INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Design und Implementierung von Shared WebDisk

Norbert Hoene

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Design und Implementierung von Shared WebDisk

Norbert Hoene

Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Thomas Niedermeier (LRZ) Dr. Helmut Reiser (LRZ)
Abgabetermin:	16. April 2008

Hiermit versichere ich, dass ich das vorliegende Fortgeschrittenenpraktikum selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 16. April 2008

. . . . . .

(Unterschrift des Kandidaten)

Abstract

Es gibt immer wieder Situationen, in denen fremde Personen und Gesellschaften digitale Daten mit dem Münchener Wissenschaftsnetz (MWN) austauschen wollen. Dieses Vorgehen konnte bislang nur manuell, zum Beispiel durch das Versenden von E-Mails oder Postpaketen, durchgeführt werden. Da jedoch dieses manuelle Vorgehen nicht nur Mehrkosten verursacht, sondern auch eine längere Durchlaufzeit benötigt, sollte eine Lösung gefunden werden, die das Austauschen der Daten beschleunigt.

Mit dieser Problemstellung befasst sich im folgenden diese Ausarbeitung. Dazu wird die Entwicklung einer Webanwendung mit dem Namen Shared WebDisk (SWD) beschrieben, die den digitalen Datenaustausch mit externen Personen und Gesellschaften ermöglichen soll. Hierzu legt die Anwendung für den Zugriff auf das MWN automatisch eine neue temporäre Benutzerkennung im Active Directory an. Außerdem erstellt sie für diesen Zweck extra ein eigenes Verzeichnis mit speziellen Zugriffsrechten, in dem dann die internen und externen Benutzer in einem zuvor definierten Zeitrahmen ihre Daten miteinander austauschen können. Dieser Austausch kann dann mit der Webanwendung IntegraTUM WebDisk geschehen, das diese Funktionalität bereitstellt.
Da natürlich jede temporäre Benutzerkennung auch ein potentielles Sicherheitsrisiko im MWN darstellt, müssen diese Kennungen nach einer gewissen Zeit wieder vom System entfernt werden. Hierzu wird ein weiteres Programm entwickelt, welches zu vorher definierten Zeitpunkten automatisch gestartet wird. Dieser Scheduler entfernt dann die temporären Accounts und Verzeichnisse wieder aus dem MWN, so dass die Sicherheit des Netzwerks weiterhin gewährleistet bleibt.

Inhaltsverzeichnis

1 Einleitung

In der heutigen Zeit ist eine Kommunikation mit anderen Geschäftspartnern wie Hochschulen, Firmen und externen Personen ein wichtige Aufgabe im Leibnitz-Rechenzentrum. Dabei müssen sehr oft auch digitale Daten ausgetauscht werden, die für das Weiterarbeiten am LRZ sowie für die externen Personen und Gesellschaften benötigt werden. Dieses Austauschen der digitalen Daten konnte bis jetzt nur händisch, wie zum Beispiel durch das Versenden der Daten in einem Paket oder durch E-Mail, durchgeführt werden, was nicht nur zu einer erheblich längeren Laufzeit für den Datenaustausch führte, sondern auch zu höheren Kosten.

Aus diesen Gründen wurde überlegt, wie man einen reibungslosen Daten- und Informationsaustausch zwischen dem MWN und externen Personen realisieren könnte. Daher ist das Ziel dieses Projekts die Beschleunigung des Datenaustausch mit externen Personen. Um dieses zu erreichen wird eine Anwendung entwickelt, die zum einem ein spezielles Zugangskonto im Münchner Wissenschaftsnetz (MWN) einrichtet und weiterhin ein Verzeichnis auf einem Dateiserver erstellt, was letztendlich zum Datenaustausch verwendet werden kann.
Für den Zugriff der externen Personen auf dem Dateiserver kann dann eine weitere Anwendung mit dem Namen *IntegraTUM WebDisk* hergenommen werden, mit der bereits Mitarbeiter und Studenten anhand ihrer Benutzerkennung auf ihre Daten innerhalb des MWN zugreifen, was letztendlich über eine Webseite geschieht.

Das zu entwickelnde Projekt muss also die folgenden Anforderungen erfüllen:

- Es soll ein Datenaustausch mit externen Personen/Gesellschaften ermöglicht werden
- Der Umfang der Daten kann beliebig groß sein
- Die Sicherheit des MWN darf nicht beeinträchtigt werden
- Die Bedienung für den Datenaustausch muss leicht durchzuführen sein

2 Analyse

Um das oben genannte Problem zu lösen wird also eine Anwendung entwickelt, die ein neues Benutzerkonto im Münchner Wissenschaftsnetz erstellt und zusätzlich ein neues Verzeichnis einrichtet, welches dann für den externen Datenaustausch verwendet werden kann. Dabei darf jedoch die neue Zugangskennung aus Sicherheitsgründen nur für einen gewissen Zeitraum zur Verfügung stehen, bevor sie dann letztendlich gesperrt und anschließend wieder aus dem System entfernt wird. Anhand dieses eingerichtete Benutzerkonto kann dann der externe Anwender auf das erstellte Verzeichnis lesend oder schreibend zugreifen und so den Datenaustausch vornehmen.
Damit jedoch den Speicherplatzbedarf nicht unnötig verschwendet wird, muss nach einer gewissen Zeitspanne das für den externen Benutzer angelegte Verzeichnis wieder vom Dateiserver gelöscht werden. Der Zugriff selber auf das eingerichtete Verzeichnis erfolgt mittels des bereits erwähnten Tools *IntegraTUM WebDisk*, welches ebenfalls eine Webanwendung darstellt. Diese zu entwickelnde Anwendung soll den Namen *Shared WebDisk* (SWD) tragen.

2.1 Funktionsweise der Shared WebDisk

Im folgenden wird der Ablauf der SWD mit Hilfe der IntegraTUM noch einmal WebDisk näher erläutert.

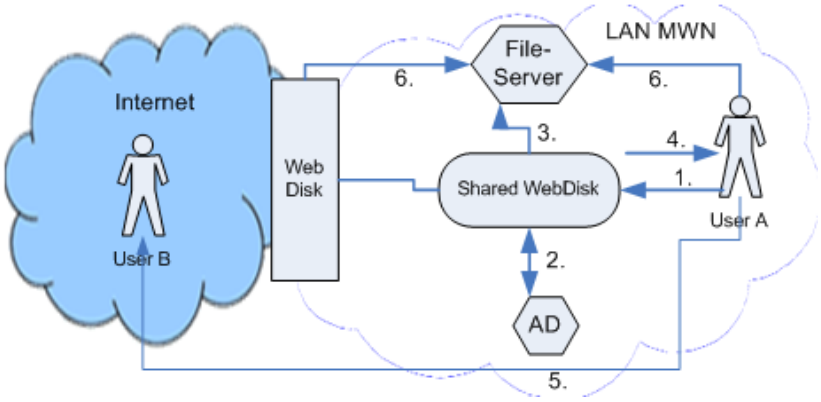


Abbildung 2.1: Zu implementierendes Szenario

Der externe Benutzer (User B) möchte mit dem internen Benutzer (User A) des MWN beliebige Daten austauschen. Dazu meldet sich User A an die SWD mit seinen Benutzerdaten an (Schritt 1). Falls User A für die Benutzung der SWD autorisiert ist, wird nun über die Weboberfläche (Shared WebDisk) eine neue Kennung für den externen User B erstellt (Schritt

2). Als nächstes (Schritt 3) wird ein Verzeichnis auf dem Fileserver mit den entsprechenden Lese- und Schreib-rechten erzeugt, auf welches der User A und B Zugriff haben. Die notwendigen Zugangsinformationen, die in Schritt 2 und 3 erzeugt wurden, werden nun dem User A mitgeteilt (Schritt 4). In Schritt 5 gibt nun der User A die Zugangsdaten an den externen User B. Dieser Austausch kann zum Beispiel per E-Mail oder Telefon geschehen. Jetzt kann User A das Verzeichnis, welches in Schritt 3 erzeugt wurde, mit beliebigen Daten füllen (Schritt 6). Im letzten Schritt kann der User B sich nun mit den von User A erhaltenen Zugangsdaten über die WebDisk-Anwendung anmelden und auf das Verzeichnis auf dem Fileserver zugreifen.

Um also eine solche Anwendung zu entwickeln, wird das Hauptproblem in zwei Teilaufgaben zerlegt. Das erste Teilproblem beschäftigt sich mit der Erstellung eines Benutzerkontos, sowie das Einrichten einer neuen temporären Kennung für den externen Benutzer.
Um sämtlichen Anwendern des MWN diesen Dienst zur Verfügung zu stellen, wird eine Weboberfläche entwickelt, die diese Funktionen bereitstellt. Um jedoch Missbrauch von internen Anwendern des MWN vorzubeugen, muss darauf geachtet werden, dass nur bestimmte, berechtigte Personen diesen Dienst verwenden können. Dieser Teil wird im weiteren Verlauf der Ausarbeitung *Shared WebDisk* (SWD) genannt.

Das zweite Problem beschäftigt sich hierbei mit dem Löschen der temporären Verzeichnisse sowie mit dem Säubern von abgelaufenen temporären Benutzerkonten. Dafür wird ein zusätzliches Programm entwickelt, das auf der Kommandozeile ausgeführt wird. Dieses wird im weiteren Verlauf Scheduler genannt.

2.2 Vorüberlegungen

Für die Realisierung dieses Projekts müssen vorab noch wichtige Fragen geklärt werden. Diese sollen nun im weiteren Verlauf zunächst beantwortet werden.

Wahl der Programmiersprache

Da die SWD später auf einem Microsoft Internet Information Server läuft, bietet sich die Möglichkeit an, die zu programmierende Applikation mit C-Sharp zu entwickeln. Diese von Microsoft konstruierte Programmiersprache baut auf dem ASP.NET Framework auf und zählt heutzutage zu den modernsten Programmiersprachen in der Softwareentwicklung. Der Scheduler hingegen soll in der Programmiersprache *Visual Basic Script*[1] entwickelt werden, da die Systembetreuer diese Sprache beherrschen und sie zu späterer Zeit etwaige Anpassungen selber vornehmen können.

Benötigte Soft- und Hardware

Um eine genaues Abbild mit der späteren Produktivumgebung zu simulieren, wurden für diesen Zweck extra mehrere verschiedene Systeme eingerichtet, welche jeweils eine andere

[1] Von MS entwickelte Skriptsprache - http://de.wikipedia.org/wiki/Visual_Basic_Script

Aufgabe besaßen. Dadurch konnte nun die SWD bzw. der Scheduler entwickelt und ausführlich getestet werden, ohne das produktive System zu dabei zu stören.
Im folgenden wurden folgende Komponenten benötigt:

- Windows 2003 Server mit Active Directory und eigene Domain
- Dateiserver für die temporären Verzeichnisse
- Testrechner mit MS IIS, .NET Framework 2.0 und C-Sharp Entwicklungsumgebung

2.3 Anforderungsanalyse

Des Weiteren war es sinnvoll, sich bereits im Vorfeld mit einigen grundlegenden Problemen auseinander zu setzen, um eine zielgerichtete Entwicklung der beiden Programme zu gewährleisten. Dazu wurde zunächst eine entsprechende Anforderungsanalyse durchgeführt, welche die SWD sowie der Scheduler erfüllen müssen.

2.3.1 Anforderungen an SWD

Im ersten Schritt wurde dazu ein Anforderungskatalog für die SWD entworfen. Dazu wurden zunächst die sicherheitsrelevanten Aspekte und danach die allgemeinen Anforderungen an die SWD näher betrachtet. Mit diesen vorgegebenen Richtlinien konnte später im Kapitel 3 die SWD entwickelt werden.

Sicherheitsrelevante Aspekte

Da bei der SWD tiefe Systemeingriffe vorgenommen wurden, musste hierbei besonders auf die Sicherheit geachtet werden. Dabei können getroffene Fehlentscheidungen bereits große Auswirkungen auf die Zuverlässigkeit der Sicherheit des MWN haben, so dass die folgenden Vorüberlegungen eine wichtige Information darstellte.

Verwendung nur für ausgewählte Anwender

Um nicht jeden internen Benutzer die Möglichkeit zu geben, temporäre Shares anzulegen, muss die SWD eine Authentisierung und Autorisierung zur Verfügung stellen, in der nur die berechtigten Personen die Benutzung der SWD erlaubt wird. Aus diesem Grund musste eine entsprechende Lösung entwickelt werden, die es ermöglicht, auf einfache Weise festzulegen, welcher Benutzer die erforderliche Berechtigung besitzt.

Eigener Benutzeraccount für die Ausführung der SWD

Da die SWD Schreibrechte im Active Directory sowie für das Dateisystem benötigt, sollte dafür extra ein eigenes Benutzerkonto eingerichtet werden, welches nur die notwendigen Berechtigungen besitzt, die es letztendlich auch benötigt.

Gültigkeit der temporären Benutzerkennung

Ein wichtiger Punkt ist die Erstellung einer gültigen Benutzerkennung in der Domäne. Hierbei muss besonders auf die Sicherheit geachtet werden, so dass kein Missbrauch mit dieser Kennung betrieben werden kann. Dazu sollte der neue temporäre Benutzeraccount nur für

eine gewisse Zeitspanne aktiviert bleiben, die der Systembetreiber der SWD beliebig festlegen kann. Mit dieser Aktion kann ein externer Benutzer sich nur in der zuvor definierten Zeitspanne an der SWD mit seiner temporären Kennung anmelden.

Begrenzung der Einrichtung von Shares

Eine weitere Maßnahme zur Beibehaltung der Sicherheit des MWN, war die Beschränkung der anzulegenden temporären Verzeichnisse für die internen Benutzer. Mit dieser Richtlinie sollte verhindert werden, dass ein interner Benutzer unzählige neue temporäre Benutzeraccounts mit ihren dazugehörigen Verzeichnissen anlegt und so eine Unmenge an potentiellen Löchern im MWN verursacht.

Generierung sicherer Passwörter

Damit keine trivialen Passwörter durch den internen Benutzer gewählt werden können und so die Sicherheit des MWN beeinträchtigt wird, musste ein Algorithmus entworfen werden, der automatisch sichere Passwörter nach den Vorgaben der Administratoren erstellt.

Protokollierung der getätigten Aktionen

Das Mitloggen von Aktionen ist ebenfalls eine wichtige Aufgabe um die Sicherheit auf einem System zu gewährleisten. Daher war eine weitere Maßnahme das Mitprotokollieren aller wichtigen Informationen. Dieses sollte entweder im Windows EventLog oder in einer Logdatei geschehen.

Allgemeine Aspekte

Neben den sicherheitsrelevanten Punkten existieren natürlich auch einige andere Aspekte die für die Durchführung des Projekts erforderlich sind.

Design der Webseite

Da das LRZ bereits ein eigenes Design für ihre Webseiten entworfen hatte, sollte die SWD ebenfalls an dieses Aussehen der anderen Webseiten angepasst werden. Dazu sollte von einer beliebigen Seite des LRZ das Seitenlayout extrahiert und in die SWD integriert werden.

Benutzerfreundlichkeit

Um die SWD möglichst vielen Anwendern nahe zu bringen, war die Benutzerfreundlichkeit ein ein weiterer wichtiger Punkt bei der Entwicklung der einzelnen Webseiten. Dazu zählte auch die Darstellung der einzelnen Webseiten in deutsche Sprache.

Zusammenfassen aller Einstellungen in einer Konfigurationsdatei

Um ein leichtes Abändern von Einstellungsmöglichkeiten zu gewährleisten, sollten alle Einstellungen, die bei der Entwicklung der SWD verwendet werden, in einer Konfigurationsdatei zusammengefasst werden. Dadurch sollte ein Abändern des Quellcodes vermieden werden. Um zu verhindern, dass jede Person diese Datei abändert, sollten die Zugriffsrechte soweit beschränkt werden, dass nur die berechtigten Administratoren einen Zugriff auf diese Datei haben.

2.3.2 Anforderungen an Scheduler

Auch bei dem Scheduler müssen im Vorfeld noch wichtige Details für eine korrekte Implementierung abgesprochen werden. Dabei müssen wiederum einige Sicherheitsaspekte sowie allgemeine Aspekte für die Umsetzung betrachtet werden.

Sicherheitsrelevante Aspekte

Im folgenden werden alle wichtigen Punkte, die für die Durchführung entscheidend sind, noch einmal aufgeführt und näher erläutert.

Zugriffsrechte auf temporäres Verzeichnis entfernen

Um einen Zugriff auf das temporäre Verzeichnis nach Ablaufen der temporären Benutzerkennung zu verhindern, müssen alle Zugriffsrechte vom internen und externen Benutzer für dieses Verzeichnis entfernt werden. Dadurch wird erreicht, dass keine Daten mehr ausgetauscht werden können.

Löschen nicht mehr benötigter Benutzeraccounts

Eine wichtige Aufgabe des Scheduler ist das Löschen von abgelaufenen temporären Benutzerkonten damit ein Ausbruch des Systems vermieden wird. Das Entfernen der einzelnen Kennungen darf jedoch erst nach einer gewissen Schonfrist geschehen, da es möglich ist, dass ein temporärer Benutzer noch einmal einen kurzen Zugang benötigt.

Alte Verzeichnisse löschen

Eine weitere wichtige Funktion ist das Löschen von alten, nicht mehr benötigten temporären Verzeichnissen. Das Löschen dieser Verzeichnisse darf dabei ebenfalls erst nach einer gewissen Schonfrist erfolgen, da es in der Praxis häufig vorkommt, dass zwischen dem internen und externen Benutzer noch einmal kurzfristig Daten ausgetauscht werden müssen, da der eigentliche definierte Zeitraum für diese Aktion nicht ausreichte.

Eigener Benutzer zur Ausführung des Scheduler

Ein weiterer Sicherheitsaspekt ist die Ausführung des Scheduler nur mit so vielen Rechten, wie er minimal benötigt (Need-to-know-Prinzip). Dafür muss ein weiterer Benutzer im System angelegt werden, der nur für die Ausführung des Scheduler verantwortlich ist. Alle sonstigen Optionen, wie das Anmelden an der Domäne mit dieser Kennung müssen deaktiviert werden.

Protokollierung aller durchgeführten Aktionen

Um alle Änderungen am Dateisystem und im Active Directory nachvollziehen zu können, ist es ebenfalls wichtig, dass alle durchgeführten Transaktionen vom Scheduler mitprotokolliert werden. Dieses kann wahlweise im Windows EventLog oder in einer extra dafür vorgesehenen Logdatei passieren.

Allgemeine Aspekte

Bei der Ausführung des Scheduler sind ebenfalls einige Punkte zu betrachten.

Zeitbasierte Ausführung

Damit nicht der Scheduler immer wieder von Hand gestartet werden muss, wurde empfohlen, dass diese Aufgabe als zeitbasierter Job (Cronjob) ausgeführt wird. Um dieses zu erreichen, darf der Scheduler nach dem Starten keine weiteren Eingaben vom Benutzer mehr entgegennehmen. Es müssen also am Anfang des Aufrufs vom Scheduler alle benötigten Informationen für die Ausführung des Programms vorhanden sein. Zum zeitbasierten Starten des Scheduler kann dabei auf betriebssystemeigene Hausmittel zurückgegriffen werden.

3 Implementierung

Da das Hauptproblem auf zwei Anwendungen aufgeteilt wurde, wird im ersten Schritt aufgezeigt, wie die Shared WebDisk (SWD) entworfen und entwickelt wurde. Dabei wird die Vorgehensweise dargestellt, wie die SWD in den einzelnen Hauptaufgaben implementiert wurde. Aber es wird auch auf die Kernprobleme bei der Programmierung eingegangen, die dabei aufgetreten sind. Im zweiten Teil dieses Kapitels wird dann die Entwicklung des Scheduler mit den dort aufgetretenen Hauptproblemen dargestellt. Dabei wird ebenfalls aufgezeigt, wo die Skriptsprache VBScript bei der Entwicklung des Scheduler an ihre Grenzen gestoßen ist und wie man dennoch mit hauseigenen Hilfsmitteln eine gute Lösung erreichen konnte.

3.1 Entwicklung der SWD

Um die SWD mit den genannten Kriterien aus Kapitel 2 zu entwickeln ist es sinnvoll sich bereits am Anfang der Entwicklung, über den Ablauf der SWD, ein klares Bild zu machen.

Zunächst muss auf einer Logon-Seite, zu überprüfen werden, ob der interne Benutzer die notwendigen Berechtigungen für das Anlegen eines temporären Verzeichnisses (Share) besitzt. Bei einer erfolgreicher Anmeldung soll dann der Benutzer zur Hauptseite (Sharing-Seite) der SWD weitergeleitet, in der er nun alle wichtigen Informationen zum Erstellen eines Shares eintragen kann. Nach Beendigung dieses Vorgangs sollen dem Benutzer auf einer Summary-Seite noch einmal alle wichtige Daten, wie temporärer Benutzername und Passwort angezeigt werden. Nach diesen Schritten kann der Benutzer sich wieder von der SWD abmelden.

Es müssen also für das obige Vorgehen im folgenden drei Webseiten implementiert werden, die im ganzen die SWD bilden:

- Logon-Seite
- Sharing-Seite
- Summary-Seite

Dieser Ablauf ist nun in Abbildung 3.1 noch einmal in grober Funktionsweise grafisch dargestellt. Dabei ist zu erkennen, wie der interne Benutzer durch die einzelnen Webseiten geführt wird, bis er am Ende alle notwendigen Informationen noch einmal in einer Zusammenfassung angezeigt bekommt. Die genaue Implementierung wird im Abschnitt 3.1.1 ausführlich dargestellt.

Ausführen der SWD mit speziellen Benutzerrechten

Wie bereits in der Anforderungsanalyse im Kapitel 2.3.1 beschrieben wurde, darf die SWD aus Sicherheitsgründen nur mit speziellen Benutzerrechten arbeiten. Dieses ist notwendig, da

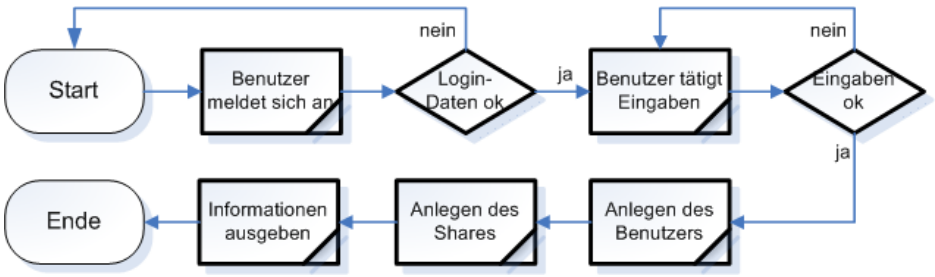


Abbildung 3.1: Grober Ablauf der SWD

sie für die Arbeiten im Active Directory sowie beim Anlegen neuer Verzeichnisse bestimmte Schreibrechte benötigt.
Um das zu erreichen, wurde eine neue Benutzerkennung mit dem Namen *swd_user* und den oben angegebenen Rechten im Active Directory angelegt. Danach wurden in der Konfigurationsdatei *Web.config* der SWD spezielle Flags für die Ausführung der SWD hinzugefügt.

```
01 <identity impersonate="true"
02     userName="registry:HKLM\SOFTWARE\ShareWebDisk
           \identity\ASPNET_SETREG,userName"
03     password="registry:HKLM\Software\ShareWebDisk
           \identity\ASPNET_SETREG,password"/>
```

Damit die Zugangsdaten vom *swd_user* nicht direkt in der Konfigurationsdatei im Klartext enthalten sind, sind diese aus Sicherheitsgründen in der *Registry* vom Windows Betriebssystem mit entsprechenden Zugriffsrechten abgelegt.

Extrahieren des Webseiten-Layouts

In einem weiteren Schritt, noch vor der Erstellung der Webseiten, musste zunächst das passende Layout von einer beliebigen Seite des LRZ extrahiert werden, damit es bei der späteren Entwicklung der einzelnen Seiten hineinintegriert werden konnte. Hierzu wurde einfach eine beliebige Seite des LRZ besucht, die das vorgegebene Design enthält und aus dem Quelltext der Webseite heraus kopiert, so dass dieser Schritt keine große Hürde darstellte.

Nachdem nun alle Maßnahmen für eine korrekte Implementierung durchgeführt wurden, kann nun die Entwicklung der einzelnen Webseiten näher beschrieben werden.

3.1.1 Logon-Seite

Die Logon-Seite besteht aus einer klassischen Benutzername/Passwort Überprüfung. Dabei darf sich jeder Benutzer von einer vorgegebenen Domäne[1] bei der SWD anmelden, wenn dieser extra in einer vorher eingerichteten Gruppe (*swd_group*) eingetragen ist. Dabei kann es auch möglich sein, dass der Benutzer kein direktes Mitglied der *swd_group* ist, sondern dass er einer Gruppe angehört, die als Gruppe selbst wieder ein Mitglied der Gruppe *swd_group* ist. Dieses stellte jedoch ein Problem dar, da dafür extra ein rekursiver Algorithmus entwickelt werden musste, der weiter unten im Abschnitt 3.1.1 näher erläutert werden soll.

[1] Eine Domäne ist ein lokaler Sicherheitsbereich mit einer zentralen Verwaltung der Ressourcen.

Um nun einen derartigen Login-Mechanismus zu entwickeln, kann man das Problem in drei Teile aufspalten:

- 1. Authentisierung gegenüber des Active Directory
- 2. Autorisierung gegenüber der SWD
- 3. Setzen der benötigten Werte für das spätere Anlegen eines temporären Benutzers

Authentisierung gegenüber des Active Directory

Um sich, wie bereits oben erwähnt, mit Hilfe seines Benutzernamen und das dazugehörige Passwort an der SWD anzumelden, muss zunächst eine Verbindung zum Active Directory[2] aufgebaut werden, damit im Nachhinein die Autorisierung gegenüber der SWD überprüft werden kann. Da dieses eine Standard-Aufgabe für viele Webapplikationen darstellt, existiert bereits im .NET-Framework eine entsprechende Klasse *DirectoryEntry*, welche sich an einen Active Directory Server bindet und dabei den Benutzername und das Passwort überprüft.

Verwendung der Klasse DirectoryEntry zum Überprüfen von Benutzername und Passwort

```
DirectoryEntry entry =
        new DirectoryEntry(AD-Server, Benutzername, Passwort);
```

Diese einzelne Zeile reicht für die SWD aus, um eine einfache Benutzername/Passwort Überprüfung zu realisieren und sich gegen das Active Directory zu authentisieren. Im nächsten Schritt muss sich nun der Benutzer gegen die Webanwendung autorisieren.

Autorisierung gegenüber der SWD

Da die Autorisierung des Benutzers gegenüber der SWD auch Untergruppen von der *swd_group* den Zugang erlauben soll, muss dafür nun ein zusätzlich Algorithmus entwickelt werden. Dazu werden im ersten Schritt die die Beziehungen zwischen den einzelnen Gruppen näher betrachtet, um dieses Verhalten besser zu verstehen.

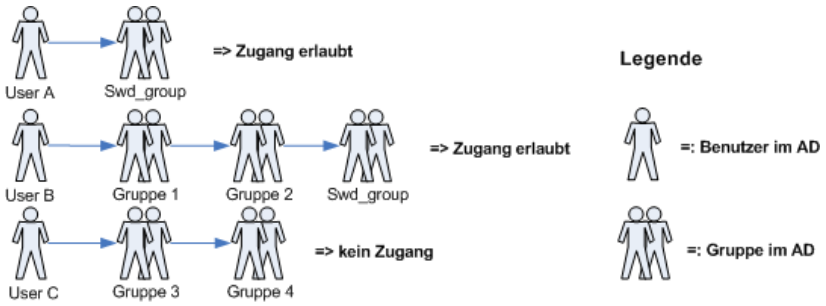


Abbildung 3.2: Zugangsberechtigungen bei Untergruppen von *swd_group*

Die Abbildung 3.2 zeigt, dass der User A Zugang zur Webanwendung hat, da dieser direkt der Gruppe *swd_group* angehört. Im zweiten Fall hat User B ebenfalls die nötige Berechtigung,

[2]Verzeichnisdienst von Microsoft, in dem sämtliche Informationen, wie Benutzerdaten abgelegt sind

da er Mitglied der Gruppe 1 ist, welche wiederum Mitglied der Gruppe 2 ist und diese ist Mitglied der Gruppe *swd_group*. Es handelt sich also hierbei um eine indirekte Mitgliedschaft von User B zur *swd_group*, welche ebenfalls erlaubt ist. Im dritten Fall hingegen ist weder die Gruppe 3, noch die Gruppe 4 Mitglied der *swd_group*, weswegen der User C hier keine Zugangsberechtigung erhält.

Nachdem nun das Problem analysiert wurde, kann nun ein rekursiver Algorithmus entwickelt werden, der das Verhalten von Abbildung 3.2 in einer Programmiersprache umsetzt.

C# Code zur rekursiven Suche der Gruppenmitgliedschaft

```
01 public bool isMemberOf(string groupname, string accountname)
02 {
03     StringCollection membersof = this.GetMemberOf(accountname);
04     foreach (string entry in membersof)
05     {
06         /* swd_group wurde in den Untergruppen gefunden */
07         if (groupname.Equals(entry))
08             return true;
09         /* die Untergruppe ist wieder eine neue Gruppe */
10         if (isMemberOf(groupname, entry))
11             return true;
12     }
13     /* wenn nirgends ein Eintrag gefunden wurde,
               dann gib false zurück */
14     return false;
15 }
```

In Zeile 03 werden alle *memberOf*-Attribute vom jeweiligen Benutzernamen ermittelt. Dabei beinhaltet das *memberOf*-Attribut alle Namen der angehörigen Gruppen, in denen der Benutzer ein Mitglied ist. Diese Gruppennamen werden dann in eine *Collection* gespeichert, was aus anderen Programmiersprachen einem Array ähnelt. (Anhang 5.1 zeigt ein Beispiel, wie ein solcher Personeneintrag im Active Directory hinterlegt ist.) Danach wird iterativ diese Liste auf den zu findenden Gruppennamen (swd_group) untersucht. Da ein memberOf-Eintrag wiederum eine Gruppe sein kann, wird die Funktion in Zeile 10 erneut aufgerufen. Falls jedoch der gesuchte Gruppenname (swd_group) gefunden wurde (Zeile 07), liefert die Funktion *isMemberOf* ein true zurück. Sollte der entsprechende Gruppenname nicht gefunden werden, so gibt die Funktion *isMemberOf* ein false zurück, welches dann die Autorisierung gegenüber der SWD scheitern lässt.

Setzen der benötigten Werte für das spätere Anlegen eines temporären Benutzers

Zum späteren Anlegen eines neuen temporären Benutzers werden außerdem noch einige weitere Informationen benötigt, um den neuen Benutzer im Active Directory zu erstellen. Diese Parameter dienen aber mehr der Übersicht, wodurch zum Beispiel ein Administrator der Domäne jederzeit einsehen kann, wer welches Benutzerkonto eingerichtet hat.

Die folgenden Informationen werden dabei vom eigenen Benutzerprofil für die spätere Erstellung des neuen temporären Benutzers für eine bessere Identifizierung herangezogen.

- Vorname des Erstellers (Attribut firstName)
- Nachname des Erstellers (Attribut lastName)
- Telefonnummer des Erstellers (Attribut telephoneNumber)
- E-Mail Adresse der Erstellers (Attribut emailAdress)

Diese oben genannten Benutzerinformationen werden dafür extra in einer eingerichteten globalen Variablen *sessionUser* abgelegt, damit im nächsten Schritt die Sharing-Seite auf diese Informationen zugreifen kann.

Das endgültige Ergebnis der Logon-Seite für die SWD kann im Bild 3.3 mit Verwendung des bereit gestellten LRZ Webseiten Layouts betrachtet werden.

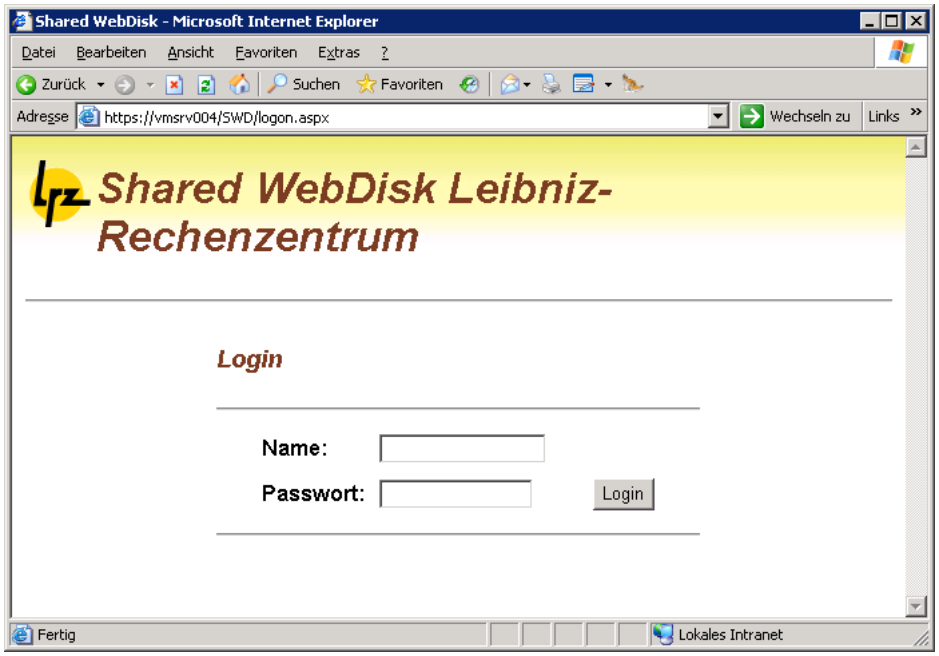


Abbildung 3.3: Die fertige Logon-Seite mit Benutzername/Passwort Authentifizierung

3.1.2 Die Sharing-Seite

Diese Seite stellt den Kern der Webanwendung dar. Hier soll der Benutzer alle wichtigen Einstellungen für die Erzeugung und der späteren Löschung des temporären Ordners vornehmen können. Um eine große Akzeptanz für die Benutzung dieser Webanwendung zu bekommen, ist ebenfalls die Benutzerfreundlichkeit dieser Seite ein sehr wichtiger Punkt. Aber auch diverse Sicherheitsaspekte wie zum Beispiel das unzählige Erstellen von temporären Benutzeraccounts (also DoS-Attacken) oder das Mitprotokollieren sämtlicher getätigter Aktionen muss diese Seite entsprechend behandeln.

Die Erstellung einer solchen Webseite kann dabei in folgende Aufgaben zerlegt werden:

- 1. Erstellen des neuen temporären Benutzerkontos
- 2. Erstellen des temporären Verzeichnisses
- 3. Zusammenfassen aller getätigten Aktionen
- 4. Durchgeführte Aktionen protokollieren

Erstellen des neuen temporären Benutzerkontos

Damit sich der externe Benutzer in das Netzwerk des LRZ für den Datenaustausch einloggen kann, benötigt er eine entsprechende temporäre Benutzerkennung, welche nach einer bestimmten Zeit wieder automatisch (mit Hilfe des später beschriebenen Scheduler) aus dem Active Directory entfernt wird. Dazu müssen zum Anlegen eines neuen Benutzerkontos im Active Directory einige Pflicht- und optionale Parameter gesetzt werden.
Um diese besser zusammenfassen zu können, wurde ein neuer Datentyp mit dem Namen *UserInfo* entworfen, der alle wichtigen Attribute zur weiteren Verwendung beinhaltet.

Im nachfolgenden Code-Auszug wird der neue Datentyp *UserInfo* dargestellt.

```
public struct UserInfo
{
    public string username;
    public string sAMAccountName;
    public DateTime accountExpires;
    public string userPrincipalName;
    public string password;
    public string initials;
    public string firstName;
    public string lastName;
    public string displayName;
    public string emailAdress;
    public string telephoneNumber;

}
```

Beim Füllen der einzelnen Werte der neuen Struktur *UserInfo* kann nun auf die zuvor beim Logon-Mechanismus gespeicherten Werte, wie *firstName*, *lastName*, *telephoneNumber* und *emailAdress* zurückgegriffen werden. (Siehe dazu Abschnitt 3.1.1). Das Setzen der anderen Parametern soll im weiteren Verlauf näher beschrieben werden.

Generierung des neuen Benutzernamens

Um nun einen temporären Benutzernamen zu erzeugen, der eine gewisse Beziehung zur SWD aufzeigt, wurde ein Algorithmus entwickelt, der dabei den zuvor in der Konfigurationsdatei *Web.config* (Anhang 5.2) gesetzten Parameter *TmpUserPrefix* verwendet. Dieser definierte

Wert (zum Beispiel *tmp_*) wird an jeder neuen Kennung am Anfang des Namens vorangestellt. Das ermöglicht nicht nur fremden Benutzern, wie zum Beispiel Administratoren zu erkennen, dass diese Kennung im Bezug zur SWD steht, sondern man kann dadurch auch nun leicht sämtliche Kennungen mit Zuhilfenahme einer weiteren Zahl eindeutige Kennungen generieren.

Im unteren Listing sind dazu drei Beispiele zur Veranschaulichung noch einmal dargestellt.

```
tmp_0; tmp_3; tmp_4
```

Um dabei zu vermeiden, dass eine Benutzerkennung doppelt angelegt wird, muss zunächst im Active Directory eine entsprechende freie Kennung ermittelt werden. Dazu dient der folgende Codeauszug.

Ermitteln einer freien Benutzerkennung

```
01 for (int i = 0; i < 300; i++)
02 {
03     adTmpSearch.Filter = "(sAMAccountName=" +
                        tmpUserPrefix + i + ")";
04     SearchResult adResult = adTmpSearch.FindOne();
05     if (adResult == null)
06     {
07         // set the new user index
08         return (tmpUserPrefix + i);
09     }
10 }
```

Der Algorithmus baut zunächst (in Zeile 03) eine Zeichenkette für einen Filter zusammen. Danach wird anhand dieses Filters eine Abfrage im AD durchgeführt (Zeile 04). Existiert nun diese gesuchte Zeichenkette (also die Benutzerkennung) nicht in dieser Domäne (Zeile 05), liefert die Suchanfrage einen *null-Wert* zurück, welcher dann die ermittelte Kennung an eine Überfunktion zurückgibt (Zeile 08).

Die aus diesem Algorithmus erzeugte Kennung kann nun den Parametern *username*, *sAMAccountName*, *displayName* und in erweiterter Form *userPrincipalName* der Struktur *UserInfo* übergeben werden.

Passworterzeugung des temporären Benutzers

Die automatische Erzeugung eines Passwortes sollte möglichst komplex sein und sich auf eine Länge von acht Zeichen beschränken. Hierbei konnte dem Ideenreichtum freien Lauf gelassen werden, so dass dieses Problem mit Hilfe eines Zufallsgenerators gelöst werden konnte.

Bei vielen Testdurchläufen funktionierte dieses Standardverfahren sehr gut, aber dennoch ist bei einigen Durchläufen die SWD mit einer eher ungenauen Fehlermeldung wie *'Fehler beim Anlegen des Benutzers'* abgebrochen.
Erst nach vielem Suchen und Debuggen mit dem Tool *MS Visual Studio 2005* konnte herausgefunden werden, dass die Passwortgenerierung bei der SWD auch Passwörter erstellt,

die das Active Directory jedoch als unsicher einstuft. Daher musste eine weitere Funktion geschrieben werden, die nach der Generierung überprüft, ob das Passwort als sicher eingestuft werden kann.

Also muss die Generierung des Passworts so oft wiederholt werden, bis es den Ansprüchen der Komplexität an das Active Directory entspricht. Der folgende kurze Code-Auszug soll dieses noch einmal verdeutlichen.

Generierung eines gültigen Passwortes in C#

```
while (not ADUser.IsPasswordValid(newUserInfo.password,
                    PasswordRules.All, null))
{
    newUserInfo.password = newUser.GeneratePassword(8);
}
```

Gültigkeit der temporären Benutzerkennung

Wie bereits im Kapitel 2.3.1 erwähnt wurde, soll eine eingerichtete temporäre Benutzerkennung nur maximal für eine gewisse Zeitspanne verfügbar sein. Durch diese Maßnahme soll ebenfalls die Sicherheit des MWN weiterhin gewährleistet werden.
Dazu bietet das Active Directory die Möglichkeit an, den Parameter *accountExpires*, für jedes Benutzerkonto explizit zu setzen. Nach Ablauf dieses Datums wird der einzulogende Benutzer mit einer Fehlermeldung darauf hingewiesen, dass er sich nicht mehr an dieser Domäne mit den Benutzerdaten anmelden kann.

Um nun für die SWD die Zeitspanne möglichst variabel zu halten, wird ein weiterer Parameter *MaxDifferenceDays* in der Konfigurationsdatei *Web.config* verwendet. Damit auch der Benutzer, der einen temporären Account anlegt, über diesen Gültigkeitszeitraum informiert ist, wird auf der Webseite nach einmal das maximale einzustellende Datum angezeigt und bei einer eventuellen Fehleingabe nochmals auf den gültigen Zeitraum hingewiesen. Außerdem wurde am Anfang der Entwicklung ebenfalls zum Datum noch die Uhrzeit mit angegeben. Diese Funktion wurde jedoch von vielen Benutzern dieser Webanwendung nicht verwendet, so dass die Auswahl auf den Stunden beschränkt wurde.

Abbildung 3.4: Datumsauswahl bei der SWD

Erstellen des temporären Verzeichnisses

Dieses temporäre Verzeichnis dient in der späteren Anwendung der SWD als Datenaustausch zwischen dem internen und externen Benutzer. Bei der Erzeugung eines solchen temporären Verzeichnisses werden nicht nur einige Zusatzinformationen für die Verwendung bei den Benutzern benötigt, sondern es muss auch auf die Rechteverwaltung für diesen Ordner geachtet werden, damit die Sicherheit des Dateisystems auch weiterhin gewährleistet wird.

Das temporäre Verzeichnis muss also folgende Informationen für die Benutzer bereitstellen:

- Welcher interne und externe Benutzer kann dieses Verzeichnis verwenden
- Gültigkeitsdauer des Account für den temporären Benutzer
- Art des Zugriffs auf das Verzeichnis (Schreiben, Lesen)

Am Anfang wurden alle oben benötigten Informationen in den Verzeichnisnamen hineinkodiert, was aber bei den Benutzern für eine schwere Lesbarkeit des Verzeichnisnamens führte (siehe dazu auch Kapitel 3.2). Daher wurde eine neue Möglichkeit entworfen, die alle benötigten Informationen in eine Datei *directory.dat* schreibt, welche für die internen und externen Benutzer aus Sicherheitsgründen und Schutz vor Manipulation der Daten nur lesbar ist.

Inhalt der Datei directory.dat

```
TMP_USER=tmp_1
CREATOR=root
VALID_FROM=2007-03-12
EXPIRED_DATE=2007-03-14
EXPIRED_TIME=13:00
```

Dabei steht der Parameter *TMP_USER* für den externen Benutzer, der Parameter *CREATOR* für den internen Benutzer und die letzten drei Parameter geben das Intervall an, in dem der externe Benutzer die Daten mit dem internen Benutzer austauschen kann.

In dieser Variante muss nun allerdings noch überprüft werden, ob das anzulegende Verzeichnis bereits existiert oder nicht. Dazu wurde eine Methode *isExistsShare* entwickelt, die überprüft, ober der eingegebene Verzeichnisname bereits im Dateisystem vorhanden ist.

```
01 if (directory.isExistsShare(sharename))
02 {
03     throw new Exception("Verzeichnis existiert bereits.
04                 Bitte anderen Namen wählen");
05 }
06 ...
07 directory.CreateShare(sharename);
```

In Zeile 01 wird die Methode *isExistsShare* mit dem anzulegenden Sharenamen aufgerufen. Falls dieser nun bereits existiert, wird in Zeile 03 eine entsprechende Fehlermeldung generiert und an den Benutzer der SWD weitergegeben. Falls der Verzeichnisname nicht existiert, wird der neue Share in Zeile 07 angelegt.

Damit nun aber kein interner Benutzer eine beliebige Anzahl von temporären Verzeichnissen einrichten kann, und dadurch eventuelle potentielle Sicherheitslöcher im Dateisystem entstehen, wurde ein weiterer Parameter *MaxSharesForUser* in der *Web.config* hinzu genommen, der dieses Vorgehen verhindern soll. Dabei hat diese Option zum Ergebnis, dass sobald ein interner Benutzer die maximale Anzahl von eingerichteten Shares erreicht hat, mit einer entsprechenden Fehlermeldung kein weiteres temporäres Verzeichnis mehr erstellen kann, bis wieder einige *alte* Shares vom Scheduler entfernt werden.

Vererbung der temporären Ordnerrechte

Ein noch viel größeres Problem bei der Erstellung des Verzeichnisses war die vernünftige Rechteverwaltung für die temporären Ordner. Dabei werden bei der Erstellung eines neuen Ordners automatisch vom Betriebssystem die Benutzerrechte vom Elternordner, sowie vom Ersteller hinzugefügt. Um allerdings die Shared WebDisk sicher zu gestalten, ist dieses Feature für die Erstellung eines neuen Ordners nicht sinnvoll, da unter Umständen auch unberechtigte Personen auf einen beliebigen Share zugreifen könnten.

Um das Problem dennoch zu lösen, musste zuerst auf Betriebssystemebene die Vererbung für den Elternordner explizit deaktiviert werden (Ein genaues Vorgehen auf Seite des Betriebssystem ist in der Installationsanleitung im Anhang 6 noch einmal beschrieben.) In einem weiteren Schritt wird auf der Programmierseite bei der Erzeugung eines neuen Ordners explizit eine *Access Control List* (ACL) vom Ersteller hinzugefügt, welche die Berechtigung der Dateien und Ordnern festlegt. Im letzten Schritt müssen nun die anderen ACLs, welche den Share noch verwenden sollen, hinzugefügt werden. Dabei müssen beim Setzen jeder ACL noch zusätzliche Flags zur Aktivierung der Vererbung angegeben werden, da ansonsten alle neu erzeugten Dateien und Unterordner vom externen und internen Benutzer im Share automatisch keine entsprechenden Rechte bekommen, da diese am Anfang beim Betriebssystem deaktiviert wurden.

Setzen wichtiger Flags auf Programmiererseite

```
...
DirectoryInfo info = new DirectoryInfo(path);
DirectorySecurity security = info.GetAccessControl();

security.AddAccessRule(new FileSystemAccessRule(user, rights,
      InheritanceFlags.ContainerInherit, // aktiviere Vererbung
      PropagationFlags.None,
      AccessControlType.Allow));

security.AddAccessRule(new FileSystemAccessRule(user, rights,
      InheritanceFlags.ObjectInherit, // aktiviere Vererbung
      PropagationFlags.None,
      AccessControlType.Allow));
info.SetAccessControl(security);
...
```

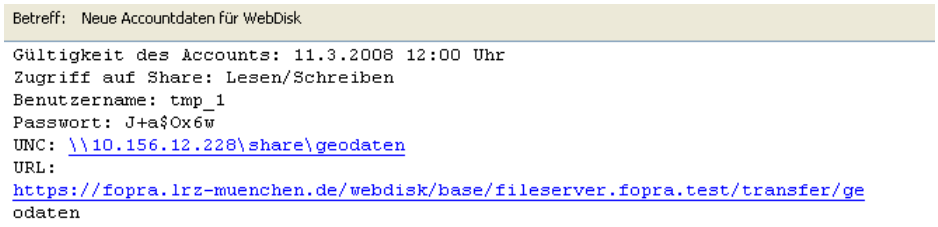
Danach werden von der SWD die richtigen Rechte für die jeweiligen Benutzer entsprechend

gesetzt, so dass kein externer und interner Benutzer fremde Daten löschen kann.

Zusammenfassen aller getätigten Aktionen

Um den internen Benutzer nun noch einmal über alle getätigten Vorgänge zu informieren, können optional sämtliche Informationen, die für die Verwendung des neuen Shares benötigt werden, noch einmal zusammenfassend an die zuvor definierten E-Mail Adressen versendet werden. Außerdem kann der Benutzer auch fremde E-Mail-Adressen angeben, an denen die Informationen ebenfalls gesendet werden sollen. Unabhängig vom Versenden der E-Mails werden auch alle wichtigen Informationen an die *Summary-Webseite* (siehe Kapitel 3.1.3) noch einmal weitergeleitet.

Die Abbildung 3.5 zeigt eine solche versendete E-Mail mit allen notwendigen Informationen für den externen Benutzer.


Betreff: Neue Accountdaten für WebDisk

Gültigkeit des Accounts: 11.3.2008 12:00 Uhr
Zugriff auf Share: Lesen/Schreiben
Benutzername: tmp_1
Passwort: J+a$Ox6w
UNC: \\10.156.12.228\share\geodaten
URL:
https://fopra.lrz-muenchen.de/webdisk/base/fileserver.fopra.test/transfer/ge
odaten

Abbildung 3.5: Ausgabe der versendeten E-Mail

Durchgeführte Aktionen protokollieren

Eine weitere Sicherheitsmaßnahme war das Protokollieren der erfolgreichen ausgeführten Aktionen von den internen Benutzern. Dazu werden in der Konfigurationsdatei *Web.config* die Parameter *LogOnLogFile*, sowie *LogOnEventLog* verwendet, um die Art des Loggings festzulegen. Je nachdem, ob diese beiden Parameter den Wert *true* oder *false* besitzen, wird die vom Benutzer durchgeführte Aktion in einer Logdatei oder in den Windows EventLog geschrieben.

Auszug aus der einer Logdatei

```
10.03.2008 12:58:23; User 'root' created for 'tmp_1' the share 'geodaten'
10.03.2008 13:02:13; User 'test' created for 'tmp_2' the share 'europe'
```

Dadurch können jederzeit berechtigte Personen, wie Administratoren, einsehen, welcher Benutzer einen Share angelegt hat.

Im Bild 3.6 ist die fertige Sharing-Seite noch einmal abgebildet.

3.1.3 Die Summary-Seite

Die Summary Webseite zeigt noch einmal alle wichtigen Informationen an, die auf der Sharing-Seite im Kapitel 3.1.2 für einen externen Benutzer durchgeführt wurden. Dazu zählen unter anderem:

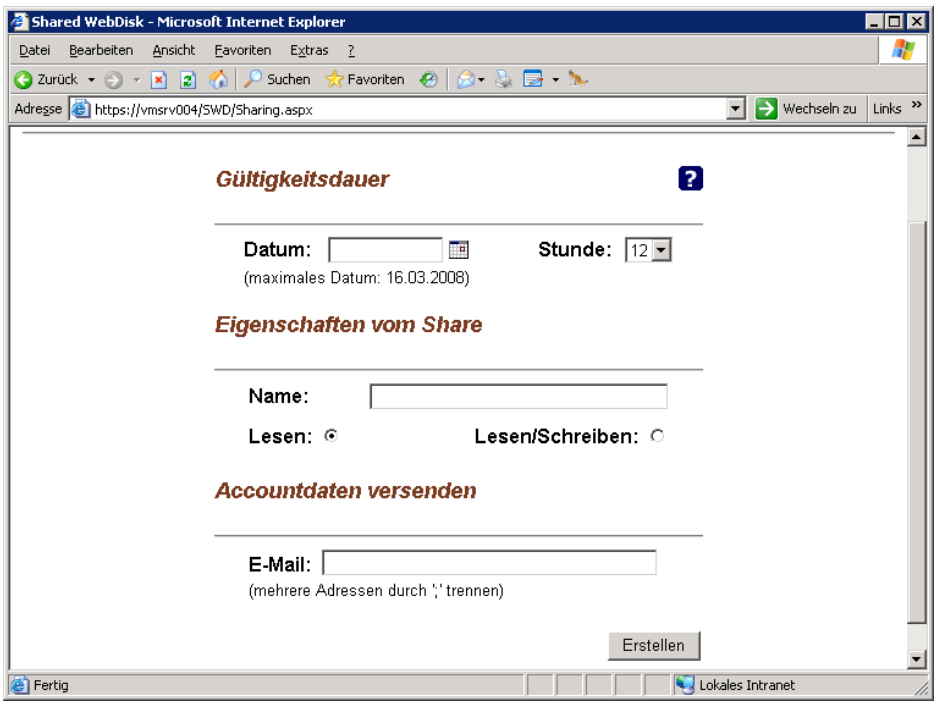


Abbildung 3.6: Ergebnis der Sharing-Seite

- Gültigkeit des externen Benutzeraccounts
- Zugriffsmöglichkeit auf das temporäre Verzeichnis (lesen/schreiben)
- der eingerichtete temporäre Benutzername mit Passwort
- der komplette Verzeichnispfad als UNC
- die URL, unter dem man den Ordner im Webbrowser erreichen kann

Dabei zeigte sich jedoch, dass die Weiterleitung der benötigten Daten von der Sharing-Seite zur Summary-Seite eine nicht ganz triviale Aufgabe bei Webanwendungen in ASP darstellt.

Problem: Parameterübergabe zwischen einzelnen Seiten

Bei der Parameterübergabe zwischen einzelnen Seiten werden mehrere Werte zwischen mindestens zwei Seiten untereinander ausgetauscht. Um diese Aufgabe zu lösen, existieren bereits einige Möglichkeiten, welche jedoch alle ihre Vor- und Nachteile haben[Sch06]. Leider sind jedoch einige der wenigen Möglichkeiten ungeeignet, wenn man eine relativ sichere Anwendung konstruieren will. So kann man zum Beispiel alle auszutauschenden Daten von einer Seite zur anderen in einem *Query-String* (also clientseitig) angeben, den allerdings ein eventueller Angreifer mithören oder auch manipulieren kann.

Im Bild 3.7 sind alle drei Möglichkeiten noch einmal aufgezeigt, die für eine Datenübergabe zwischen zwei Seiten verwendet werden können. In den ersten Zeilen sind dabei die drei *Seitenübergangsmöglichkeiten* dargestellt. Die Funktion *Response.Redirect()* ist die klassische

	Datenübergabe per Query-String	Datenübergabe per Sitzungszustand	Datenübergabe per Page.PreviousPage
Möglich bei Response.Redirect()	Ja	Ja	Nein
Möglich bei Server.Transfer()	Ja	Ja	Ja
Möglich bei Cross-Page-Postback	Nein	Nein	Ja
Übergabe von Zeichenketten	Ja	Ja	Ja
Übergabe von anderen Objekten	Nein	Ja, wenn serialisierbar	Ja, alle
Längenbeschränkung	Ja	Nein	Nein
Sicherheit	Problematisch	Kein Problem	Kein Problem
Belastung des Servers	Nein	Ja	Kaum

Abbildung 3.7: Vergleich der Datenübergabearten[Sch06]

Variante zur Weiterleitung auf einer neuen Webseite. Jede dieser Seitenübergangsmöglichkeiten hat dabei ihre Vor- und Nachteile, welche noch einmal im Dokument [Sch06] nachgelesen werden können.
In den nächsten Zeilen werden die drei Datenübergabearten näher untersucht. So sieht man zum Beispiel, dass die *Query-String*-Methode bei der Sicherheit und der Längenbeschränkung große Nachteile gegenüber den anderen beiden Methoden hat, weshalb sie aus diesen Gründen bei einer entsprechenden Lösungsfindung nicht weiter beachtet wurde. Eine geringere Gewichtung bei der Entscheidung der drei Methoden war die Serverbelastung, da die Anzahl der Zugriffe auf die SWD nicht so enorm sind wie beispielsweise bei einer Suchmaschine. Außerdem zeigt die Tabelle die Möglichkeiten zur Übergabe von Informationen wie Zeichenketten und Objekte. Diese Information hatte ebenfalls nur einen geringen Einfluss auf die Entscheidungsfindung, da in der SWD nur Zeichenketten übergeben werden.

Um nun einen Datenübergabe zwischen den einzelnen Seiten zu erreichen, wurde dieses mit Hilfe von Sitzungsvariablen ermöglicht. Da bei dem Einsatz von Sitzungsvariablen die Weitergabe der Werte serverseitig abläuft, kann kein Angreifer die sensiblen Daten wie Passwörter, mithören bzw. manipulieren, was die Nachteile bei der *Query-String*-Methode sind. Außerdem bietet diese Variante die Möglichkeit den Seitenübergang mit der Funktion *Response.Redirect()* zu gestalten. Da bereits diese Funktion *Response.Redirect()* beim Login-Mechanismus verwendet wurde, kam letztendlich diese Methode zum Einsatz.

Sobald die Daten auf der Webseite dargestellt wurden, werden die einzelnen Sitzungsvariablen wieder gelöscht, um einen Missbrauch der Daten auszuschließen.

Ausgeben und Löschen von Sitzungsvariablen

```
01 username.Text = Session["username"].ToString();
02 Session["username"] = "";
03 password.Text = Session["password"].ToString();
04 Session["password"] = "";
```

Logout von SWD

Nach dem Anzeigen der Ausgabe war die Einrichtung eines neuen Shares abgeschlossen. Da aber für viele Benutzer der SWD das Ende der Webanwendung nicht ersichtlich war, wurde zusätzlich ein simpler *Logout-Button* integriert, der den Benutzer wieder zur Login-Maske führt. Auf diese Weise konnte der Benutzer nun sicher gehen, dass er sich korrekt von der Webanwendung abgemeldet hat.

Die folgende Abbildung 3.8 zeigt noch einmal die fertige Webseite.

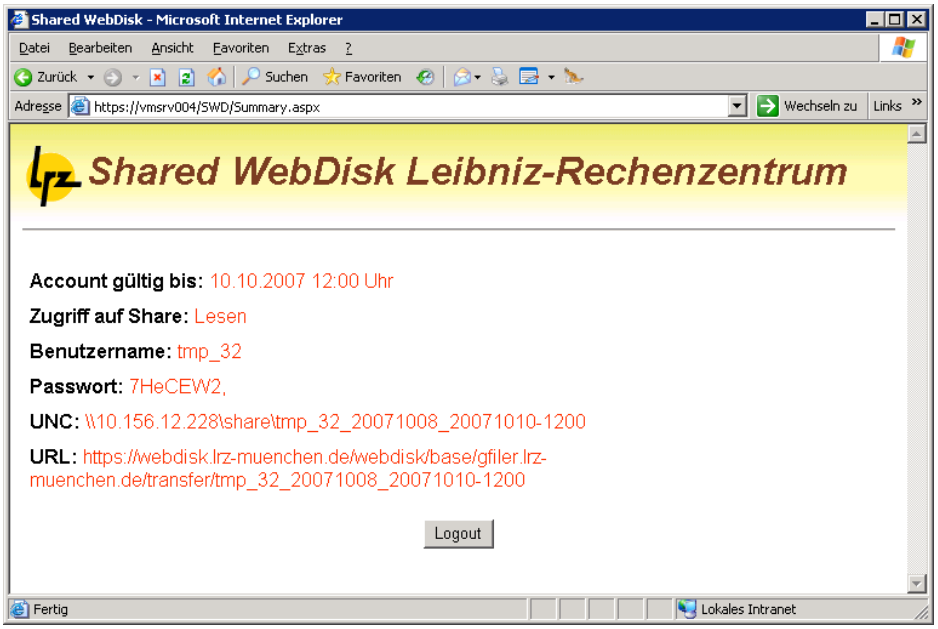


Abbildung 3.8: Ergebnis der Summary-Seite

3.2 Entwicklung des Scheduler

Wie bereits weiter oben erwähnt wurde, muss eine weitere Anwendung entwickelt werden, die die zuvor durch die Webanwendung erstellten temporären Benutzerkonten und Verzeichnisse für den verwendeten Datenaustausch wieder komplett aus der Umgebung und vom System entfernt. Diese Anwendung wird dann zu vorher definierten Zeiten automatisch ausgeführt.

Zur Erstellung eines solchen Scheduler wird die Funktionsweise in zwei Teile zerlegt. Im ersten Teil des Programms werden alle Zugriffsrechte von den temporären Verzeichnissen, bei denen der Benutzeraccount bereits abgelaufen ist, entfernt bzw. dahingehend geändert, so dass die Nutzer nicht mehr auf die Verzeichnisse zugreifen können. Dadurch wird ein weiteres Austauschen der Daten verhindert. Eine solche Deaktivierung der Zugriffsrechte für die Verzeichnisse ist deswegen sinnvoll, da es unter Umständen vorkommen kann, dass einige interne Benutzer noch einmal kurzfristig auf die zuvor ausgetauschten Daten zugreifen wollen. Aus diesem Grund werden die Daten nicht sofort gelöscht. Die Säuberung des Fileservers hingegen erfolgt im zweiten Teil der Anwendung, die alle zuvor gesperrten Verzeichnisse aus

dem System löscht. Diese Löschung findet jedoch einige Tage später als die Deaktivierung der Zugriffsrechte statt. Die genauere Entwicklung soll später beschrieben werden.

Des Weiteren soll der Scheduler durch Konfigurationsparameter leicht an verschiedene Umgebungen angepasst werden können. Solche Anpassungen können zum Beispiel der Logging-Mechanismus oder die Dauer in Tagen sein, nachdem ein bereits gesperrtes Benutzerkonto vom System automatisch wieder entfernt werden soll. Aus diesem Grund müssen sämtliche wichtige Einstellungen, die für die Verwendung des Scheduler benötigt werden, als globale Konfiguration im Programm verfügbar sein.

Damit nun der Scheduler die richtige Entscheidung zur Deaktivierung bzw. Löschung für die einzelnen Verzeichnisse treffen kann, müssen einige wichtige Angaben von jedem Verzeichnis in irgendeiner Weise hinterlegt werden. Dabei sind folgende Informationen wichtig:

- der interne Ersteller des Verzeichnisses
- der externe Benutzer, der die Berechtigung für den Zugriff auf dieses Verzeichnis hat
- das Ablaufdatum, an dem der externe Benutzer den Zugriff auf das interne Netzwerk verliert

Diese ganzen Informationen wurden am Anfang in den jeweiligen Ordnernamen hinein codiert. Diese Variante wurde bis zur ersten größeren Benutzung der Shared WebDisk beibehalten. Allerdings war dabei der Verzeichnisname eher kryptisch zu lesen, weshalb schon bei einer geringen Anzahl von diesen temporären Verzeichnissen es schwierig war zu sehen, welcher Ordner dem jeweiligen internen und externen Benutzer angehört. Aus diesem Grund wurde eine neue Variante entwickelt, um die Benutzerfreundlichkeit in dieser Hinsicht zu erhöhen.

Abbildung 3.9 zeigt noch einmal, wie ein solcher *kryptischer* Verzeichnisname aussieht.

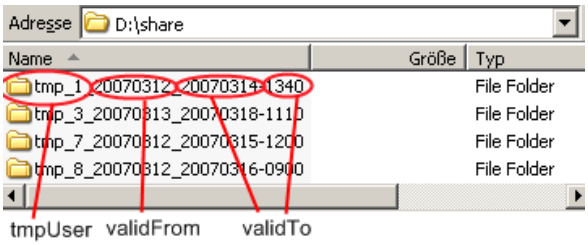


Abbildung 3.9: Beispiel einiger kryptischer Verzeichnisnamen

In der neuen Variante, kann nun der interne Benutzer einen Ordnernamen individuell selbst festlegen, was die Benutzerfreundlichkeit enorm steigert. Da jedoch der Scheduler trotzdem einige Informationen über diesen Ordner benötigt, existiert in jedem dieser Ordner eine Datei *directory.dat*, in der benutzerspezifische Daten gespeichert sind, wie dieses bereits im Abschnitt 3.1.2 erläutert wurde.
Um ein Manipulieren der directory.dat zu vermeiden, können die Benutzer nur lesend auf diese Datei zugreifen. Mit diesen Informationen kann dann der Scheduler die richtigen Entscheidungen treffen, die zum Säubern des Systems erforderlich sind.

3.2.1 Zugriff auf das Active Directory

Um nun jedoch unter der Skriptsprache VBScript einen Zugriff auf das Active Directory zu bekommen, musste zunächst ein Zusatzprogramm von Microsoft installiert werden, was diese Funktionalitäten entsprechend bereitstellt.
Dafür stellt Microsoft das Tool *Active Directory Service Interface (ADSI)* zur Verfügung, wovon allerdings nur ein kleiner Teil, nämlich die Datei *AdSecurity.dll*, für den Scheduler benötigt wurde, da in dieser Datei bereits viele Funktionen für den Zugriff und der Manipulation von Daten implementiert sind. (Siehe Anhang 6 für eine genaue Installation von ADSI).

3.2.2 Einbinden des Verzeichnispfades

Ein großes Problem bei der Ausführung des Scheduler war jedoch die Einbindung des Verzeichnispfades, in dem die einzelnen temporären Ordner abgelegt sind. Da sich die temporären Ordner auf einem anderen Server (dem Fileserver) befinden, mussten diese erst in das System, auf dem der Scheduler installiert ist, per UNC-Pfad (Uniform/ Universal Naming Convention) eingebunden werden. Dabei stellt ein UNC-Pfad eine Möglichkeit dar, mit der Ressourcen, wie Ordner und Laufwerke auf einem fremden System eingebunden werden können. Jedoch bietet die Skriptsprache *VBScript* keine Mechanismen für eine solche Einbindung an, sodass vor der Ausführung des Scheduler erst automatisch das Verzeichnis vom Fileserver mit hauseigenen Kommandozeilen-Befehlen eingebunden werden muss. Nach Beendigung des Ablaufs vom Scheduler muss dieser dann soeben erzeugte UNC-Link vom Fileserver wieder auf dieselbe Weise entfernt werden.

Dieses soeben beschriebene Problem, führte letztendlich dazu, dass ein weiteres, aber simples Batch-Script geschrieben werden musste, das den Fileserver in das lokale System einbindet, dann den Scheduler selber ausführt und im letzten Schritt den UNC-Pfad wieder entfernt.

Batch-Script "runScheduler.cmd" zum Mounten und Unmounten des externen Filesystems

```
net use z: \\fileserver.fopra.test\transfer
cscript C:\fopra\Schedule\Sharedwebdisk\scheduler.vbs
net use z: /delete
```

Der komplette Quellcode der Datei *scheduler.vbs* ist noch einmal im Anhang 5.3 abgedruckt.

3.2.3 Einrichten des Cronjobs

Im letzten Schritt bei der Erstellung des Scheduler muss nun noch ein sogenannter Cronjob eingerichtet werden, der die Aufgabe hat, dass der Scheduler täglich zu einer bestimmten Uhrzeit automatisch ausgeführt wird. Diese Aufgabe kann allerdings mit Windows eigenen Tools gelöst werden. Dazu muss nur in der *Systemsteuerung* von Windows unter *Geplante Tasks* ein entsprechender Eintrag vorgenommen werden, der das Programm zu einer definierten Uhrzeit automatisch startet.

3.2.4 Ausführen des Scheduler

Beim Starten des Scheduler muss, wie bereits oben erwähnt, nur noch die Datei *runScheduler.cmd* aufgerufen werden, der dann automatisch den Scheduler korrekt ausführt. Nach dem erfolgreichen Durchlaufen des Programms, sind alle wichtigen Änderungen, welche der Scheduler am Filesystem und im Active Directory vorgenommen hat, im Windows EventLog nachzuvollziehen.

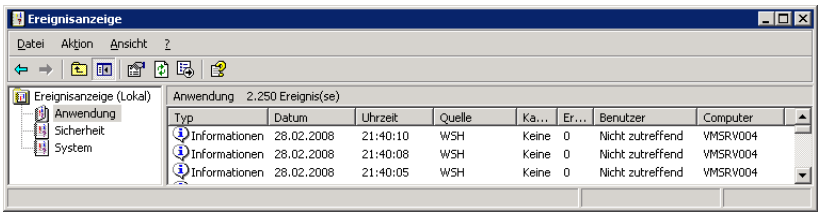


Abbildung 3.10: Ausgabe des Windows EventLogs

4 Ausblick auf Erweiterungsmöglichkeiten

Bei der Entwicklung der Webanwendung sind einige neue Gedanken zur Verbesserung der Bedienbarkeit aufgekommen, welche aber noch nicht in die Anwendung implementiert wurden. Außerdem sollen noch einige Vorschläge für die SWD aufgezeigt werden, die für eine eventuelle Weiterentwicklung interessant wären.

4.1 Logon-Seite

Beim Login-Mechanismus kommen einige interessante Erweiterungsmöglichkeiten im Betracht. Dadurch kann der Logon-Vorgang für den Benutzer komfortabler gestaltet werden. Im folgenden werden diese Ideen kurz näher erläutert.

Zugriff für andere Domänen

Eine weitere nützliche Verbesserung wäre den Login-Mechanismus dahingehend zu erweitern, dass zukünftig mehrere verschiedene Domänen für den Login-Vorgang unterstützt werden. Dadurch ist der Benutzer nicht mehr gezwungen, eine Kennung im LRZ zu besitzen. Es genügt, wenn der Benutzer in einer fremden Domäne, die für die Nutzung dieses Dienstes die notwendigen Berechtigungen besitzt, ein Benutzerkonto hat, um ebenfalls die WebDisk sowie die SWD verwenden zu können. Das wäre jedoch nur dann sinnvoll, wenn die WebDisk und SWD für andere Domänen verwendet werden soll.

4.2 Sharing-Seite

Auf der Sharing-Seite können ebenfalls noch einige Möglichkeiten zur besseren Benutzung der SWD implementiert werden, welche die Benutzerfreundlichkeit stark erhöhen würden.

E-Mail Feld automatisch vorbelegen

Da beim Anlegen eines neuen Shares auch die Zugangsdaten an E-Mail-Adressen gesendet werden können, bietet es sich an, das Eingabefeld zum Versenden an den Mail-Adressen auf der Sharing-Seite automatisch vorzubelegen. Hierbei könnte aus dem Active Directory das Attribut *mail* herangezogen werden, das bei jedem Benutzerkonto existiert. Leider ist jedoch dieses Attribut nicht bei jeder Benutzerkennung gefüllt, weswegen man dieses bei der Implementierung berücksichtigen müsste.

Anzeige des nur möglichen Zeitraums im Kalender

Ebenfalls sinnvoll wäre das Abändern des Kalenders auf der Sharing-Seite so, dass nur der mögliche Zeitrahmen im Kalender angezeigt wird. Dadurch kann der Anwender der SWD nicht mehr aus Versehen einen Tag auswählen, der nicht im vordefinierten Zeitrahmen liegt.

Eingabe einer temporären Benutzerkennung

Zur Zeit wird die neue temporäre Benutzerkennung automatisch von der SWD vorgegeben. Eine andere Möglichkeit wäre die eigene Eingabe einer temporären Benutzerkennung, ähnlich zu den selbst definierbaren Verzeichnisnamen. Dadurch können beide Seiten (interne und externe Benutzer) einen eigenen sinnvollen Namen für ihren Datenaustausch definieren, was ebenfalls zur Erhöhung der Benutzerfreundlichkeit beitragen würde.

4.3 Sonstige Verbesserungen

Weiterhin gibt es noch eine Reihe von Verbesserungen, die eine eventuelle Weiterentwicklung der SWD vereinfachen könnte.

Auslagern des Seitenlayouts vom LRZ

Um den Quellcode der SWD besser zu warten, bietet es sich an das Layout, welches vom LRZ vorgegeben ist, in eine externe Datei auszulagern. Dadurch kann eine Veränderung des Webseitenlayouts schnell in die SWD integriert werden, ohne dass dabei die gesamte SWD überarbeitet werden müsste.

Aufnahme des Projekts in Sourceforge

Da die IntegraTUM WebDisk bereits als OpenSource in Sourceforge[1] verfügbar ist, bietet es sich an, die SWD ebenfalls als OpenSource bereitzustellen. Dadurch kann nicht nur eine Weiterentwicklung der SWD garantiert werden, sondern es können auch andere Personen, die bereits die IntegraTUM WebDisk einsetzen, die SWD kostenlos benutzen.

[1] http://webdisk.sourceforge.net/

5 Quellcode

5.1 Active Directory Attribute von Benutzern

Das folgende Listing zeigt eine Ausgabe des Programms *ldp.exe*, mit dem man ein LDAP-Verzeichnisdienst abfragen kann.

In den Zeilen 01 bis 05 wird eine Verbindung zur Domäne *fopra.local* aufgebaut. Die Zeile 08 stellt dann eine Suchanfrage an diese Domäne mit dem Active Directory Attribut *samaccountname* und dem Wert *test*, was bedeutet, das er alle möglichen Parameter mit ihren dazugehörigen Werten für den Benutzernamen *test* ausgibt (Zeile 12 bis 46).

```
01 ld = ldap_open("fopra.local", 389);
02 Established connection to fopra.local.
03 res = ldap_bind_s(ld, NULL, &NtAuthIdentity, 1158); // v.3
04 {NtAuthIdentity: User='root'; Pwd= <unavailable>; domain = 'fopra'.}
05 Authenticated as dn:'root'.
06 ----------
07 ***Searching...
08 ldap_search_s(ld, "CN=Users,DC=fopra,DC=local", 2, "
       (samaccountname=test)", attrList, 0, &msg)
09 Result <0>: (null)
10 Matched DNs:
11 Getting 1 entries:
12 >> Dn: CN=test user,CN=Users,DC=fopra,DC=local
13 4> objectClass: top; person; organizationalPerson; user;
14 1> cn: test user;
15 1> sn: user;
16 1> telephoneNumber: 111111;
17 1> givenName: test;
18 1> initials: TU;
19 1> distinguishedName: CN=test user,CN=Users,DC=fopra,DC=local;
20 1> instanceType: 0x4 = ( IT_WRITE );
21 1> whenCreated: 03/01/2007 13:53:18 Mitteleuropäische Zeit
22 1> whenChanged: 11/13/2007 08:00:07 Mitteleuropäische Zeit
23 1> displayName: test user;
24 1> uSNCreated: 14556;
25 2> memberOf: CN=mygroup,CN=Users,DC=fopra,DC=local;
       CN=proxy_group,CN=Users,DC=fopra,DC=local;
26 1> uSNChanged: 335363;
27 1> name: test user;
28 1> objectGUID: 2b0622c8-69af-46d6-98d6-c69311baf9a5;
```

```
29 1> userAccountControl: 0x10200 =
        ( UF_NORMAL_ACCOUNT | UF_DONT_EXPIRE_PASSWD );
30 1> badPwdCount: 0;
31 1> codePage: 0;
32 1> countryCode: 0;
33 1> badPasswordTime: 03/20/2007 09:37:34 Mitteleuropäische Zeit
34 1> lastLogoff: 01/01/1601 01:00:00 Mitteleuropäische Zeit
35 1> lastLogon: 11/15/2007 13:34:58 Mitteleuropäische Zeit
36 1> pwdLastSet: 03/01/2007 13:53:18 Mitteleuropäische Zeit
37 1> primaryGroupID: 513;
38 1> objectSid: S-1-5-21-2722755026-1280140194-3078850485-1122;
39 1> accountExpires: 09/14/30828 02:48:05 UNC ;
40 1> logonCount: 659;
41 1> sAMAccountName: test;
42 1> sAMAccountType: 805306368;
43 1> userPrincipalName: testA@fopra.local;
44 1> objectCategory: CN=Person,CN=Schema,CN=Configuration,
        DC=fopra,DC=local;
45 1> lastLogonTimestamp: 11/13/2007 08:00:07 Mitteleuropäische Zeit
46 1> mail: nor@sss.de;
47 ------------
```

5.2 Konfigurationsdatei der SWD

Das folgende Listing zeigt die Datei *Web.Config*, welche alle Parameter für die Verwendung der SWD beinhaltet.eine

```
<?xml version="1.0" encoding="utf-8"?>
<!-- VERSION: 0.90 -->
<configuration>
<appSettings>
<!-- Define the path for the LDAP server -->
<add key="ADPath" value="LDAP://DC=fopra,DC=local"/>
<!-- Define where the temporary users are stored -->
<add key="LdapTmpUserContainer"
        value="LDAP://CN=Users,DC=fopra,DC=local"/>
<!-- every internal user can create a maximum
        ammount of user accounts -->
<add key="MaxSharesForUser" value="40"/>
<!-- The default domain from your environment -->
<add key="DefaultDomain" value="fopra.local"/>
<!-- Every internal user that is in the authorized group -->
<!-- is allowed to use this application -->
<add key="AuthorizedGroup" value="proxy_group" />
<!-- The prefix for temporary user names -->
<add key="TmpUserPrefix" value="tmp_"/>
<!-- The path to the storage server -->
```

```
<add key="UNCSharePath" value="\\10.156.12.228\share"/>
<add key="URLSharePath" value="
https://webdisk.lrz-muenchen.de/webdisk/base
        /gfiler.lrz-muenchen.de/transfer"/>
<!-- A temporary user account is valid for its
        setted time span -->
<add key="MaxDifferenceDays" value="10"/>

<!-- Logging -->
<!-- LogOnLogFile: boolean (true, false) -->
<add key="LogOnLogFile" value="true"/>
<add key="LogFile" value="c:\log.txt"/>
<!-- LogOnEnventLog: boolean (true, false) -->
<add key="LogOnEnventLog" value="true"/>

<!-- SMTP configuration -->
<add key="SMTPServer" value="mailout.lrz-muenchen.de"/>
<add key="SMTPPort" value="25"/>
</appSettings>
<system.web>
<authentication mode="Forms">
<forms loginUrl="logon.aspx" name="adAuth" timeout="10" path="/">
</forms>
</authentication>
<authorization>
<deny users="?"/>
<allow users="*"/>
</authorization>
<identity impersonate="true"
userName="registry:HKLM\SOFTWARE\ShareWebDisk
        \identity\ASPNET_SETREG,userName"
password="registry:HKLM\Software\ShareWebDisk
        \identity\ASPNET_SETREG,password"/>
</system.web>
</configuration>
```

5.3 Datei scheduler.vbs

```
' VERSION: 0.90

'-------------------------------------------------------
'-------------------------------------------------------
' Configuration (global options)
'
LdapTmpUserContainer="LDAP://CN=Users,DC=fopra,DC=local"
SharePath="z:\"
```

```
ADDomain="FOPRA"
DelayInDays=2

LogFile="c:\log_scheduler.txt"
LogToFile=1                 '1=true, 0=false
LogToEventLog=1 '1=true, 0=false

' don't change the following parameters
ConfigFile="directory.dat"

'=============================================================
'-------------------------------------------------------------
'  Main BEGIN
'
if(LogToFile=1) then
        Call CreateLogFile(LogFile)
end if

Call RemoveOldDirectory(DelayInDays, SharePath, ConfigFile)
Call DeleteDirectoryAccess(SharePath, ConfigFile)

' Main END

'=============================================================
'=============================================================
'=============================================================
'-------------------------------------------------------------
' Delete the specified Ace from the DACL
'
sub DeleteAce (oSd, trustee )
  set oDacl = oSd.DiscretionaryACL
  for each ace in oDacl
        'wscript.echo("ace="&ace.Trustee& and trustee="&trustee)
     if( ace.Trustee = trustee ) then
                'wscript.echo("treffer")
         oDacl.RemoveAce ace
     end if
  next
  oSd.DiscretionaryACL = oDacl
end Sub

'=============================================================
'-------------------------------------------------------------
' Get value from specified parameter
'
function GetDirectroyProperty(path,line)
        Const ForReading = 1
```

```
        Set objFSO = CreateObject
        ("Scripting.FileSystemObject")
        Set objTextFile = objFSO.OpenTextFile
        (path, ForReading)

        Do Until objTextFile.AtEndOfStream
                properties=objTextFile.Readline
                splitString=Split(properties,"=")
                if (StrComp(splitString(0),line)=0) then
                        GetDirectroyProperty=splitString(1)
                        exit do
                end if
        Loop
end function

'=============================================================
'-------------------------------------------------------------
'  Create a log file if this not exists
'
sub CreateLogFile(LogFile)
        ' Create file if this not exists
        Set objFSO = CreateObject
        ("Scripting.FileSystemObject")
        If not objFSO.FileExists(LogFile) Then
                Set objFile = objFSO.CreateTextFile
                (LogFile)
        end if
end sub

'=============================================================
'-------------------------------------------------------------
'  Write message into a specified logfile
'
sub WriteToLogFile(message)
        Set objFSO = CreateObject
        ("Scripting.FileSystemObject")
        Set objTextFile = objFSO.OpenTextFile
        (LogFile, 8, True)
        objTextFile.WriteLine(message)
        objTextFile.Close
        set objTextFile = Nothing
end sub

'=============================================================
'-------------------------------------------------------------
'  Write message into windows eventlog
'
```

```
sub WriteToEventLog(message)
        Const EVENT_SUCCESS = 0
        Const EVENT_ERROR = 1
        Const EVENT_WARNING = 3
        Const EVENT_INFO = 4
        Set WshShell = WScript.CreateObject
        ("WScript.Shell")
        WshShell.LogEvent EVENT_SUCCESS, message
        set WshShell = Nothing
end sub

'=============================================================
'-------------------------------------------------------------
'  Delete the ACE from tmpUser and creator from working directory
'  and the tmpUser from Active Directory
'
sub DeleteDirectoryAccess(SharePath, ConfigFile)
        set objOU = GetObject(LdapTmpUserContainer)
        Set objFSO = CreateObject
        ("Scripting.FileSystemObject")
        Set objFolder = objFSO.GetFolder(""&SharePath)
        Set colSubfolders = objFolder.Subfolders

        For each objSubfolder in colSubfolders
          Set oAdsSecurity = CreateObject("ADsSecurity")
          path=SharePath & "\" & objSubfolder.Name & "\" &
          ConfigFile
          If objFSO.FileExists(path) Then
            Username=GetDirectroyProperty(path,"TMP_USER")
                Creator=GetDirectroyProperty(path,"CREATOR")
                ExpiredDate=GetDirectroyProperty
                (path,"EXPIRED_DATE")
                ExpiredTime=GetDirectroyProperty
                (path,"EXPIRED_TIME")

                ExpiredDateAndTime=ExpiredDate & " " & ExpiredTime
                ExpiredDateAndTime=CDate(ExpiredDateAndTime)

                If (Now() >= ExpiredDateAndTime) then
                  'delete ACE from tmpUser
                  sDirPath = "FILE://" & SharePath & "\" &
                  objSubfolder.Name
                  'WScript.Echo("Directory: "&sDirPath)
                  set oFileSD = oADsSecurity.GetSecurityDescriptor
                        (CStr(sDirPath))
                  set oDacl = oFileSD.DiscretionaryACL
                  'WScript.Echo("tmpUser: "&Username)
```

```
                    Call DeleteAce(oFileSD,ADDomain&"\"&Username)
                    oAdsSecurity.SetSecurityDescriptor oFileSD,
                          CStr(sDirPath)

                    'delete ACE from creator of directory
                    Call DeleteAce(oFileSD,ADDomain&"\"&Creator)
                    oAdsSecurity.SetSecurityDescriptor oFileSD,
                          CStr(sDirPath)

                    'write log
                    message=Now() & "; User deleted: " & Username
                    if(LogToFile=1) then
                      WriteToLogFile(message)
                    end if
                    if(LogToEventLog=1) then
                      WriteToEventLog(message)
                    end if

                    'clean environment
                    set oDacl=nothing
                    set oFileSD= nothing
                  end if
          end if
        set oAdsSecurity=nothing
        next
end sub

'=============================================================
'-------------------------------------------------------------
' Remove the old directory after some days
'
sub RemoveOldDirectory(DelayInDays, SharePath, ConfigFile)
        set objOU = GetObject(LdapTmpUserContainer)
        Set objFSO = CreateObject("Scripting.FileSystemObject")
        Set objFolder = objFSO.GetFolder(SharePath)
        Set colSubfolders = objFolder.Subfolders

        For Each objSubfolder in colSubfolders
          path=SharePath & "\" & objSubfolder.Name & "\"
          & ConfigFile
          If objFSO.FileExists(path) Then
                Username=GetDirectroyProperty
                        (path,"TMP_USER")
                ExpiredDate=GetDirectroyProperty
                        (path,"EXPIRED_DATE")
                ExpiredDate=CDate(ExpiredDate)
```

```
                wscript.echo("checking folder: "&path)
                today=Date()
                if(today>(ExpiredDate+DelayInDays)) then
                  message=Now() & "; Share deleted: " &
                        SharePath & "\" & objSubfolder.Name
                  if(LogToFile=1) then
                    WriteToLogFile(message)
                  end if
                  if(LogToEventLog=1) then
                    WriteToEventLog(message)
                  end if

                  'delete share
                  wscript.echo "Share deleted: " &
                        SharePath&"\"&objSubfolder.Name
                  objSubfolder.Delete

                  'delete tmpUser from Active Directory
                  'WScript.Echo("User deleted from AD: " & Username)
                  objOU.Delete "user", "cn="& Username
                end if
          End If
        next
        set objOU = Nothing
end sub
```

6 Installationsanleitung

Die Webanwendung bzw. der Scheduler wurden auf einem deutschen Windows 2003 Server SP1 und IIS 6 entwickelt. Es sollte aber grundsätzlich auch auf anderen Windows Plattformen laufen.

6.1 Benötigte Programme

- ASP.NET Framework >=2.0 [Download]
- aspnet_setreg.exe [Download und Info]
- Active Directory Service Interfaces - ADSI [Download und Info]

6.2 Einrichten des Storage Server

Für den externen Zugriff auf die Daten muss vorher auf dem Storage Server ein Ordner erstellt werden, in dem die temporären Verzeichnisse für Benutzer erzeugt werden.

```
# mkdir c:\share
```

Jetzt diesen Ordner im Netzwerk noch freigeben:

```
Rechte Maustaste auf C:\share => Freigabe und Sicherheit =>
Auswählen: Diesen Ordner freigeben => Berechtigungen: Löschen von 'Jeder'
und Hinzufügen von: 'proxy_user' mit 'Vollzugriff' => OK
```

Außerdem müssen noch die Rechte für diesen Share richtig gesetzt werden: Dazu öffnen Sie die Zugriffsberechtigungen vom gerade freigegebenen Verzeichnis.

```
rechte Maustaste auf C:\share => Eigenschaften => Sicherheit
```

Nun müssen wir die Berechtigung von dem Eintrag *Benutzer* entsprechend anpassen, damit dieser nicht für die erzeugten temporären Unterordner automatisch vererbt wird.

```
Erweitert => Haken entfernen bei 'Berechtigungen übergeordneter
Objekte, sofern vererbbar...' => Kopieren
```

Nun noch im Tabreiter *Berechtigung* für jeden der beiden Einträge vom *Benutzer* die Vererbung ausschalten

```
Ersten Eintrag vom Benutzer auswählen =>
Bearbeiten => bei 'Übernehmen für:'
'Nur diesen Ordner' auswählen => OK
```

Das gleiche machen wir nun noch für den zweiten Eintrag und schließen danach alle Fenster. Danach können wir den Share auf dem IIS Server dauerhaft einbinden.

```
Arbeitsplatz => Extras => Netzlaufwerk verbinden =>
bei Ordner: Pfad angeben (\\STORAGESERVER\share)
=> 'anderem Benutzernamen' =>
Benutzername: proxy_user, Kennwort: password = OK
=> Fertig stellen
```

6.3 Installation und Konfiguration der Webanwendung

6.3.1 Anlegen eines virtuellen Verzeichnisses im II6

Öffnen Sie den Internetinformationsdienste-Manager und legen Sie ein neues virtuelles Verzeichnis unter *Computername–>Websites–>Standardwebsite* an.

- Alias: ShareWebDisk
- Pfad: Pfad zu dem Webordner (z.B. C:\Inetpub\wwwroot\ShareWebDisk)
- Berechtigungen: Lesen, Skripts ausführen

Nun entpacken Sie die Datei *ShareWebDisk_0.XX.zip* in dem oben erstellten Ordner (C:\Inetpub\wwwroot
\ShareWebDisk).

6.3.2 Konfiguration der Webanwendung

Im ersten Schritt muss das ASP.NET Framework 2.0 heruntergeladen und installiert werden. Nun muss die Weboberfläche an der eigenen Umgebung angepasst werden. Dazu nehmen Sie folgende Änderungen in der Datei *web.config* vor:

ADPath: Gibt den Pfad zum LDAP Server wieder

LdapUserContainer: Bestimmt den Container, in dem die internen Benutzer gehalten werden

LdapTmpUserContainer: Bestimmt den Container, in dem die temporären Benutzer gehalten werden

MaxSharesForUser: Jeder interne Benutzer darf maximal eine festgelegte Anzahl von temporären Shares einrichten

DefaultDomain: Die Domain, in der sich die internen Benutzer befinden

AuthorizedGroup: Jeder interne Benutzer, der in der angegebenen Gruppe ist, darf ein Share anlegen

TmpUserPrefix: Der Präfix, der für jede angelegte Benutzerkennung verwendet wird

SharePath: Der Pfad zum Storage Server, auf dem die Shares eingerichtet werden

MaxDifferenceDays: Ein temporärer Benutzer darf für höchstens eine bestimmte Anzahl von Tagen eingerichtet werden

Jetzt muss die Gruppe, welche oben bei der Konfiguration des Parameters *AuthorizedGroup* eingestellt wurde, noch im Active Directory erzeugt werden. Danach sollten ihr noch einige Benutzer hinzugefügt werden, die für die Erstellung eines temporären Verzeichnisses berechtigt sind.

Als nächstes muss noch ein Benutzer erstellt werden, mit dem auf dem Storage Server die Verzeichnisse erstellt werden sollen. Dazu muss im Active Directory ein neues Benutzerkonto (z.B. *proxy_user*) mit einem gültigen Passwort angelegt werden.

Damit das Passwort des eben erzeugten Benutzers nicht im Klartext in der Datei *web.config* enthalten ist, wird es verschlüsselt in der Registry von Windows abgespeichert. Dazu wird das Programm *aspnet_setreg.exe* benötigt.

Dieses Programm muss nun mit folgender Syntax in einer Kommandobox aufgerufen werden.

```
aspnet_setreg.exe -k:SOFTWARE\ShareWebDisk\identity
                  -u:"fopra.local\proxy_user" -p:"Passwort"
```

Die beiden Registry Keys, die bei dem obigen Befehl ausgegeben werden, müssen nun in der Datei *Web.config* eingetragen werden.

Zur Sicherheit sollte jedoch noch der Schlüssel in der Registry mit entsprechenden Zugriffsrechten versehen werden. Dazu startet man das Programm *regedt32.exe* und sucht den Schlüssel

HKEY_LOCAL_MACHINE\SOFTWARE\ShareWebDisk

Es reicht aus, wenn der Benutzer *Netzwerkdienst* mit Leserechte hinzugefügt wird.

Achtung: Dieses muss auch für die Unterschlüssel vorgenommen werden.

Die Webanwendung ist nun unter folgender URL erreichbar:

https://SERVERNAME/ShareWebDisk

6.4 Installation des Scheduler

Damit die alten Verzeichnisse der temporären Benutzer wieder entfernt werden und das Active Directory wieder gesäubert wird, muss der Scheduler noch installiert werden. Dazu wird das Tool *ADSI* von Microsoft benötigt.

6.4.1 Installation von ADSI

Das Tool kann unter den oben genannten Link heruntergeladen werden. Danach einfach in ein Verzeichnis (z.B. C:\) entpacken. Da wir nur ein Teil davon brauchen, reicht es aus wenn wir folgenden Befehl in der Kommandobox ausführen:

```
regsvr32 C:\adsi\ResourceKit\ADsSecurity.dll
```

6.4.2 Konfiguration des Scheduler

Um den Scheduler an die eigene Umgebung anzupassen, müssen im Script *scheduler.vbs* noch einige Parameter geändert werden.

LdapTmpUserContainer: Bestimmt den Container, in dem die temporären Benutzer gehalten werden

ADDomain: Domain, in der die internen und externen Benutzer sich befinden

SharePath: Das oben eingebundene Laufwerk, wo sich die temporären Verzeichnisse befinden

TmpUserPrefix: Der Präfix, der für jede angelegte Benutzerkennung verwendet wird

DelayInDays: Eine Zahl, nach der das temporäre Verzeichnis gelöscht wird nachdem der temporäre Benutzer bereits vom System entfernt wurde

Damit nun das Script regelmäßig ausgeführt wird, empfiehlt es sich dafür ein Cronjob einzurichten.

Abbildungsverzeichnis

Literaturverzeichnis

[Acta] ActiveXperts: *Monitoring Event Logs*. Website. http://www.activexperts.com/activmonitor/windowsmanagement/adminscripts/monitoring/eventlogs/.

[Actb] ActiveXperts: *Scripts Collection*. Website. http://www.activexperts.com/activmonitor/windowsmanagement/adminscripts/.

[Hol] Holm, Christian: *Web.Config 101*. Website. http://www.aspheute.com/artikel/20010802.htm.

[JK02] Jörg Krause, Uwe Bünning: *ASP.NET-Programmierung mit C#*. Addison-Wesley; ISBN 978-3827319746, 2002.

[Mica] Microsoft: *Active Directory Domain Services Authentication from ASP .NET*. Website. http://msdn2.microsoft.com/en-us/library/ms180890.aspx.

[Micb] Microsoft: *Delete Users From Active Directory*. Website. http://cwashington.netreach.net/depo/view.asp?Index=167.

[Mic07] Microsoft: *How to use ADsSecurity.dll to add an access control entry to an NTFS folder*. Website, 2007. http://support.microsoft.com/?scid=kb%3Ben-us%3B279682&x=12&y=13.

[MSD] MSDN: *Identitätswechsel in ASP.NET*. Website. http://msdn2.microsoft.com/de-de/library/xh507fc5(VS.80).aspx.

[MSD07a] MSDN: *Adding users to Active Directory with .NET*. Website, 2007. http://channel9.msdn.com/Showpost.aspx?postid=130700.

[MSD07b] MSDN: *Searching Active Directory with .NET*. Website, 2007. http://channel9.msdn.com/Showpost.aspx?postid=132740.

[Sch06] Schwichtenberg, Dr. Holger: *Seitenübergänge und Datenübergabe in ASP.NET 2.0*, 2006.

[SF02] Stefan Falz, Karsten Samaschke: *Das ASP Codebook*. Addison-Wesley; ISBN 978-3827319487, 2002.

[SIL06] SIL, SANJIT: *Working with File and Directory Properties, Attributes and Access Control List*. Website, 2006. http://aspalliance.com/1047_Working_with_File_and_Directory_Properties_Attributes_and_Access_Control_List.

[Smi] Smith, Eric: *C# Tip: Creating a Password-Checking Function*. Website. http://www.developer.com/net/csharp/article.php/3634946.

[Ulm] Ulm, Tobi: *Das aspnet_setreg Tool*. Website. `http://www.devtrain.de/artikel_915.aspx`.

[W3S] W3Schools: *VBScript Functions*. Website. `http://www.w3schools.com/vbscript/vbscript_ref_functions.asp`.

[Yah] Yahoo: *YUI Library Examples: Browser History Manager (beta): Calendar Control*. Website. `http://developer.yahoo.com/yui/examples/history/history-calendar.html`.