LogFile-Auswertungstool

für

CMS RedDot-LogFiles

Projektdokumentation

Thorsten Blei

xima-media GmbH
Sudhausweg 9
01099 Dresden

Projektzeitraum: 01. April bis 04. Mai 2007

Inhaltsverzeichnis

1 Problemstellung

1.1 Übersicht

Seit 1998 hat sich die xima media GmbH auf die Konzeption, Entwicklung und Implementierung von Softwarelösungen insbesondere im E-Governmentsektor konzentriert. Das Unternehmen ist seit seiner Gründung stetig gewachsen und hat zurzeit 26 feste Mitarbeiter.
Als Premium Partner der RedDot Solutions AG berät und betreut die xima media GmbH unter anderem auch die Sächsische Staatsregierung bei der Umsetzung des „Zentralen CMS“ (= zentrales Content Management System, kurz ZCMS) als Basiskomponente der E-Government-Plattform und realisiert und betreut entsprechende RedDot-Projekte.
Aufgrund seiner Komplexität treten sowohl bei der RedDot-Server-Betreuung als auch bei Kundenprojekten Fehler auf, bei deren Analyse eine Auswertung der Logdateien sinnvoll ist. Diese werden in einem nahezu validen (gültigen) XML-Format vom Server gespeichert. Diese Dateien sind aber sowohl für versierte Nutzer als auch für erfahrene Administratoren nur mit großem Suchaufwand zu analysieren und eine Filterung der gewünschten Daten stellt einen erheblichen Zeitaufwand dar.

1.2 Problemstellung

Daher stellt ein LogFile-Auswertungstool für RedDot-Logdateien eine erhebliche Arbeitserleichterung, speziell für Administratoren, dar. Aufgrund der eingeschränkten Entwicklungszeit ist es nicht möglich, auf alle Server- oder Nutzeraktionen einzugehen. Deshalb geht es darum, eine Software zu entwickeln, welche eine übersichtliche Darstellung dieser Dateien ermöglicht.

1.3 Projektumfeld für die Entwicklung

Als Plattform für das Auswertungstool dient ein RedDot-Server. Folgende Voraussetzungen sind darauf bereits geschaffen worden:

- Windows 2000: Server oder Advanced Server oder
- Windows 2003: Standard Edition oder Enterprise Edition
- MS Internet Explorer ab Version 5.5 SP2
- Anmeldung mit lokalen Administrator-Rechten auf dem Server
- Feste IP-Adresse für den Server

- Installierter Internet Information Server
- Installiertes MS XML Version 4.0 mit SP1
- Installiertes MDAC Version 2.8
- Installierte VB Runtime Version 6 mit SP5

1.4 Ausgangssituation

Zu Beginn des Projekts existieren die Logdateien als Fragment einer validen (gültigen) XML-Datei. Diese werden immer vom RedDot-Server im RedDot-Installationsverzeichnis unter „\CMS\ASP\LOG\Common“ abgelegt.

2 Planung

2.1 Projektplan

Projektphase	Aufgabe	Zeit in h	Gesamtzeit in h
Planung/Analyse	Kundenwünsche erfassen Log-Verzeichnis/ -Dateien analysieren	4 12	16
Umsetzung	Dateistruktur planen Programm umsetzen	15 19	34
Testen	Prüfung der Auswertung	5	5
Installation	Testen des Projekts	2	2
Dokumentation	Projektdokumentation erstellen Kundendokumentation erstellen	9 4	13

3 Projektverlauf

3.1 Projektstart

Im Gespräch mit der xima media GmbH wurden die Anforderungen an die Anwendung aufgenommen. Spezielle Anforderungen wurden noch nicht festgelegt, da es hier um die Entwicklung einer Vorlage geht und sich der Umfang des Projekts noch nicht genau abschätzen lässt.

3.2 Analyse

Bei der Einsicht in die Logdateien wurde noch einmal die Komplexität derselben deutlich. Die Inhalte sind in einem XML-ähnlichen Format aufgebaut, wobei keine Deklaration und kein Root-Element erkennbar waren, die bei XML-Dateien erforderlich sind. Beim Ergänzen dieser Daten, um eine XML-Darstellung im Internet Explorer zu ermöglichen, wurde festgestellt, dass in einigen Elementen doppelte Attributwerte existieren, die keine XML-Konformität darstellen.
Im weiteren Analyseverlauf wurde festgestellt, dass fast alle Dateien einen Datumswert im Dateinamen haben und die Dateiendung die Bezeichnung „.log“ hat. Diese Dateien haben eine Größe von etwa 1,0 MByte während die aktuellste den Namen „RDCMS.log“ hat und eine veränderbare Größe besitzt. Es wird also davon ausgegangen, dass diese die aktuelle Logdatei ist, während die anderen, mit Datumswert im Namen, in einem nicht mehr veränderlichen Archivzustand sind.
Um einen Überblick in die Dokumentation des Systems zu erhalten, wurde ein neues RedDot-Projekt erstellt, welches lediglich zwei Standardfelder (Textfelder ohne weitere Formatierungen) und zwei Textfelder (Textfelder, in welchen Formatierungsoptionen vom RedDot-Nutzer vorgegeben werden können) enthält. Um möglichst wenige Logvorgänge zu erhalten, wurde auch ein neuer Nutzer (tbl_abschluss) angelegt, der Autorenrechte (Nutzer mit Elementbearbeitungsrechten) besitzt. Danach wurde das Verzeichnis „common“ geleert, um die Protokollierungsvorgänge von Beginn an nachvollziehen zu können.

3.2.1 Testdatenerstellung

Mit dem Testnutzer (tbl_abschluss) wurden im RedDot die vorher angelegten Elemente gefüllt und die Bearbeitung derselben wurde abgeschlossen (RedDot-Vorgehensweise). Danach wurde der Nutzer vom System wieder abgemeldet und um ein Weiterschreiben der Logdatei zu vermeiden, wurde diese zur Auswertung in ein separates Verzeichnis kopiert.

3.2.2 Testdatenauswertung

Die Testdatei besitzt eine Größe von etwa 40 kByte, was eine umfangreiche Protokollierung der Vorgänge des RedDot-Servers erahnen lässt. Dieser Eindruck wird bei der Einsichtnahme in den Inhalt der Datei deutlich.
Es werden XML-Elemente aufgelistet, welche im regelmäßigen Wechsel die Namen „CLIENT“ und „SERVER“ tragen. In diesen sind Kindelemente enthalten, welche an RQL-Anfragen bzw. -Antworten aus der RedDot-RQL-Dokumentation erinnern.
Unter Zuhilfenahme dieser Dokumentation lassen sich diese Elemente zuordnen und eindeutig identifizieren. Sie stellen alle Vorgänge auf dem System dar und sind chronologisch aufgelistet. Jeder Anfrage (CLIENT-Element) folgt direkt eine Antwort (SERVER-Element). Es ist der komplette Anmeldevorgang, das Laden des Projektes, die Elementbearbeitung und der Abmeldevorgang protokolliert. Diese Vorgänge sind so komplex dargestellt, dass es unerlässlich ist, eine Filterung der Daten vorzunehmen, um einzelne Aktionen zu betrachten, da viele Hintergrundanfragen und –antworten enthalten sind, welche selbst für den Administrator nur in seltenen Fällen interessant sind.
Aufgrund dieser Komplexität wurde beschlossen, sich bei der Erstellung des Projektes ausschließlich auf die Elementbearbeitung zu konzentrieren.
Da in dieser Datei keine doppelten Attribute vorkommen, wird diese Datei zum analysieren und auswerten beibehalten um zusätzlichen Programmieraufwand vorerst zu vermeiden.

3.3 Entwurfsphase

3.3.1 Entwurf zum Einlesen und Umwandeln in XML-Daten

Es wurde ein ASP-Script entwickelt, welches ein Dateiobjekt einer XML-Datei erstellt, um diese mit dem Internet Explorer in der XML-Ansicht darzustellen. Zu diesem Zweck wird der Microsoft XML Parser benötigt, welcher auch in jeder RedDot-Installation enthalten ist und verwendet wird. Nach der gewünschten Anzeige wurde nach Elementen gesucht, welche mit dem Attributwert der „guid“ am häufigsten vorkommen, um ein erstes Analysekriterium auszuwählen. Hierbei fiel auf, dass die meistverwendete Guid sich auf die Login-Guid des angemeldeten Nutzers bezog. Daher wurde versucht, mittels ASP ein weiteres XML-Objekt zu erstellen, in welchem ausschließlich alle Elemente mit dem Attribut „guid“ und dessen Wert der Login-Guid aufgelistet sind. Um die einheitliche XML-Struktur zu wahren wurden auch die dazugehörigen Eltern- und gegebenenfalls Großelternelemente in das neue Objekt geschrieben. Die Auswahl der Elemente erfolgte mit Hilfe von Xpath. Nach der Ausgabe in

eine neue XML-Datei wurde festgestellt, dass dies eine wesentliche Verkleinerung des Datenaufkommens und Verbesserung der Übersichtlichkeit bedeutete.

3.3.2 Entwurf zum Anzeigen gewünschter XML-Daten

Zur benutzerdefinierten Auswahl und Darstellung wurde die XSL Technologie ausgewählt. Mit ihrer Zuhilfenahme wurde ein erster XSL-Stylesheet entwickelt, welcher RedDot-Textelemente, deren Bearbeitung abgeschlossen wurde, anzeigt. Um diese Elemente aufzufinden und auszuwählen wurde die RQL-Dokumentation zu Hilfe genommen. Nach erfolgreicher Anzeige konnte nun über den Programmablauf und Programmaufbau nachgedacht werden.

3.3.3 geplanter Programmablauf

Es soll ein XML-Objekt erstellt werden, aus welchem die angemeldeten Nutzer und deren Session-ID ausgelesen und dargestellt werden. Nach der Auswahl sollen die vom Anwender durchgeführten Aktionen in einer HTML-Seite dargestellt werden.

3.4 Umsetzungsphase

Die Umsetzung des Projekts wurde nach dem Spiralmodell vorgenommen, da sich die Komplexität der Logfilestruktur nicht im Vorfeld erfassen ließ. So ergeben sich aus den einzelnen XML-Elementen immer neue Zusammenhänge, welche bei der Auswertung berücksichtigt werden können. In solchen Fällen ist dann über deren Notwendigkeit zu entscheiden und über eine Integration in das Projekt nachzudenken.

3.4.1 Umsetzung der Auswahlseite 1 (auswahl.asp)

Bei der Programmierung der Oberfläche zum Einlesen der Logdateien wurde aufgrund des begrenzten Zeitrahmens darauf verzichtet, mehrere Logdateien für einen bestimmten Zeitraum auswählbar zu machen, um zum Beispiel die Vorgänge eines Tages, welcher in mehreren Dateien vorhanden sein kann, für die Auswahl anzuzeigen. Hierfür ist später eine Funktion zu entwickeln, welche die vorhandenen Dateien im Verzeichnis „common“ einliest, die Dateinamen auswertet und als Datumsauswahl anbietet. Im vorliegenden Projekt ist ausschließlich auf das Auswerten der *RDCMS.log* eingegangen worden. Eine weitere Einschränkung wurde bei Erstellung der selektierten XML-Datei vorgenommen. Hier wird nur auf die Aktionen eingegangen, welche sich an den Session-Guid’s orientieren. Eine mögliche Erweiterung wird aber durch die Anzeige der Guid’s der bearbeiteten RedDot-Projekte

aufgezeigt.
Schwerwiegende Probleme sind bei der Erstellung dieser Datei nicht aufgetreten. Die Entwicklung erfolgte auf der Grundlage der unter Punkt 3.3.1 erlangten Erkenntnisse.

3.4.2 Umsetzung der Auswahlseite 2 (selectansicht.asp)

Bei der Umsetzung dieses Scripts ging es darum, die ausgewählten Daten aus 3.3.1 in den Programmablauf zu übernehmen, erneut das XML-Objekt aufzubauen und aus den übernommenen Daten ein neues selektiertes XML-Objekt zu erstellen. Aus diesem Objekt sollte nach einer Auswahlverfeinerung auf bestimmte Nutzeraktionen eingegangen werden, um diese in einem nutzerfreundlichen Format anzuzeigen. Dies wurde erreicht, indem ein Teil des Scripts der auswahl.asp übernommen wurde. Eine Auswahl der auszuwertenden RedDot-Elemente wurde erstellt und in einem HTML-Formular bereitgestellt.
Für die nutzerfreundliche Anzeige der ausgewählten Daten wurde nach einer Möglichkeit gesucht, die Daten nicht als XML-Datei mit einem XSL-Stylesheet anzuzeigen, sondern die Ausgabe in ein, von den meisten gebräuchlichen Browsern unterstützten, HTML-Format umzuwandeln, da zum Beispiel der Opera-Browser die XML-Umwandlung nicht unterstützt. Nach einem Hinweis von Herrn Gehre (xima media GmbH) wurde dies dann mittels Objekttransformierung umgesetzt. Diese Art der Transformierung weist vor allem den Vorteil auf, dass die Anzeigeform der Daten nicht über das Script erfolgt, sondern über den XSL-Stylesheet, welcher im Nachhinein noch angepasst werden kann.

3.4.3 Umsetzung der XSL-Stylesheets

Die Umsetzung der Stylesheet-Dateien wurde auf der Grundlage der unter 3.3.2 gewonnenen Erkenntnisse durchgeführt und erweitert. Ein geringes Problem bei der Anzeige der Ergebnisdaten war, dass ein Element bei der Auswertung mittels XPath durchaus mehrfach gefunden werden konnte und dessen Daten durch den erforderlichen Schleifenaufruf auch entsprechend oft angezeigt wurden. Nach einer intensiven Recherche wurde diese Unstimmigkeit allerdings schnell behoben, indem nur das erste aller gefundenen Elemente angezeigt wird.

3.5 Testphase

Eine ausgiebige Testphase konnte aufgrund des sonstigen Arbeitsumfanges in der xima media GmbH nicht stattfinden. Eine nicht im Firmenumfeld beschäftigte Person konnte hierfür auch nicht herangezogen werden, da es recht wenige RedDot-Administratoren gibt, welche mögliche Fehler oder Mängel erkennen könnten. Daher bezieht sich die Testphase

auf die Durchläufe, welche während oder nach der Erstellung vom Entwickler durchgeführt wurden. Somit ist eine Fehlerbehebung oder –analyse erst zu einem späteren Zeitpunkt möglich.

4 Zusammenfassung / Schlussfolgerung

Im Rahmen der Projektarbeit ist ein eingeschränkt nutzbares Modul entstanden, welches es den RedDot-Administratoren erleichtert, nutzerbezogene Vorgänge auf dem RedDot-System darzustellen und zu analysieren. Dieses Modul ist mit einfachen Mitteln erweiterbar, da zusätzliche XSL-Dateien entsprechend den gewünschten Anforderungen schnell in das Projekt einzubinden sind.

Für eine spätere Weiterentwicklung ist in Erwägung zu ziehen, ob in einem eigenen, projektbezogenen Log- oder Tempverzeichnis die XML-Objekte als Datei abgelegt werden sollten, um ein erneutes Berechnen zu vermeiden und damit Rechenlast einzusparen. Dafür sollte aber zusätzlich über einen Wartungsworkflow nachgedacht werden, um ein übermäßiges Datenaufkommen auf dem Server zu vermeiden.
Desweiteren ist bei der Erweiterung zu beachten, dass die in der Analysephase aufgetretene Problematik der doppelten Attribute in XML-Elementen des Logfiles noch nicht behoben wurde. Daher könnten immer wieder Fehler auftreten, da die Ursache dieser noch nicht bekannt ist.

Bei einer kurzen Präsentation in der xima media GmbH wurde die Erfüllung des Projektauftrages anerkannt. Eine wirtschaftliche Nutzung und Weiterentwicklung ist vorgesehen.
Das LogFile-Auswertungstool wird weiterhin betreut und erweitert. Als nächstes Ziel wurde die Ursachenforschung und Fehlerbehebung der o.g. doppelten Attribute gesetzt um im weiteren Entwicklungsverlauf das komplette Verzeichnis einlesen und auswerten zu können.

5 Quellenverzeichnis

Zur Entwicklung des Projekts wurden folgende Quellen genutzt:

Das ASP.NET Codebook
von Stefan Falz, Karsten Samaschke
Verlag: Addison-Wesley; Auflage: 1 (Oktober 2003)
ISBN-10: 3827320496

XML in der Praxis . Professionelles Web-Publishing mit der Extensible Markup Language
von H Behme, S Mintert
Verlag: Addison-Wesley; Auflage: 2
ISBN-10: 3827316367

XML, m. CD-ROM
von Heinz Wittenbrink, Werner Köhler
Verlag: Teia Lehrbuch Verlag; Auflage: 2
ISBN-10: 3935539692

MSDN Library
http://www.microsoft.com/germany/msdn/library

LogFile-Auswertungstool

für

CMS RedDot-LogFiles

Anhang

Thorsten Blei

xima-media GmbH
Sudhausweg 9
01099 Dresden

Projektzeitraum: 01. April bis 04. Mai 2007

Inhaltsverzeichnis

A1 Abkürzungsverzeichnis

ASP	Active Server Pages
CMS	Content Management System
Guid	Globally Unique Identifier
RQL	RedDot Query Language
XML	Extensible Markup Language
XPath	XML Path Language
XSL	Extensible Stylesheet Language
ZCMS	Zentrales Content Management System (siehe http://www.egovernment.sachsen.de/53.htm)

A2 Kundendokumentation

A2.1 Installation

Die Installation des Projektes setzt lediglich eine vorhandene RedDot-Installation voraus.

Kopieren Sie den kompletten Projektordner in das RedDot-Log-Verzeichnis ihres Servers in den Ordner hinein (Beispiel: Z:\Programme\RedDot\CMS\ASP\LOG). Durch die von RedDot festgelegte Ordnerstruktur erübrigen sich weitere zusätzlichen Einstellungen.

A2.2 Programmaufruf

Wenn Sie das Programm direkt auf dem RedDot-Server durchführen wollen, können Sie im Internetbrowser die lokale Adresse des Programms angeben.
Diese lautet bei einer RedDot-Standardinstallation dann:

```
localhost/cms/log/projektordner/auswahl.asp
```

Wurde das Programm auf einem anderen Server installiert, den Sie überdas Internet erreichen können, so rufen sie es entsprechend der Serveradresse auf.

Beispiel: `www.meinserver.de/cms/log/projektordner/auswahl.asp`

Falls Sie keine Verbindung zum Programm erhalten sollten, setzen Sie sich mit dem betreuenden Administrator in Verbindung um eventuelle Fehler zu beheben oder Adressänderungen zu erfahren.

A2.3 Programmablauf

Nach Aufruf des Programms erhalten Sie eine Übersicht der angemeldeten Benutzer in Verbindung mit ihrer Login-Guid in Tabellenform angezeigt. Es kann durchaus vorkommen, dass ein Benutzername mehrfach mit unterschiedlichen Login-Guid's vorkommt, da einem Nutzer bei jeder Anmeldung eine neue Login-Guid zugewiesen wird. Die Auswertung der Nutzerdaten bezieht sich auch auf diese Guid. Eine Auswertung ausschließlich nach Nutzernamen ist noch nicht implementiert und ist auch in nächster Zeit noch nicht geplant. Als Aussicht auf näher liegendere Erweiterungen ist unterhalb dieser Auswahl schon eine Tabellenansicht der bearbeiteten Projekte vorbereitet. Später kann hier die Ansicht

projektorientiert ausgewählt werden.
Wählen Sie also nun den Nutzer aus, dessen Aktionen Sie sich näher anschauen wollen und drücken Sie auf „senden“.

Sie erhalten nun eine verfeinerte Auswahl für möglicherweise bearbeitete Elemente. Wählen Sie hier eines aus und drücken Sie abermals auf „senden“.
Jetzt bekommen Sie alle Änderungen und relevanten Daten für die ausgewählte Elementart angezeigt.
Um sich für ein weiteres Element die Ergebnisse anzuschauen, navigieren Sie mit der Browsernavigation zurück und starten Ihre neue Auswahl.
Zum Beenden des Programms schließen Sie entweder das Browserfenster oder geben Sie eine neue URL ein.

A3 Verfügbare Elementattribute im Log-File

Der Umfang und die Übersichtlichkeit der nachstehend aufgeführten XML-Elemente soll gleichzeitig noch einmal veranschaulichen, dass die Projektentwicklung eine Verbesserung der Wirtschaftlichkeit im Rahmen des CMS-Supports darstellt, indem die relevanten Daten in einem leicht überschaubaren Format angezeigt werden.

A3.1 Das Element Standardfeld im LogFile

Folgende Attribute sind im Element ELT mit dem type-Attribut = 1 enthalten. Einige davon wurden in der XSL-Datei *element_std.xsl* bereits verwendet und sind „fett“ hervorgehoben:

```
<ELT action="load" subelements="1" actionflag="" extendedinfo="" headlinepageguid=""
notsamepage="" reddotcacheguid="BA27BBDCA2304C3DA2630C4844602928"
sessionkey="F608C0EDF27D4839A8EDA3EA55A77260" dialoglanguageid="DEU" languagevariantid="DEU"
ok="1" eltlanguageindependent="0" guid="85B0D1F10437461EADE755BD43EC1639"
templateelementguid="83C59417860143EC9C1C1A3770DD7860"
pageguid="3D78EF57ACB7475D94204024BF13A5D2" eltflags="2097152" eltrequired="0" eltdragdrop="0"
islink="0" formularorderid="0" orderid="1" status="0" name="std_std1" eltname="std_std1"
aliasname="std_std1" variable="std_std1" maxsize="255" type="1" elttype="1"
templateelementflags="2097152" templateelementislink="0" value="std_abc" reddotdescription=""
flags="18874368" manuallysorted="-1" useconnection="1"
userguid="3FB2796F0FCD42AFA0B3E5C0BAB999E4" changeuserguid="3FB2796F0FCD42AFA0B3E5C0BAB999E4"
glrights1="-1" glrights2="-1" glrights3="-1" glrights5="-1" glrights6="-1" gldenys1="0"
gldenys2="0" gldenys3="0" gldenys5="0" gldenys6="0" rights2="-1" rights3="-1" rights4="-1025"
rights5="-1" rights6="-1" userreleasestatus="4" rights1="-33554433" pagestatus="4"/>
```

A3.2 Die RedDot-Seite im LogFile

Aus diesem Element werden im aktuellen Projekt die seitenspezifischen Informationen ausgelesen. Auch hier sind die verwendeten Attribute „fett“ hervorgehoben:

```
<PAGE action="load" loginguid="BAD920C7B2F640ABB6AB28E3C5B11B77"
editlinkguid="00000000000000000000000000000001" sessionkey="F608C0EDF27D4839A8EDA3EA55A77260"
dialoglanguageid="DEU" languagevariantid="DEU" guid="3D78EF57ACB7475D94204024BF13A5D2" id="1"
templateguid="DE516E7EAB4149B7821A1828E180BAE0" templaterights="2147483647"
mainlinkguid="00000000000000000000000000000001" parentguid="00000000000000000000000000000001"
headline="Startseite" name="" sitemapname="" templatepath="Content-Klassen"
templatetitle="Basisseite" templateflags="536870912" headlinedescription="" status="1"
flags="525312" breadcrumbstartpoint="0" breadcrumbdonotuse="0" releaseguid=""
createdate="39182,5387847222" createuserguid="4CCC399102C94CFAB35413E877BF0779"
createusername="tbl" changedate="39195,4713541667"
changeuserguid="4CCC399102C94CFAB35413E877BF0779" changeusername="tbl"
releasedate="39195,4734606482" releaseusername="tbl"
releaseuserguid="4CCC399102C94CFAB35413E877BF0779" lockdate="39195,4713657407"
lastchangesincelocked="39195,4713657407" lockusername="" checkindate="39195,4734606482"
drafttemplates="0" templateswaitforrelease="0" templatelockuserguid="" templatelockdate=""
```

```
templatelockusername="" templatelock="0" useconnection="1"
userguid="3FB2796F0FCD42AFA0B3E5C0BAB999E4" btflags="0"/>
```

A3.3 Element zur Erfassung einer Speicheranfrage

Aus diesem Element werden Datum und Uhrzeit ausgelesen, gleichzeitig werden u.a. Client-Elemente bei der nutzerspezifizierten XML-Objekterstellung genutzt.

```
<CLIENT guid="F87A876A693E4B7B81A2ED21FA2A4AC6" date="2007-04-23T09:22:59.443+02:00">
<IODATA loginguid="BAD920C7B2F640ABB6AB28E3C5B11B77"
sessionkey="F608C0EDF27D4839A8EDA3EA55A77260">
<ELEMENTS translationmode="0" action="save" reddotcacheguid="BA27BBDCA2304C3DA2630C4844602928"
>
<ELT guid="85B0D1F10437461EADE755BD43EC1639" extendedinfo="" type="1" value="std_abc_tbl" >
</ELT></ELEMENTS></IODATA>
</CLIENT>
```

A4 Quelltexte

A4.1 auswahl.asp

```
<%

Response.ContentType = "text/html"
Dim objFileSys
Dim objTextStream
Dim strFileContent
Dim xmldoc
Dim success

'eine Instanz des FileSystemObjekts erstellen
Set objFileSys = Server.CreateObject("Scripting.FileSystemObject")

' der Instanz den Textstream des Logfiles hinzufügen
Set objTextStream = objFileSys.OpenTextFile(Server.MapPath("../Common/RDCMS.log"))

'alles aus dem Textstream einlesen
strFileContent = objTextStream.ReadAll

' Inhalt des virtuellen XML-Files aufbauen/zusammensetzen
strXMLContent = "<?xml version=""1.0"" encoding=""iso-8859-1""?>" & vbCrLf & _
                                "<REDDOT_LOG>" & vbCrLf & strFileContent & vbCrLf &
                                "</REDDOT_LOG>"

' schließen des TextStream-Objektes
objTextStream.close

' Objektvariablen trennen
Set objTextStream = nothing
Set objFileSys = nothing

' Instanz des XML - Objektes erstellen
SET xmldoc = Server.CreateObject("MSXML2.DOMDocument.4.0")

' XML - Instanz mit Inhalt füllen
success = xmldoc.loadXML(strXMLContent)


If success Then
%>

<html>
        <head>
                <title>Vorauswahl</title>
                <style type="text/css">
                        body {
                                background-color: lightgrey;
                                font-family: arial;
                                font-size: 1em;
                                }
                        h3 {
                                font-size: 1em;
                                font-weight: bolder;
                                }
                        table {
                                font-size: 0.8em;
                              }
                </style>
        </head>
        <body>
<%
        Dim nodeList
        Dim objDictUser
        Dim nodeItem
        Dim nextParentNode
```

```
        ' Aktionen herausfiltern
        Set nodeList = xmldoc.SelectNodes("//IODATA/ADMINISTRATION[@action = 'login']")
        Set objDict = Server.CreateObject("Scripting.Dictionary")


        %>
                <h3>Benutzer im gespeicherten Zeitraum:</h3>
                <form method="GET" action="selectansicht.asp">
                <table border="1">
                        <tr><th>Nutzer</th><th>Login-Guid</th></tr>
        <%
         For i = 0 To nodeList.Length - 1
         Set nodeItem = nodeList.Item(i)
         Set nextParentNode = NodeItem.ParentNode.ParentNode.NextSibling.firstChild.firstChild

         ' LogInGuid und zugehörigen User ermitteln und in Dictionary schreiben
         objDict.add nextParentNode.getAttribute("guid"), nodeItem.getAttribute("name")

         Response.Write "<tr><td> " & nodeItem.getAttribute("name") & _
         " </td><td> " & nextParentNode.getAttribute("guid") & " </td></tr>"

         Next
%>
                </table>
                <br/>
                Nutzer auswählen:
                <select name="user" size="1">
<%
           For Each key in objDict
           Response.Write "<option value='" & key & "'>" & objDict(key) & "</option>" & VbCrLf

           Next

           Set objDict = nothing

%>
                </select>
                <input type="submit" name="senden" value="auswahl"/>
        </form>
                <h3>Aufgerufene Projekte im gespeicherten Zeitraum:</h3>
                <table border="1">
                        <tr><th>Projektname</th><th>ProjektGUID</th></tr>
<%
          Set nodeList = xmldoc.SelectNodes("//IODATA/ADMINISTRATION/PROJECT[@guid != '']")
          Set objDict = Server.CreateObject("Scripting.Dictionary")

          For i=0 To nodeList.Length - 1
          Set nodeItem = nodeList.Item(i)
          Set nextParentNode =
nodeItem.parentNode.parentNode.parentNode.NextSibling.firstChild.FirstChild

          If Not objDict.Exists(nextParentNode.getAttribute("guid")) Then

             objDict.add nodeItem.getAttribute("guid"), nextParentNode.getAttribute("name")
             Response.Write "<tr><td> " & nextParentNode.getAttribute("name") & _
              " </td><td> " & nodeItem.getAttribute("guid") & " </td></tr>"

          End IF
          Next
          Set objDict = nothing
%>
       </table>
    </body>
</html>

<%
Else
        Response.Write "Es sind Fehler aufgetreten!"
End If
Set nodeItem = Nothing
Set xmldoc = Nothing
%>
```

A4.1.1 Quelltextbeschreibung (auswahl.asp)

Der Aufruf dieser Datei erfolgt über den Server. Dies kann sowohl von außen erfolgen, als auch lokal mit *„localhost/…/auswahl.asp“*.

Ablauf:

- Erstellen eines FileSystem-Objektes: `objFileSys`
- dem Objekt den Inhalt des Logfiles hinzufügen: `objTextStream`
- Alles aus der Instanz auslesen und an eine Stringvariable übergeben: `strFileContent`
- Um den Inhalt des Strings die XML-Deklaration (`<?xml version="1.0" encoding="iso-8859-1"?>`) und das Root-Element (`<REDDOT_LOG> … </REDDOT_LOG>`) schreiben um ein valides XML-Dokument zu erhalten: `strXMLContent`
- Schließen und Trennen der nicht mehr benötigten Objekte
- XML-Objekt (`xmldoc`) erstellen und mit dem Inhalt des XML-Strings (`strXMLContent`) füllen
- Prüfen, ob das XML-Objekt gefüllt wurde: … `If success Then`
  Falls nicht: Anzeige, dass es einen Fehler gegeben hat und Ende des Programms
  Ansonsten:
- Aufbau der ersten HTML-Seite für Nutzer- und Projektdarstellung und Auswahlformular
- Alle Nodes des XML-Objektes mittels XPath nach den Elementen durchsuchen welche einer Login-Aktion entsprechen und diese in einem Dictionary-Objekt `objDictUser` (entspricht einem 2-dimensionalen Array) ablegen.
- Nach dem Füllen des Dictionarys werden die darin befindlichen Nodes ausgelesen und nach den Attributwerten „name“ und „guid“ (entspricht in diesem Fall der Session-GUID) in eine Inhaltstabelle gefüllt
- Anschließend wird mit Hilfe desselben Dictionarys das Formular aufgebaut, mit dem die Session-GUID an das Script *selectansicht.asp* übergeben wird.
- Um das Dictionaryobjekt weiter verwenden zu können, wird es getrennt und wieder neu instanziiert.
- Jetzt wird mittels XPath nach Nodes gesucht, welche einem Projekt-Ladevorgang entsprechen. Diese werden wieder im Objekt `objDictUser` abgelegt, um sie speziell für CMS-Projektaufrufe auszulesen. Diese werden wiederum nach den

Attributen „name“ und „guid“ (ensprechend der Projekt-GUID) ausgelesen und in die Projekttabelle geschrieben.

- Auch hier kann im Erweiterungsfall ein Formular eingebunden werden, welches die Projekt-GUID übergibt.
- Jetzt wird das Dictionary-Objekt wieder getrennt und die HTML-Seite wird zu Ende aufgebaut.
- Falls ein Fehler beim Füllen des XML-Objektes aufgetreten ist, wird an dieser Stelle der If-Schleife der Else-Zweig aufgerufen, der die Fehlermeldung ausgibt.
- Nach dem Schließen der If-Schleife findet nur noch ein Trennen der Objekt-Elemente statt.

A4.2 selectansicht.asp

```
<%
        ' Uebergabe der ausgewaehlten Auswahlanfrage
         ********************************************************************************
        Dim objRequest
        Dim strField

        If Request.ServerVariables("REQUEST_METHOD") = "GET" Then
                Set objRequest = Request.QueryString
        Else
                Response.Write "Keine Daten zum Auswerten empfangen!"
        End If

        strUser = "user"
        strOption = "senden"
        strXSL = "xslAuswahl"
        userGUID = objRequest(strUser)
        anzeige = objRequest(strOption)
        xslSelect = objRequest(strXSL)

        Set objRequest = Nothing

        ' Aufbau des XML - Dokuments
        ********************************************************************************


        Dim objFileSys
        Dim objTextStream
        Dim strFileContent
        Dim xmldoc
        Dim success
        Dim nodeList
        Dim parentNode
        Dim nextNode
        Dim completeNode
        Dim objDictNode
        Dim strNodes

        Set objDictNode = Server.CreateObject("Scripting.Dictionary")
        Set objFileSys = Server.CreateObject("Scripting.FileSystemObject")
        Set objTextStream = objFileSys.OpenTextFile(Server.MapPath("../Common/RDCMS.log"))
        strFileContent = objTextStream.ReadAll
        strXMLContent = "<?xml version=""1.0"" encoding=""iso-8859-1""?>" & vbCrLf & _
                        "<REDDOT_LOG>" & vbCrLf & strFileContent & vbCrLf & "</REDDOT_LOG>"

        objTextStream.close
        Set objTextStream = nothing
```

```
        Set xmldoc = Server.CreateObject("MSXML2.DOMDocument.4.0")
        success = xmldoc.loadXML(strXMLContent)


        If success Then

        Set nodeList = xmldoc.SelectNodes("//child::*/IODATA[@loginguid = '" & userGUID &"']")

               For i = 0 To nodeList.Length - 1
                   Set nodeItem = nodeList.Item(i)
                   Set parentNode = nodeItem.parentNode
                   Set nextNode = nodeItem.parentNode.NextSibling
                   objDictNode.add i, parentNode.xml & nextNode.xml
               Next

               Set xmldoc = nothing
               strNodes = ""

               For each c in objDictNode
                       strNodes = strNodes & objDictNode(c)
               Next

               strXMLContent = "<?xml version=""1.0"" encoding=""iso-8859-15""?>" & vbCrLf & _
                               "<REDDOT_LOG>" & vbCrLf & strNodes & vbCrLf & "</REDDOT_LOG>"

               Set xmldoc = Server.CreateObject("MSXML2.DOMDocument.4.0")
               userXML = xmldoc.loadXML(strXMLContent)

       End If



        ' Feinauswahl abfragen
        ********************************************************************************

        If anzeige = "auswahl" Then
                Response.ContentType = "text/html"
                %>
                        <html>
                                <head>
                                        <title>Elementauswahl</title>
                                </head>
                                <body bgcolor="#DCDCDC">
                                        <h3>Elementauswahl der bearbeiteten Elemente</h3>
                                        <form action="selectansicht.asp" method="GET">
                                                Bitte wählen Sie eine Elementart aus, um sich
                                                darauf bezogene Nutzeraktionen und Details
                                                anzeigen zu lassen:
                                                <p><font face="courier">
<input type="radio" name="xslAuswahl" value="element_txt" checked="checked"> Textelement<br/>
<input type="radio" name="xslAuswahl" value="element_std"> Standardfeld<br/>
                                                </font></p>
<input type="hidden" name="user" value="<%=userGUID%>">
<input type="submit" name="senden" value="Anzeige"/>
                                        </form>
                                </body>
                        </html>
                <%
        Else

        'Ausgabe mit verfeinerten Auswahlkriterien
        ********************************************************************************

        Dim xslDoc
                Set xslDoc = Server.CreateObject("MSXML2.FreeThreadedDOMDocument.4.0")
        Dim xslTpl
                Set xslTpl = Server.CreateObject("MSXML2.XSLTemplate.4.0")
        Dim xmlDoc1
                Set xmlDoc1 = Server.CreateObject("MSXML2.DOMDocument.4.0")
```

```
        boolSuccessXML = xmlDoc1.LoadXML(strXMLContent)
        boolSuccessXSL = xslDoc.Load(Server.MapPath("xsl_elements/" & xslSelect & ".xsl"))




            If boolSuccessXML Then
                    If boolSuccessXSL Then
                      Set xslTpl.stylesheet = xslDoc
                      Set xslProc = xslTpl.createProcessor()
                      xslProc.input = xmlDoc1
                      xslProc.transform()
                      response.write unescape(xslProc.output)

                    Else
                      response.write xslDoc.parseError.reason
                    End If
            Else
                          response.write xmlDoc1.parseError.reason
            End If
        End If
   Set objFileSys = nothing
%>
```

A4.2.1 Quelltextbeschreibung (selectansicht.asp)

Der Aufruf dieses Scripts erfolgt aus dem Formular der Seite *auswahl.asp*.

- zuerst wird geprüft, ob Daten an das Script übergeben wurden und im Erfolgsfall werden diese an das Objekt `objRequest` übergeben.
- wurden keine Daten empfangen, wird der Nutzer mit einer Meldung darauf aufmerksam gemacht.
- jetzt wird wieder aus der Logdatei mit analoger Vorgehensweise ein valides XML-Objekt erstellt, welches weiterverarbeitet werden kann.
- Wenn beim Scriptaufruf der Wert `anzeige = „auswahl“` übergeben wurde, so wird nun eine HTML-Seite mit weiteren Auswahlkriterien (xslAuswahl) angeboten. Diese werden beim Absenden übergeben. Das Formular ruft wiederum das Script erneut auf und übergibt die neuen Werte.
- Beim Absenden aus dem eigenen Formular wurde der Wert für die Variable `anzeige` in „`Anzeige`“ geändert und somit wird jetzt der Scriptblock zum Aufbau der HTML-Seite übergangen. Es wird der „`Else`“-Zweig durchgeführt.
- Wenn in der Else-Schleife die Variable `anzeige` auf den Wert „`anzeige`“ geprüft wird, werden 3 neue Objekte zur Erstellung eines XML-Objektes, eines XSL-Objektes und eines DOM-Dokumentes erstellt.
- Das DOM-Dokument-Objekt wird angewiesen, den validen XML-String aus der Variable strXMLContent einzulesen

- Das XSL-Objekt wird angewiesen, die XSL-Datei einzulesen
- Falls beide Einlesevorgänge einen „`true`“-Wert zurückgeben beginnt das Zusammenführen und Transformieren der beiden Objekte HTML-Instanz
- Diese Instanz wird angezeigt, nachdem sämtliche ASCII-codierte Zeichen zurückgewandelt (unescaped) wurden. Anschließend werden alle Objekte getrennt.

A4.3 element_std.xsl (stellvertretend für alle XSL-Stylesheets)

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="iso-8859-1"/>
        <xsl:template match="/">
                <html>
                <head>
                  <title>Auswertungsansicht</title>
                </head>
                  <body bgcolor="#DCDCDC">
                     <h2>bearbeitete bzw. geaenderte Standardtextfeld-Elemente:</h2>
                     <font face="courier">
                         <xsl:for-each select="//CLIENT/IODATA/ELEMENTS">
                             <xsl:call-template name="element" />
                         </xsl:for-each>
                     </font>
                  </body>
                </html>
        </xsl:template>

        <xsl:template name="element">
                <xsl:choose>
                   <!-- Hier wird der Elementtyp festgelegt -->
                   <xsl:when test="@action = 'save' and child::ELT/@type = '1'">
                       <table border="1">
                       <xsl:variable name="elt_guid"><xsl:value-of select=
 "child::ELT/@guid"/></xsl:variable>
                       <tr><td width="200px">Element-GUID:</td><td width="600px"><xsl:value-
of select="$elt_guid"/></td></tr>
                       <tr><td>gesendeter Inhalt:</td><td><font color="green"><b><xsl:value-
of select="child::ELT/@value"/></b></font></td></tr>
                       <tr><td>Datum:</td><td><xsl:value-of select="substring(../../@date,
1, 10)"/></td></tr>
                       <tr><td>Uhrzeit:</td><td><xsl:value-of select="substring(../../@date,
12, 12)"/></td></tr>
                       <xsl:for-each select="../preceding::*/IODATA/ELT[@guid=$elt_guid and
@name!='']">
                         <xsl:call-template name="elementoptions" />
                       </xsl:for-each>
                       </table>
                       <p></p>
                   </xsl:when>

                   <xsl:otherwise>
                   </xsl:otherwise>

                </xsl:choose>
        </xsl:template>

        <xsl:template name="elementoptions">
           <xsl:variable name="page_guid"><xsl:value-of select="@pageguid"/></xsl:variable>
           <xsl:if test="position() = '1'">
             <tr><td>Elementname:</td><td><b><xsl:value-of select="@name"/></b></td></tr>
             <tr><td>vorheriger Inhalt:</td><td><font color="red"><xsl:value-of
select="@value"/></font></td></tr>
             <tr><td>Sprachvariante:</td><td><xsl:value-of
```

```
select="@languagevariantid"/></td></tr>
             <xsl:for-each select="../preceding::*/IODATA/PAGE[@guid=$page_guid and
@headline!='']">
                <xsl:if test="position() = '1'">
                   <tr><td>Seitenname:</td><td><xsl:value-of select="@headline"/></td></tr>
                   <tr><td>Seiden-ID:</td><td><xsl:value-of select="@id"/></td></tr>
                   <tr><td>Template-Bezeichnung:</td><td><xsl:value-of
select="@templatetitle"/></td></tr>
                </xsl:if>
             </xsl:for-each>
          </xsl:if>
        </xsl:template>
</xsl:stylesheet>
```

A4.3.1 Quelltextbeschreibung (element_std.xsl)

Die XSL-Dateien dienen der Darstellung von XML-Dateien und XML-Objekten.
In diesem Projekt haben die Dateien *element_std.xsl* und *element_txt.xsl* die Aufgabe, die nutzerspezifizierten XML-Objekte in eine für den Anwender lesbare HTML-Syntax umzuwandeln und darzustellen. Anhand der *element_std.xsl* soll die prinzipielle Arbeitsweise gezeigt werden:

- Die Datei beginnt mit der XML-Deklaration und wird gefolgt vom Dokumentelement, in dem auch der Namensraum für die XSLT-Elemente angegeben wird.
- das XSL-Element „`output`" gibt an, welches Format bei der Ausgabe erzeugt werden soll
- das erste Template-Element wird aufgerufen und liefert die Rahmenelemente für die HTML-Seite. Zudem wird für jedes Element „`ELEMENTS`" aus dem XML-Objekt, welches innerhalb der Elemente `CLIENT` und `IODATA` liegt, das Template `element` aufgerufen. Diese Bedingung ergibt sich aus dem Aufbau der Log-Datei.
- Das Template `element` prüft, ob das Attribut `action` gefüllt ist und das Attribut `type` des Kindelements den Wert `1` besitzt. Falls ja handelt es sich um ein RedDot-Element vom Typ „Standardfeld" und es können die dafür relevanten Daten ausgelesen werden.
- es wird die HTML-Tabelle aufgebaut und die einzelnen Tabellenzeilen mit Beschreibung und den dazugehörigen Daten in die Zeilen gefüllt.
- zur Identifizierung der elementspezifischen Daten wird eine Variable geschrieben, welche die GUID des ausgewählten Elements enthält. Danach wird die Variable in die erste Tabellenzeile eingetragen.
- Das Attribut `value` enthält die Daten, welche beim Abspeichern in RedDot dem Element übergeben wurden

- den nächsten beiden Zeilen wird jeweils eine Zeichenkette mit der Zeichenposition x und der Zeichenlänge y in der Form „`attributwert, x, y`“ vom Attribut `date` dargestellt, da in diesem sowohl Datum als auch Uhrzeit des Speichervorgangs dargestellt sind.
- Die nächste Schleife ruft alle vorangegangenen Elternelemente auf, welche die Elemente `IODATA` mit dem Element `ELT` enthalten, welches im Attribut den Inhalt der vorher gesetzten Variable aufweisen und deren Attributwert für das Attribut `name` nicht leer ist. Aus diesem lassen sich analog die noch fehlenden Attribute, die das Element Standardfeld beschreiben, auslesen. Zu diesem Zweck wird das Template `elementoptions` aufgerufen und ausgeführt.
- da diese Node im XML-Baum mehrfach auftreten kann, weil das Element mehrfach geladen wurde, wird die erste Node, welche zu den Auswahlkriterien gefunden wurde, selectiert und ausgewertet.
- Hier lässt sich auch im Attribut `name` die vergebene Bezeichnung des Elements ermitteln und die vorherigen Inhaltsdaten sind hier im Attribut `value` enthalten. Außerdem ist das Attribut für die Sprachvariante lesbar.
- Zum Schluss wird die Seiten-GUID aus dem Attribut `pageguid` in eine variable übergeben.
- Die dritte und letzte Schleife sucht wiederum nach dem Seitenelement, welchem das Standardfeld zugeordnet ist. Dieses ist in einem vorangegangenen Elternelement im Rootbaum als Kindelement des Elements `PAGE` zu finden und das Attribut `guid` hat den Inhalt der übergebenen Variable der Page-GUID.
- Hier werden zum Schluss die Attribute für die Seitenbezeichnung, der Seiten-ID und der Bezeichnung des Templates, welches die Seite erzeugt, ausgelesen.

Andere XSL-Stylesheets müssen ähnlich aufgebaut sein, um spätere elementspezifische Daten zu erfassen und darzustellen.