalls zwischen Client und Server stehen. Allerdings muss die PUT-Methode auf dem Webserver (und in der entsprechenden Webanwendung) implementiert sein. Ein Beispiel-Script für PHP liegt dem JUpload-Paket bei.

```
<param name="usePutMethod" value="true">
```

maxFreeSpaceOnServer Gibt die maximale Größe in Bytes an, die ein Request haben darf.

```
<param name="maxFreeSpaceOnServer" value="20000000">
```

maxFreeSpaceOnServerWarning Wird die maximale Platz auf dem Webserver überschritten, wird diese Warnung ausgegeben.

```
<param name="maxFreeSpaceOnServerWarning" value="Ihr Server-Account ist überfüllt!">
```

maxFreeSpaceOnServerTitle Der Fenstertitel für o.g. Warnung.

```
<param name="maxFreeSpaceOnServerTitle" value="Serverplatz reicht nicht aus">
```

maxTotalRequestSize Gibt die maximale Größe eines einzelnen Requests an. Ist die Bytemenge aller Dateien in der Warteschlange größer als dieser Wert, werden die Dateien in einzelne Requests aufgesplittet. Eine einzelne Datei kann nicht größer als dieser Wert sein.

```
<param name="maxTotalRequestSize" value="2000000">
```

maxTotalRequestSizeWarning Wird der o.g. Wert überschritten, wird diese Meldung ausgegeben.

maxTotalRequestSizeTitle Der Fenstertitel für o.g. Warnung.

customFileFilter Schaltet einen benutzerdefinierten Dateifilter ein. Dateifilter basieren rein auf der Dateierweiterung (auf den letzte Ziffern eines Dateinamens ab dem letzten Punkt)

```
<param name="customFileFilter" value="true">
```

customFileFilterDescription Ändert die Kurzbeschreibung für den benutzerdefinierten Dateifilter. (z.B. „Bilddateien“)

```
<param name="customFileFilterDescription" value="Bilddateien (Jpeg und GIF)">
```

customFileFilterExtensions Gibt eine Liste von Dateierweiterungen für den benutzerdefinierten Dateifilter an. Diese Liste ist kommagetrennt. Die einzelnen Erweiterungen werden ohne Punkt angegeben.

```
<param name="customFileFilterExtensions" value="gif,jpg,jpeg">
```

maxNumberFiles Gibt eine maximale Anzahl von Dateien an, die Hochgeladen werden kann.

```
<param name="maxNumberFiles" value="1000">
```

maxNumberFilesWarning Wird die maximale Anzahl von Dateien überschritten, wird diese Warnmeldung ausgegeben.

maxNumberFilesTitle Gibt den Fenstertitel für die Warnmeldung an.

fixJakartaBug Sollten Probleme im Zusammenhang mit Jakarta-Fileupload auftreten, versuchen Sie diesen Schalter auf true zu setzen.

```
<param name="fixJakartaBug" value="false">
```

maxFilesPerRequest Gibt die maximale Anzahl von Dateien pro Request an. Wird hier z.B. 1 angegeben, wird jede Datei einzeln hochgeladen. Eine Angabe von -1 deaktiviert die Limitierung.

showServerResponse Schaltet das Anzeigen der Serverantwort ein. realTimeResponse muss dabei deaktiviert sein.

```
<param name="showServerResponse" value="true">
```

Ist die Serverresponse ausgeschalten, bedeutet dies jedoch nicht, dass keine Anzeige stattfindet. Hierbei wird nur die Antwort von einem modalen Dialog in das Realtimewindow verlegt:

preselectedFiles Gibt eine Liste von Dateien an, die JUpload bereits beim Start in die Warteschlange stellen soll. Der Benutzer hat immernoch die Möglichkeit, die Dateien manuell zu entfernen und/oder weitere Dateien hinzuzufügen, bevor er mit dem Upload fortfährt. Eine weiterführende Automatisierung kann durch JavaScript erzielt werden, z.B. das automatische Anstarten des Uploads nach dem Laden des Applets ohne weiteres Benutzerzutun.

```
<param name="preselectedFiles"
 value="d:/webcam1.jpg;d:/webcam2.jpg">
```

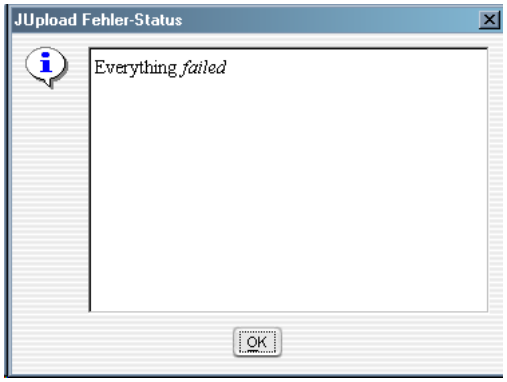

Abbildung 5.5: Fenster mit HTML-Antwort des Servers bei missglücktem Upload

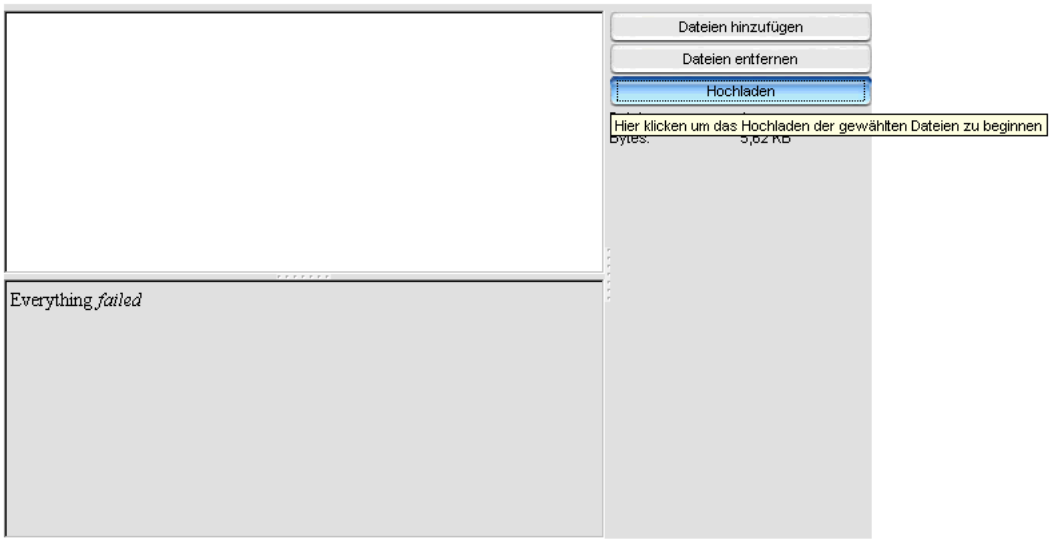

Abbildung 5.6: Fehlermeldung im Realtime-Window

realTimeResponse Schaltet die Anzeige der Realtime-Serverantwort ein bzw. aus. Diese ist im linken unteren Bereich des JUpload Hauptbildschirms zu sehen. Die Realtime-Anzeige kann die einfache HTML-Ausgabe des serverseitigen Scripts in Echtzeit angezeigt werden, sofern das Script keine Ausgabepuffering einsetzt. Ausgabepuffering kann in PHP mit flush() geleert werden.

```
<param name="realTimeResponse" value="false">
```

debugLogfile Gibt einen lokalen Dateinamen für die Logdatei an. Die Logdatei wird (sofern Debugging eingeschaltet ist) im Dateisystem des Clientrechners abgelegt und enthält größtenteils die gleiche Ausgabe wie die Java Console Log.

```
<param name="debugLogfile" value="c:/jupload.log">
```

mainSplitpaneLocation Gibt die absolute Position des mittleren Trennbalkens an (lin-

ke Seite: Dateiwarteschlange, rechte Seite: Buttons etc). Die Position ist die Angabe des Abstands vom linken Appletrand in Pixel.

```
<param name="mainSplitpaneLocation" value="450">
```

hideAddButton Schalter versteckt den Hinzufügen-Button

```
<param name="hideAddButton" value="true">
```

hideRemoveButton Schalter versteckt den Entfernen-Button

hideUploadButton Schalter versteckt den Hochladen-Button

showStatusPanel Schalter für das Anzeigen bzw. Verstecken der rechten Statusseite des Applets.

```
<param name="showStatusPanel" value="false">
```

skinThemePackURL Eine Adressangabe zu einem SkinLF Theme Pack. Achtung: für diese Funktion muss das Paket skinlf.jar auf dem Server vorhanden sein und im Applet-Tag miteingebunden werden.

```
<applet ...
   archive="skinlf.jar,jupload.jar"
  ... alt="JUpload Applet">
<param name="skinThemePackURL" value="aquathemepack.zip">
</applet>
```

useProxy Forciert das Benutzen eines Proxy Servers, welcher in den Java Plugin Einstellungen bereits vorkonfiguriert sein muss.

```
<param name="useProxy" value="false">
```

showThumbnails Schaltet die Anzeige von Vorschaubildern (Miniaturansichten) in der Warteschlange ein bzw. aus.

```
<param name="showThumbnails" value="false">
```

targetFrame Gibt den Namen eines Zielframes für die Dokumente an, welche für completeURL-Weiterleitungen dorthin geladen werden sollen. Spezialwerte wie z.B. _blank können verwendet werden, um die Seite in einem neuen Fenster zu öffnen.

```
<param name="targetFrame" value="MyLeftFrame">
```

useAbsolutePaths Schalter für das Übermitteln von absoluten Pfadangaben. Standardeinstellung ist ausgeschalten, es werden nur relative Pfadangaben übermittelt. Relative Pfadangaben beziehen sich dann auf das Verzeichnis, von welchem aus der Benutzer die rekursiv hochzuladenden Unterordner angewählt hat. Achtung: useRecursivePaths muss hierfür eingeschalten sein.

```
<param name="useAbsolutePaths" value="false">
```

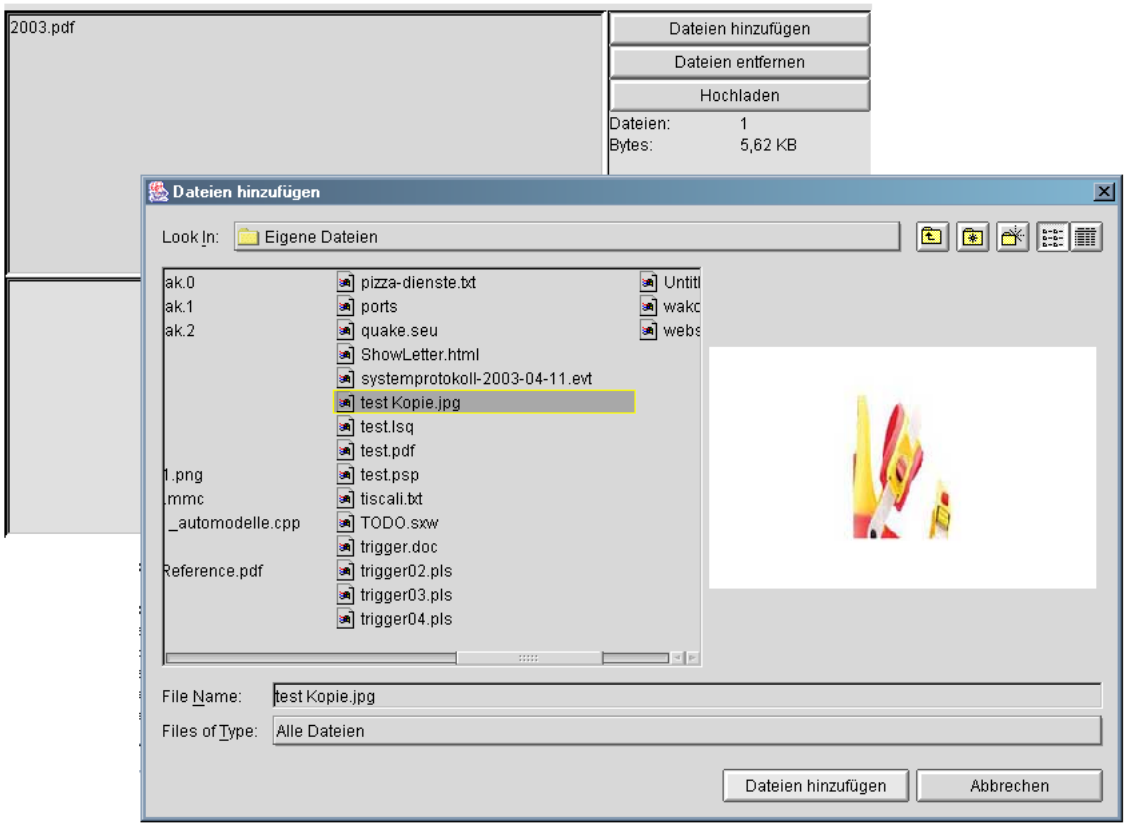

Abbildung 5.7: Es können leicht eigene Skins benutzt werden

Abbildungsverzeichnis

Literaturverzeichnis

[1] **E. Nebel, L. Masinter** (1995), „Form-based File Upload in HTML“, http://www.faqs.org/rfcs/rfc1867.html

[2] **E. Levinson** (1998), „The MIME Multipart/Related Content-type“, ftp://ftp.rfc-editor.org/in-notes/rfc2387.txt

[3] **R. Fielding, J. Gettys, J. Mogul et al.** (1999), „Hypertext Transfer Protocol – HTTP/1.1“, ftp://ftp.rfc-editor.org/in-notes/rfc2616.txt

[4] **Keld Simonsen et al.** (1990-1996), „Code for the representation of names of languages“, http://ftp.ics.uci.edu/pub/ietf/http/related/iso639.txt

[5] **Maintenance Agency for ISO 3166 country codes** (1999), „ISO 3166 Codes (Countries)“, http://www.iso.org/iso/en/prods-services/iso3166ma/index.html

[6] **Philippe Le Hégaret et al.** (2002), „Document Object Model (DOM)“, http://www.w3.org/DOM/

[7] **Dave Raggett et al.** (1999), „HTML 4.01 Specification“, http://www.w3.org/TR/html401/struct/objects.html#edef-OBJECT