Kapitel I.2: Upload-Applet

Ein Applet zum Uploaden von einzelnen Dateien bzw. Verzeichnissenb umen

a: Ziel der Entwicklung

Mit HTML-Formularen lassen sich nur einzelne Dateien via http uploaden. Der Upload von ganzen Verzeichnissen ist nicht m glich. Diese Funtionalit t ist f r Digilib w nschenswert. Die L sung f r dieses Problem ist die Entwicklung eines Java-Applets was dieses kann und dar ber hinaus auch eine Versionskontrolle erlaubt.

b: Anforderungsprofil

Das Upload-Applet erm glicht den kontrollierten Upload einzelner Dateien bzw. ganzer Ordner. Kontrolliert bedeutet hierbei das sowohl die Pr fsumme (MD5), wie auch die Gr §e, sowie auch die Anzahl der Dateien berpr ft wird.

Der Upload erfolgt immer in einen tempor ren Ordner, erst wenn der Upload erfolgreich abgeschlossen wurde , wird die Datei bzw. der Ordner an die eigentliche Stelle verschoben.

Hierbei sind zwei F lle zu unterscheiden:

- Der Ordner bzw die Datei existiert noch nicht. Der Upload wird einfach ausge- f hrt.
- Der Ordner bzw die Datei existiert an dem angebenen Ziel schon. Der Benutzer muss nun entscheiden ob er berschreiben oder im Falle eines Ordners erg nzen m chte. Grunds tzlich werden keine Dateien berschrieben. Die Dateien bzw. Ordner werden umbenannt. Es wird das aktuelle Datum und Uhrzeit angeh ngt, z. B. aus `kopernikus` wird `kopernikus.15.09.2002.15.04.00`. In dem Fall das einzelne Dateien in einem vorhandenen Ordner ersetzt werden sollen, ist die Frage ob man sie nicht in einen nach dem voherigen Schema umbenannten Ordner verschiebt, anstatt die Dateien umzubenennen.

c: Featureliste

Feature	Status der Implementierung
Upload von Dateien bzw. eines Ordners und anschließendes verschieben	implementiert
Ersetzen des Ordners falls er existiert. Der original Ordner wird umbenannt,wobei der Benutzer gefragt wird, ob er dies tun möchte	implementiert

Tabelle 1: Featureliste

Feature	Status der Implementierung
Überprüfen der Prüfsumme	implementiert
Bei ungleicher Prüfsumme wird der Benutzer gefragt ob er den Upload wiederholen möchte.	implementiert
Upload grosser Dateien	implementiert
Anpassung an Java 1.1(MacOS 9)	implementiert
Zielordner angeben falls der lokale Ordnername nicht dem Projektordnernamen entspricht.	implementiert
Mehrere Uploads gleichzeitig (Multi-Threading)	implementiert
Abbruch des Uploads	implementiert
Manuelle Einstellung von Proxys	implementiert
Eingeschränktes automatisches auslesen der Proxy Einstellung	teilweise möglich muss noch verbessert werden
Grafische und Numerische Fortschritts Anzeige mit Dateigrößenangabe	implementiert
Abschlussdialog mit Uploadzeit	implementiert

Tabelle 1: Featureliste

d: Anwendungsbeispiele

Schritt 1: Browserstarten + URL eingeben
http://erebos.mpiwg-berlin.mpg.de:8088/upload/up/?baseURL=%22%2fheute%f%22

Abb. 1: Zertifikat akzeptieren

Das obige Dialog fenster erscheint. Als n chstes akzeptiert man das Zertifikat nuf f r die Sitzung. Wenn alles geklappt hat erscheint der Upload -Button.

Schritt 2: Start des Uploads

Klicken des Upload -Buttons.

Abb. 2: Upload Button

Schritt 3: Auswahl der Datei bzw. des Ordners

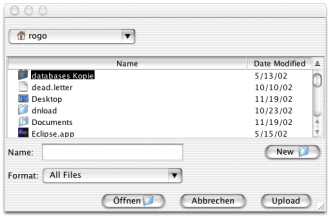

Abb. 3: Datei Dialog

Aus obigen Dateidialog w hlt man den Ordner aus, hier databases Kopie.
Anschlie§end dr ckt man auf Upload.

Schritt 4: Der ausgew hlte Ordner existiert schon .
Folgender Dialog erscheint.

Abb. 4: Ordner exisitiert!!!

Fall 1: Replace ersetzt den ganzen Ordner!
Fall 2: Cancel beendet den Upload und der Dialog verchwindet.
Schritt 5: Nach dem Replace ausgew hlt wurde beginnt der Upload

Folgender Progressdialog wird angezeigt:

Abb. 5: Progressdialog

Nach Beendigung des Uploads verschwindet der Progressdialog wieder.
Und ein Abschlussdialog wird angezeigt.

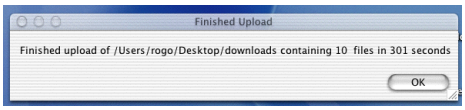

Abb. 6: Abschlussdialog

e: Installation des Servlets für Tomcat

Out of the box

Im ersten Schritt mu§ Tomcat installiert werden, wenn diese Serversoftware noch nicht auf dem Server residiert.

Der Installationsordner `upload`, der aus dem Archiv `upload.zip` nach dem Entpacken entsteht, enth lt folgende Dateien:

- ¥ applet.html Beispiel HTML Datei
- ¥ upload.jar Das Applet-Archiv.
- ¥ upload.xml Tomcat- Context-Datei
- ¥ WEB-INF Order
- ¥ WEB-INF/classes/ Ordner
- ¥ WEB-INF/classes/RWServlet.class
- ¥ WEB-INF/web.xml

Den Installationsordner kann an man an eine beliebige Stelle auf dem Zielrechner kopiert werden. Die Datei `upload.xml` wird in den Ordner `/<tomcat-install-folder>/webapps` kopiert.

ffnen Sie die Datei `upload.xml` und ndern sie den Parameter `docBase`. Die `docBase` sollte auf den Ordner zeigen, den Sie als Installationsordner gew hlt hatten, z.B. `/Users/Shared/upload`, dann lautet `docBase = "/Users/Shared/upload"`.

Der Parameter `basePath` in der Datei web.xml in ihrem Installationsordner `upload` kann noch angepasst werden, z. B. wenn alle Uploads im /tmp Ordner landen sollen lautet der basePath /tmp. Die Voreinstellung lautet:

```
<param-name>basePath</param-name>

<param-value>/tmp </param-value>
```

Wenn Sie nun einen Browser starten und in die URL-Adress-Zeile `<servername>:<serverport>/upload/up` eintragen, kann das Servlet angesprochen werden.

```
Bsp.: http://localhost:8080/upload/up.
```

Expert modus

Im ersten Schritt m ssen Sie einen neuen Context erstellen. Hierzu wird die `server.xml` Datei im Ordner `<tomcat-installations>/conf` editiert. Es wird ein neues `Context - Tag` eingef gt, in der gleichen Hierarchiestufe wie die anderen `Context - Tags`.

```
<Context path="/rogo" docBase="/Users/Shared/upload" debug="0"

                reloadable="true" crossContext="true">

          <Logger className="org.apache.catalina.logger.FileLogger"

                    prefix="localhost_rogo_log." suffix=".txt"
```

```
timestamp="true"/>
          <Ejb   name="ejb/EmplRecord" type="Entity"
                 home="com.wombat.empl.EmployeeRecordHome"
               remote="com.wombat.empl.EmployeeRecord"/>
          </Context>
```

Der `path` Parameter ist die URI, unter der der Ordner, der im `docBase` Parameter angegeben wird, angesprochen werden kann. Wenn also z.B. `http://myhost:8080/rogo/` im Browser angegeben wird, liefert der Browser den Inhalt des Ordners, oder wenn eine `index.html` existiert, wird diese angezeigt. Damit Tomcat das Servlet findet, muss in dem Ordner ein WEB-INF Ordner angelegt werden.

In diesen legt man eine web.xml Datei an mit folgendem Inhalt:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2/
/EN"
  "http://java.sun.com/j2ee/dtds/web-app_2.2.dtd">
<web-app>
<display-name>Upload Servlet</display-name>
    <description>
Uploads files or directorys via PUT Requests
    </description>


<servlet>
<servlet-name>Upload</servlet-name>
<servlet-class>RWServlet</servlet-class>
<init-param>
<param-name>basePath</param-name>
<param-value>/Users/Shared/rogo</param-value>
</init-param>
</servlet>
 <servlet-mapping>
        <servlet-name>invoker</servlet-name>
        <url-pattern>/servlet/*</url-pattern>
    </servlet-mapping>
<servlet-mapping>
        <servlet-name>Upload</servlet-name>
        <url-pattern>/up/*</url-pattern>
```

```
        </servlet-mapping>

    </web-app>
```

Die Details kann man in der Tomcat-Documentation nachlesen. Die Parameter `servlet-name`, sowie `basePath` sollten editiert werden.
In dem WEB-INF Ordner wird der Ordner `classes` angelegt. Die Datei `RWServlet.class` wird dann in diesen Ordner kopiert.

Servlet Parameter :

- `basePath` legt den Basispfad fest. Der Uploadpfad wird an diesen angeh ngt, z.B. basePath ist /Users/myPictures und Uploadpfad ist /heute/tolleBilder/ dann wird der Ordner /Users/myPictures/heute/tolleBilder/ angelegt.

Das Servlet kann dann unter `/<Context>/up/`?....
angesprochen werden.
Das Servlet RWServlet.class muss in den WEB-INF/classses Ordner kopiert werden.

f: Appletparameter und installation

Das Applet hat drei Parameter:

- `baseURL` das Wurzel Verzeichniss wo die Dateien oder Ordner angelegt werden.
- `servletURL` die URL f r das Servlet oder CGI-Programm was den PUT -Request verarbeitet.
- destinationFolder der Ordnername der Anstelle des lokalen Ordnernamens verwendet werden soll.

Die ersten beide m ssen angegeben werden. Der ditte Parameter ist optional.
Beispiel einer Applet HTML-Datei.

```
<APPLET
CODE     = "ServerWriter.class"
ARCHIVE="upload.jar"
NAME     = "Upload Applet"
  WIDTH    = 160
  HEIGHT   = 25
  HSPACE   = 0
  VSPACE   = 0
  ALIGN    = middle>
  <PARAM NAME="baseURL" VALUE="/heute/">
  <PARAM NAME="servletURL" VALUE="/upload/up">
</APPLET>
```

Das Servlet muss nicht notwendiger weise vom selben Host wie das Applet stammen. In diesem Fall muss aber die Komplette URL angegeben werden, z.B. `http://www.myhost.de/servlet/MyServlet/`

Der Jar-File, der das Applet enth lt, muss im selben Verzeichnis sein wie die HTML Datei.