

icoyaDrive
WebDAV client für das icoya XML CMS

Joachim Bauch
Niels Mache

Struktur AG
Jungansstr. 5
D-70469 Stuttgart
Germany

www.struktur.de
www.strukturag.com

www.icoya.de
www.icoya.com
www.icoya.jp

3. April 2002

Abstract

icoyaDrive bietet eine anwendungsunabhängige Möglichkeit, auf das icoya CMS Repository zuzugreifen. Durch die Einbindung in den Windows Date Explorer können beliebig Dateien in das Repository kopiert oder von dort ausgecheckt werden. Weiterhin können einzelne Versionen und Sprachen eines Dokuments angesehen und bearbeitet werden.

Die Kommunikation mit dem icoya CMS läuft über eine HTTP bzw. HTTP/S Verbindung mit dem WebDAV Protokoll und icoya spezifischen Erweiterungen.

1. Inhaltsverzeichnis

1.	Inhaltsverzeichnis.....	2
2.	Übersicht.....	3
2.1.	Konventionen.....	3
3.	Implementierung.....	3
3.1.	Speicherverwaltung.....	4
3.2.	Klassendiagramm der implementierten Objekte.....	4
3.3.	IPtemIDList.....	5
3.4.	TPidMgr.....	6
3.5.	Interface IClassFactory.....	7
3.6.	Interface IShellFolder.....	8
3.7.	Interface IShellView.....	9
3.8.	Interface IExtractIcon.....	10
3.9.	Interface IEnumIDList.....	10
3.10.	Interface IDataObject / IEnumFormatETC.....	11
3.11.	Interface IDropTarget.....	12
3.12.	Interface IDropSource.....	13
3.13.	Interface IContextMenu.....	13
4.	Virtual Filesystem.....	15
5.	icoyaDrive ClassicView.....	16
6.	icoyaDrive Service.....	19
7.	Installationsprogramm.....	19
8.	Zukünftige Arbeiten / Ausblick.....	21
9.	Referenz.....	22
9.1.	Links.....	22
9.2.	Screenshots.....	22

2. Übersicht

icoyaDrive ist als Namespace Extension realisiert. Eine Namespace Extension ist eine DLL, die im System registriert wird und daraufhin einen virtuellen Ordner bereitstellt. Alle Zugriffe auf diesen Ordner laufen über die DLL.

icoyaDrive bietet folgende Möglichkeiten:

- Das Anlegen von Dokument-Containern. Dokument-Container verwalten die Versionen, Sprachen und Varianten der Dokumente die in ihnen gespeichert werden. Dokument-Container speichern außerdem Meta-Informationen wie Titel, Autor, etc. zu jedem Dokument. Diese Dokument-Container werden im icoyaDrive als Verzeichnis dargestellt.
- Das Erstellen und Bearbeiten von Dateiinhalten. Neue Dokumente können per Drag and Drop vom lokalen Dateisystem in das icoya Repository kopiert werden. Dabei werden auf Wunsch verschiedene Sprachen für das neue Dokument angelegt. Dateien, die in icoya gespeichert sind, können direkt bearbeitet werden. Die Änderungen werden von icoyaDrive wieder ins CMS hochgeladen wo automatisch eine neue Version erstellt wird.
- Das Anzeigen und Ändern einzelner Versionen / Sprachen. Zu jedem Dokument können die vorhandenen Versionen / Sprachen angezeigt und kann eine bestimmte Version / Sprache direkt bearbeitet werden.

Um das icoya CMS Repository auf icoyaDrive abzubilden wurde ein virtuelles Dateisystem implementiert, das über WebDAV und spezielle icoya Befehle mit dem icoya Server kommuniziert.

icoyaDrive wurde für Windows 2000 / XP entwickelt. Für die Anwendung von icoyaDrive wird der MS-Internet Explorer in der Version 4 oder höher vorausgesetzt.

2.1. Konventionen

Angelehnt an den Object Pascal Styleguide von Borland [1] werden folgende Konventionen verwendet:

Interfaces beginnen mit einem großen **I**
Typdeklarationen beginnen mit einem großen **T**
Pointer auf Typen beginnen mit einem großen **P**

3. Implementierung

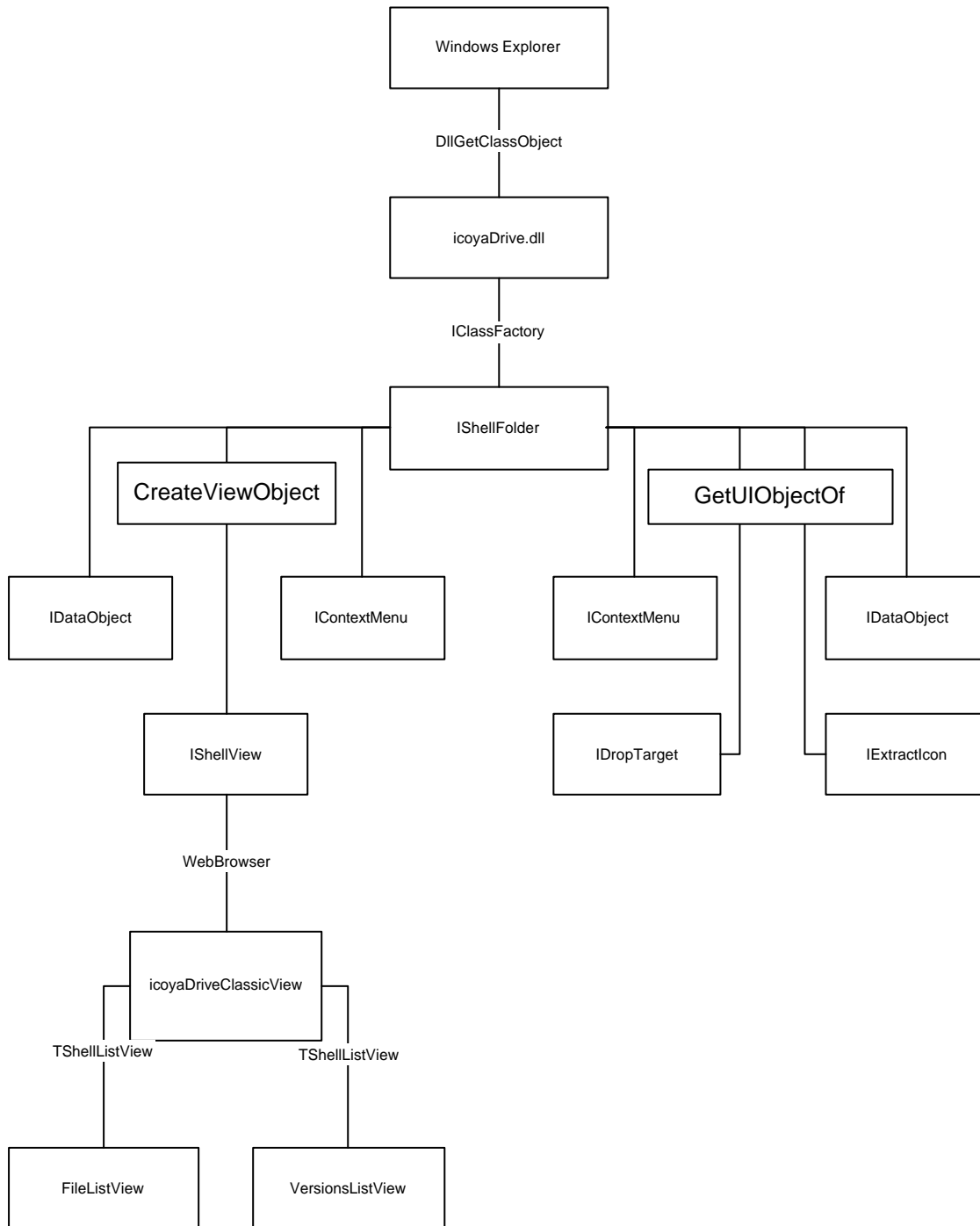
Eine Namespace Extension DLL muss eine Methode `DllGetObject` exportieren, die von Windows aufgerufen wird, um Zugriff auf die bereitgestellten Objekte der DLL zu bekommen. Alle weitere Kommunikation zwischen dem System und der DLL läuft daraufhin über diese Objekte ab, die im Folgenden beschrieben werden.

3.1. Speicherverwaltung

Alle Objekte von icoyaDrive benutzen den Memorymanager des Explorers um Speicher zu belegen bzw. wieder freizugeben. Das ist notwendig, da viele Speicherblöcke sowohl vom Explorer als auch von icoyaDrive benutzt und die Speicherblöcke zentral verwaltet werden müssen.

Der Memorymanager kann über `SHGetMalloc` ausgelesen werden und stellt ein Objekt bereit, das das `IMalloc` Interface [2] implementiert.

3.2. Klassendiagramm der implementierten Objekte



3.3. PItemIDList

Alle Objekte (Dateien / Ordner) im Windows Dateisystem werden durch eine Liste von `TItemID` Strukturen eindeutig identifiziert. Die Liste entspricht dem Pfad des Objekts ausgehend vom Desktop (der gleichzeitig die Wurzel des Verzeichnisbaums ist). Die Liste kann aus einer beliebigen Anzahl von `TItemID` Strukturen bestehen.

Jede `TItemID` entspricht dabei einer Ebene im Dateisystem. Die Größe jeder Struktur kann unterschiedlich sein und kann von der Anwendung frei definiert werden.

Der Aufbau einer `TItemID` ist wie folgt:

```
type
  TItemID = record
    cb: UShort;
    abID: array[0..0] of Byte;
  end;
```

`cb` enthält die Größe dieser `TItemID`
`abID` kann beliebige Daten enthalten

Der Typ `PItemIDList` deklariert einen Pointer auf eine Liste von `TItemID` Strukturen.

Beispiel für eine `PItemIDList`:

My Computer	C:\	MyDocuments	MyFile.html	2 Byte NULL
TItemID 1	TItemID 2	TItemID 3	TItemID 4	TItemID 5

In `icoyaDrive` werden in einer `TItemID` die Metadaten der Dokumente gespeichert.

```
type
  TPidlType = (ptFolder, ptFile, ptVersion);

  TTagPidlData = record
    ContentSize: Integer;
    LastModified: TDateTime;
    CID: array[0..MAX_PATH-1] of char;

    case Typ: TPidlType of
      ptFolder,
      ptFile:
        (Name:          array[0..MAX_PATH-1] of char;
         DisplayName:  array[0..MAX_PATH-1] of char;
         ContentType:  array[0..MAX_PATH-1] of char);

      ptVersion:
        (Owner:        PItemIDList;
         Version:      array[0..31] of char;
         Language:     array[0..7] of char;
         Author:       array[0..MAX_PATH-1] of char;
         Comment:      array[0..1024-1] of char);
    end;
```

3.4. TPidlMgr

Beschreibung

Das Objekt `TPidlMgr` stellt Methoden bereit um mit `PItemIDList` Strukturen zu arbeiten. In `icoyaDrive` gibt es drei verschiedene Typen von `PItemIDList` Strukturen:

- Verzeichnisse
- Dateien
- Versionen

Je nach Typ werden im benutzerdefinierten Bereich unterschiedliche Daten gespeichert wobei die Daten für Verzeichnisse und Dateien die gleichen sind. Jede Version hat zusätzlich zu den Meta-Daten noch einen Verweis auf die zugehörige Datei um Daten wie z.B. den Dateinamen auszulesen.

Erstellt und initialisiert werden die Listen von `IEnumIDList` Objekten, ausgelesen hauptsächlich vom `icoyaDriveClassicView`.

Methoden

Methode	Beschreibung
<code>Delete</code>	Löscht eine komplette Liste und gibt belegten Speicher frei.
<code>GetNextItem</code>	Gibt das nächste Element einer Liste zurück.
<code>Copy</code>	Gibt die Kopie einer Liste zurück.
<code>CopySingleItem</code>	Gibt die Kopie des ersten Elements einer Liste zurück.
<code>GetLastItem</code>	Gibt das letzte Element einer Liste zurück.
<code>Concatenate</code>	Hängt zwei Listen aneinander und gibt die zusammengesetzte Liste zurück.
<code>CreateFolderPID</code>	Erstellt ein neues Element, das als Verzeichnis benutzt werden kann.
<code>CreateFilePID</code>	Erstellt ein neues Element, das als Datei benutzt werden kann.
<code>CreateVersionPID</code>	Erstellt ein neues Element, das als Version benutzt werden kann.
<code>SetPidlDisplayName</code>	Setzt den Namen, der angezeigt wird.
<code>SetPidlContentSize</code>	Setzt die Größe der Datei / Version.
<code>SetPidlContentType</code>	Setzt den Content-Type der Datei. (z.B. <code>text/xml</code>)
<code>SetPidlLastModified</code>	Setzt das Datum der letzten Änderung.
<code>SetPIDLCID</code>	Setzt die <code>CID</code> der Datei.
<code>SetPidlVersionLang</code>	Setzt die Sprache der Version.
<code>SetPidlVersionComment</code>	Setzt den Kommentar der Version.
<code>SetPidlVersionAuthor</code>	Setzt den Author der Version.
<code>SetPidlVersionString</code>	Setzt die Versionsnummer.
<code>SetPidlVersionOwner</code>	Setzt die zugehörige Datei <code>PItemID</code> einer Version.
<code>IsFolder</code>	Prüft, ob eine <code>PItemIDList</code> ein Verzeichnis ist.
<code>IsVersion</code>	Prüft, ob eine <code>PItemIDList</code> eine Version ist.
<code>GetName</code>	Gibt den Dateinamen einer <code>PItemIDList</code> zurück.
<code>GetRelativeName</code>	Liefert den kompletten Namen einer <code>PItemIDList</code> .
<code>GetFileDisplayName</code>	Gibt den Namen einer <code>PItemIDList</code> zurück, der für die Darstellung verwendet werden kann. Bei Dateien / Ordnern ist das der <code>DisplayName</code> , bei Versionen hat er die Form <code><filename>-<date>-<version>-<language>.<ext></code>
<code>GetCID</code>	Gibt die <code>CID</code> einer Datei zurück.
<code>GetFileSize</code>	Gibt die Größe einer Datei / Version zurück.

GetFileIcon	Gibt das zu diesem Dateitypen gehörende Icon zurück.
GetFileDate	Gibt das Datum der letzten Änderung zurück.
GetFileType	Gibt den Dateitypen zurück. (z.B. „Textdatei“)
GetVersionLang	Gibt die Sprache einer Version zurück.
GetVersionComment	Gibt den Kommentar zu einer Version zurück.
GetVersionAuthor	Gibt den Author einer Version zurück.
GetVersionString	Gibt die Versionsnummer zurück.
GetVersionOwner	Gibt die zugehörige Datei <code>PItemIDList</code> zur Version zurück.
GetSize	Gibt die Größe (in Bytes) zurück, die eine <code>PItemIDList</code> benötigt.
GetFirstSize	Gibt die Größe (in Bytes) des ersten Elements einer <code>PItemIDList</code> zurück.

3.5. Interface `IClassFactory`

Beschreibung

Der Explorer kommuniziert über `IClassFactory` Objekte mit der `icoyaDrive` DLL. Ein Aufruf von `DllGetClassObject` der DLL liefert eine `IClassFactory` Instanz zurück, die über `CreateInstance` `IShellFolder` Objekte erzeugen und dem Explorer zurückgeben kann.

Methoden

Methode	Beschreibung
<code>CreateInstance</code>	Erstellt eine neue Instanz eines <code>IShellFolder</code> und gibt das angefragte Interface weiter an den <code>ShellFolder</code> .
<code>LockServer</code>	wird von <code>icoyaDrive</code> nicht unterstützt.

3.6. Interface IShellFolder

Beschreibung

Ein Ordner im Windows Explorer wird durch ein `IShellFolder` Objekt verwaltet. So kann über dieses Objekt der Inhalt des Ordners bestimmt / angezeigt werden (`EnumObjects / CreateViewObject`), können Elemente umbenannt werden (`SetNameOf`) und Meta-Daten der Elemente ausgelesen werden (`GetAttributesOf / GetDisplayNameOf`).

Wird im Explorer das aktuelle Verzeichnis gewechselt, wird vom übergeordneten Ordner des neuen Verzeichnisses über `BindToObject` das Objekt des neuen Ordners erstellt. Die Dateiansicht erstellt Windows durch einen Aufruf von `CreateViewObject` und der GUID von `IShellView` als Parameter. Jetzt kann der Inhalt mit Hilfe von `EnumObjects` bestimmt werden und in der Ansicht angezeigt werden.

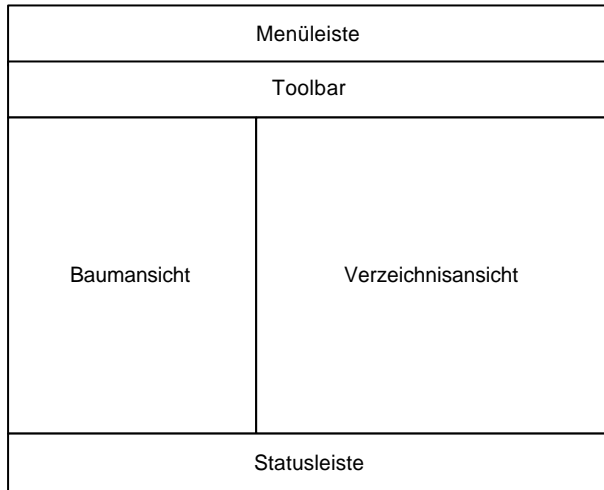
Alle Objekte, die den Inhalt des Ordners verändern können, werden von `GetUIObjectOf` initialisiert. Das sind vor allem `IContextMenu` (kann Dateien löschen, umbenennen und einfügen) sowie `IDropTarget` (fügt Dateien ein).

Methoden

Methode	Beschreibung
<code>GetClassID</code>	Gibt den Class Identifier (CLSID) des Objekts zurück.
<code>Initialize</code>	Initialisiert das Objekt mit den übergebenen Parametern.
<code>ParseDisplayName</code>	Transformiert den Namen eines Objekts in seine <code>PITEMIDLIST</code> . Wird von <code>icoyaDrive</code> nur unterstützt um die <code>PITEMIDLIST</code> des Ordners zu bekommen.
<code>EnumObjects</code>	Gibt ein <code>IEnumIDList</code> Objekt zurück, mit dem sich der Inhalt des Ordners bestimmen lässt.
<code>BindToObject</code>	Gibt das <code>IShellFolder</code> Objekt für einen bestimmten Unterordner zurück.
<code>BindToStorage</code>	wird von <code>icoyaDrive</code> nicht unterstützt.
<code>CompareIDs</code>	Vergleicht zwei <code>PITEMIDLIST</code> Strukturen und gibt die Reihenfolge zurück.
<code>CreateViewObject</code>	Erstellt ein Objekt für die Darstellung dieses Ordners bzw. der (markierten) Dateien im Ordner. Unterstützt werden <code>IShellView</code> , <code>IDataObject</code> und <code>IContextMenu</code> .
<code>GetAttributesOf</code>	Gibt Attribute eines Objekts in diesem Ordner zurück.
<code>GetUIObjectOf</code>	Gibt ein Objekt zurück, mit dem Operationen auf den Inhalt des Ordners ausgeführt werden können.
<code>GetDisplayNameOf</code>	Gibt den Name eines Objektes zurück, der für die Darstellung verwendet werden kann.
<code>SetNameOf</code>	Benennt ein Objekt in diesem Ordner um.

3.7. Interface IShellView

Beschreibung



Die Verzeichnisansicht im rechten Bereich des Explorers wird von einem `IShellView` Objekt gezeichnet. Es wird mit „seinem“ `IShellFolder` initialisiert und zeigt je nach Explorer Einstellung dessen Inhalt als Liste, Icons oder mit Details an.

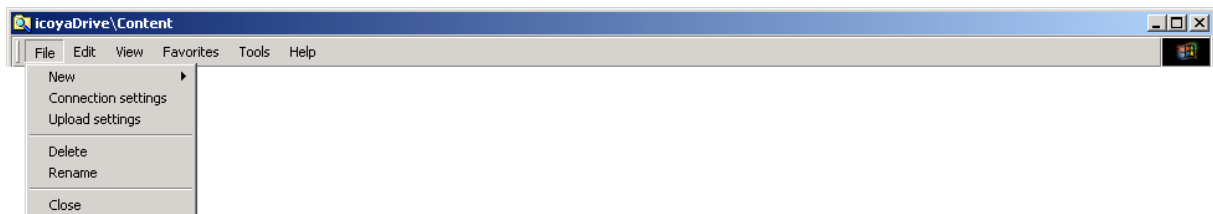
In icoyaDrive wird (wie in der Web-Ansicht des Explorers) für die Darstellung eine Instanz des Internet Explorers erzeugt, in der ein HTML Template geladen wird. Dieses HTML Template enthält ein ActiveX Objekt, das die Liste der Dateien / Ordner darstellt. Über DHTML können Meta-Daten der markierten Elemente abgefragt und die Ansicht dementsprechend geändert werden. Die Verwendung von DHTML hat den Vorteil, dass das Layout der

Ansicht sehr einfach an Kundenwünsche angepasst werden kann und nicht jedesmal der Sourcecode bearbeitet werden muss.

Weitere Informationen zum ActiveX Objekt: siehe 5. *icoyaDriveClassicView*.

Ist der Benutzer noch nicht am icoya Server angemeldet, wird statt des HTML Templates die Login-Seite von icoya geladen und über Callbacks die eingegebenen Werte (Benutzername und Passwort) ausgelesen.

Eine weitere Aufgabe der `IShellView` Objekte ist die Erweiterung der Explorer Menüs, der Toolbar und Statusleiste durch icoyaDrive spezifische Einträge wie z.B. „Connection Settings“ bzw. „Upload Settings“.



Screenshot 1 - Geändertes Explorer Menü bei markiertem Ordner

Methoden

Methode	Beschreibung
<code>GetWindow</code>	Gibt das Fensterhandle des ShellViews zurück.
<code>ContextSensitiveHelp</code>	wird von icoyaDrive nicht unterstützt.
<code>TranslateAccelerator</code>	Führt Tastenkombinationen aus falls diese von icoyaDrive unterstützt werden.
<code>EnableModeless</code>	wird von icoyaDrive nicht unterstützt.
<code>UIActivate</code>	Wird aufgerufen wenn sich der Fokus der Ansicht ändert.
<code>Refresh</code>	Zeichnet den Inhalt der Ansicht neu.
<code>CreateViewWindow</code>	Erstellt eine neue Ansicht eines Ordner.
<code>DestroyViewWindow</code>	Gibt belegten Speicher einer Ansicht frei.

GetCurrentInfo	Liefert Informationen über die aktuellen Einstellungen.
AddPropertySheetPages	wird von icoyaDrive nicht unterstützt.
SaveViewState	wird von icoyaDrive nicht unterstützt.
SelectItem	wird von icoyaDrive nicht unterstützt.
GetItemObject	wird von icoyaDrive nicht unterstützt.

3.8. Interface IExtractIcon

Beschreibung

Dieses Interface wird aufgerufen um Icons für die Dateien in der Verzeichnisansicht zu bekommen. Um das Icon für Dateien zu erhalten, die nicht über ihre Endung bestimmt werden können, wird ein Dateiname aus dem Content-Type der Datei erzeugt.

Beispiel:

```
text/xml      wird zu text.xml
text/plain    wird zu text.txt
```

Jetzt kann wieder anhand der Endung der Dateityp und damit das Icon bestimmt werden.

Methoden

Methode	Beschreibung
GetIconLocation	Liefert Datei und Index für ein bestimmtes Icon. Da icoyaDrive die Standard-Windows Icons benutzt, gibt die Methode nur einen Index zurück.
Extract	Gibt das Handle für ein Icon zurück.

3.9. Interface IEnumIDLList

Beschreibung

Mit Hilfe der `IEnumIDLList` Objekte können die Inhalte eines Ordners bestimmt werden. Erstellt werden sie durch einen Aufruf von `IShellFolder.EnumObjects`. Über das virtuelle Dateisystem wird der Inhalt des Verzeichnisses in icoya bestimmt und eine verkettete Liste mit `PITEMIDLList` Strukturen aller Dateien / Verzeichnisse relativ zu diesem Ordner erstellt. Der Aufrufer kann über die bereitgestellten Methoden auf dieser Liste navigieren.

Verwendet wird dieses Objekt vom `icoyaDriveClassicView` um die Liste der Dateien bereitzustellen und anzuzeigen.

Methoden

Methode	Beschreibung
Next	Liest die geforderte Anzahl Elemente aus der Liste.
Skip	Überspringt Elemente.
Reset	Setzt die Position des Lesezeigers auf den Anfang der Liste zurück.
Clone	Erstellt eine Kopie des Objekts.

3.10. Interface IDataObject / IEnumFormatETC

Beschreibung

IDataObject Objekte werden bei sämtlichen Kopieroperationen erzeugt, die von icoyaDrive ausgehen. Dazu gehören:

- Cut / Copy und Paste icoyaDrive > lokales Dateisystem
- Cut / Copy und Paste icoyaDrive > icoyaDrive
- Drag-and-Drop icoyaDrive > lokales Dateisystem
- Drag-and-Drop icoyaDrive > icoyaDrive

Bei Drag-and-Drop Operationen wird das IDataObject Objekt direkt übergeben, bei Operationen mit Cut bzw. Copy wird das IDataObject Objekt im Clipboard gespeichert.

Unterstützte Datenformate für QueryGetData / GetData:

Format	Beschreibung
CFSTR_ICOYAFILE "icoyaDrive Clipboard Format"	Wird bei Operationen innerhalb von icoyaDrive verwendet.
CF_TEXT	Wird verwendet wenn die URL eines Dokuments ermittelt werden soll.
CFSTR_FILEDESCRIPTORA "FileGroupDescriptor"	Werden bei Operationen mit dem Windows Explorer verwendet. Das Objekt liefert Informationen über die Dateien bzw. Stream Objekte auf den Inhalt der Dateien. Siehe auch [3].
CFSTR_FILEDESCRIPTORW "FileGroupDescriptorW"	
CFSTR_FILECONTENTS "FileContents"	
CFSTR_SHELLURL / CFSTR_INETURL "UniformResourceLocator"	Siehe CF_TEXT.
CFSTR_PREFERREDDROPEFFECT "Preferred DropEffect"	Wird verwendet um festzustellen, ob die Daten kopiert oder verschoben werden sollen.

Die URL eines Dokumentes bei CF_TEXT bzw. CFSTR_SHELLURL / CFSTR_INETURL wird in der Form `http://server:port/path/to/doc_id?cid=xxx&version=xxx&language=xxx` zurückgegeben. Durch die Angabe der CID kann auch noch auf das Dokument zugegriffen werden, wenn es innerhalb von icoya in ein anderes Verzeichnis kopiert wurde. Die CID (Content ID) eines in icoya gespeicherten Dokumentes ist eindeutig und unabhängig von der Verzeichnisstruktur, d.h. vom Ort des Dokumentes.

Die Version und Sprache sind optional und werden nur angegeben wenn eine bestimmte Versionen / Sprache eines Dokumentes kopiert wurde.

Informationen, die vom Format CFSTR_ICOYAFILE verwendet werden, enthalten die PItemIDList Strukturen der ausgewählten Dateien und ein Flag, das angibt ob kopiert oder verschoben werden soll. Der Speicherblock sieht dabei wie folgt aus:

Feld	Typ	Beschreibung
NumFiles	Integer	Gibt an wieviele Dateien enthalten sind. Ist die Zahl negativ, wird verschoben, sonst kopiert.
Path	array[0..MAX_PATH-1]	Der Pfad des Ordners in dem die Dateien liegen.

	of Char	
FilePids	array of PItemIDList	Kopien der PItemIDList Strukturen aller ausgewählten Dateien.

Unterstützte Datenformate für setData:

Format	Beschreibung
CFSTR_PASTESUCCEEDED "Paste Succeeded"	Wird gesetzt um zu signalisieren, dass die Kopieroperation abgeschlossen ist.
CFSTR_PERFORMEDDROPEFFECT "Performed DropEffect"	Werden gesetzt um zu signalisieren, ob die Dateien verschoben oder kopiert wurden. Da icoyaDrive ein eigenes „Dateisystem“ ist, kann der Explorer die Dateien nur kopieren und wird daher immer „Datei kopiert“ melden. icoyaDrive setzt je nach durchgeführter Operation das entsprechende Flag.
CFSTR_LOGICALPERFORMEDDROPEFFECT "Logical Performed DropEffect"	

Methoden

Methode	Beschreibung
Next	Liest die geforderte Anzahl Elemente aus der Liste.
Skip	Überspringt Elemente.
Reset	Setzt die Position des Lesezeigers auf den Anfang der Liste zurück.
Clone	Erstellt eine Kopie des Objekts.
GetData	Stellt die Daten im angeforderten Format bereit.
GetDataHere	wird von icoyaDrive nicht unterstützt.
QueryGetData	Überprüft ob die Daten im angeforderten Format bereitgestellt werden können.
GetCanonicalFormatEtc	Gibt ein dem angeforderten Format ähnliches (oder das gleiche) Format zurück.
SetData	Setzt ein bestimmtes Format im IDataObject.
EnumFormatEtc	Gibt ein IEnumFormatETC Objekt zurück, mit dem alle unterstützten Formate ermittelt werden können.
DAdvise	wird von icoyaDrive nicht unterstützt.
DUnadvise	wird von icoyaDrive nicht unterstützt.
EnumDAdvise	wird von icoyaDrive nicht unterstützt.

3.11. Interface IDropTarget

Beschreibung

IDropTarget Objekte werden benötigt um Dateien von einer beliebigen Datenquelle (z.B. dem lokalen Dateisystem) aufzunehmen und ins icoya CMS zu kopieren. Sie werden für jeden icoyaClassicView erzeugt um Drag-and-Drop Operationen mit dem Windows Explorer zu unterstützen. Auch bei Copy/Cut Paste Vorgängen werden IDropTarget Objekte verwendet, da die zu kopierenden Daten bei beiden im gleichen Format vorliegen und daher gleich angesprochen werden können (über ein IDataObject Objekt).

Das IDropTarget Objekt versucht die Daten in einem unterstützten Format zu bekommen und lädt dann je nach gewünschter Methode die Dateien hoch oder kopiert sie innerhalb von icoyaDrive.

Unterstützte Formate:

Format	Beschreibung
CFSTR_ICOYAFILE "icoyaDrive Clipboard Format"	Wird bei Operationen innerhalb von icoyaDrive verwendet. (Format: siehe 3.10. <i>IDataObject</i>)
CF_HDROP	Wird vom Explorer verwendet. Im Speicher werden die absoluten Pfade der zu kopierenden / verschiebenden Dateien abgelegt.

Wenn Dateien vom lokalen Dateisystem verschoben werden sollen und die Datenquelle es unterstützt, löscht icoyaDrive die Dateien nach dem Hochladen und informiert die Datenquelle darüber (siehe CFSTR_PERFORMEDDROPEFFECT bei 3.10. *IDataObject*).

Methoden

Methode	Beschreibung
DragEnter	Wird aufgerufen wenn bei einer Drag-and-Drop Operation der Cursor auf das Objekt bewegt wird.
DragOver	Wird aufgerufen während die Maus noch über dem Objekt ist.
DragLeave	Wird aufgerufen wenn die Maus vom Objekt wegbewegt wird.
Drop	Wird aufgerufen wenn die Maus den Inhalt der Drag-and-Drop Operation auf dem Objekt loslässt.

3.12. Interface *IDropSource*

Beschreibung

Ein *IDropSource* Objekt wird erzeugt, sobald eine Datei / ein Verzeichnis / eine Version von icoyaDrive per Drag-and-Drop „weggezogen“ wird. Während des DaD Vorgangs wird periodisch *QueryContinueDrag* aufgerufen um zu testen, ob der Benutzer die Operation fortsetzen will oder ob er sie abgebrochen hat (z.B. durch Drücken von Escape).

Methoden

Methode	Beschreibung
<i>QueryContinueDrag</i>	Wird aufgerufen um festzustellen ob die Drag-and-Drop Operation fortgesetzt werden soll.
<i>GiveFeedback</i>	Wird aufgerufen um dem Benutzer zu signalisieren, dass eine Drag-and-Drop Operation durchgeführt wird (z.B. durch einen anderen Mauscursor).

3.13. Interface *IContextMenu*

Beschreibung

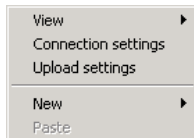
Um dem Benutzer ein Menü mit Befehlen zu den gerade ausgewählten Dateien bereitzustellen, wurde das *IContextMenu* Interface implementiert. Klickt der Benutzer mit der rechten Maustaste auf Datei(en) oder den Hintergrund im icoyaDrive, öffnet sich ein Menü abhängig von der aktuellen Markierung.

Sind eine oder mehrere Dateien markiert, öffnet sich folgendes Menü:

Die bereitgestellten Funktionen sind im großen und ganzen die gleichen wie im normalen Windows Kontextmenü. Zusätzlich kann der Benutzer von der Dateiansicht zur Versionsansicht umschalten und die verfügbaren Versionen einer Datei anschauen.

Klickt der Benutzer auf „Properties“, wird ein Formular geöffnet das in einem eingebetteten Webbrowser vom eingestellten icoya Server die Seiten mit den Eigenschaften lädt. Beispiele hierfür sind in 8. Screenshots.

Sind keine Dateien ausgewählt, ändert sich das Menü:



Hier kann der Benutzer die Verzeichnisansicht ändern (große / kleine Symbole, Liste oder Details), die Konfiguration von icoyaDrive bearbeiten oder neue Ordner anlegen. Sind kompatible Daten im Clipboard, ist außerdem der Menüpunkt „Paste“ verfügbar und es können die Daten im aktuellen Ordner eingefügt werden.

Bei einem Doppelklick auf eine Datei oder ein Verzeichnis wird die Standardmethode des Kontextmenüs ausgeführt:

- Verzeichnisse werden je nach Einstellung in Windows im gleichen bzw. in einem neuen Fenster geöffnet.
- Dateien werden in dem Programm geöffnet, das mit der entsprechenden Dateierweiterung verknüpft ist. Falls kein Programm mit der Endung verknüpft ist oder die Datei keine Endung hat, wird der „Öffnen mit...“ Dialog von Windows angezeigt und es kann ein Programm ausgewählt werden.

Methoden

Methode	Beschreibung
QueryContextMenu	Wird aufgerufen um einem Kontextmenü Einträge hinzuzufügen.
InvokeCommand	Führt einen Befehl des Kontextmenüs aus.
GetCommandString	Liefert den Befehlsnamen eines Befehlsindex.

4. Virtual Filesystem

Beschreibung

Die gesamte Kommunikation zwischen icoyaDrive und dem icoya Server läuft über HTTP ab. Um die Standard-Dateifunktionen bereitzustellen wurde ein virtuelles Dateisystem erstellt, das Funktionen wie z.B. „Datei kopieren“ oder „Ordner erstellen“ auf die entsprechenden WebDAV / icoyaDrive Befehle abbildet.

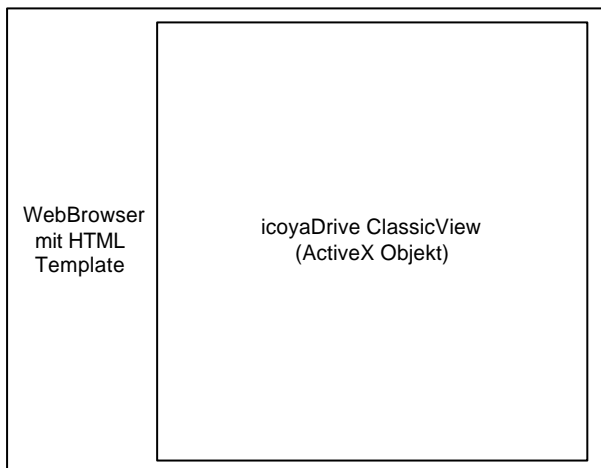
Die Authentifizierung mit dem Server läuft über die normale „basic authentication“ des HTTP Protokolls bzw. über verschlüsseltes SSL Protokoll.

Da der WebDAV Standard manche Funktionen nicht oder nur unzureichend definiert, wurden serverseitig neue Funktionen implementiert, um z.B. den Verzeichnisinhalt eines Ordners auszulesen oder die Versionen eines Dokuments zu erhalten. Als Ergebnis wird vom Server ein XML Stream zurückgeschickt, der die angeforderten Informationen beinhaltet.

Methoden

Methode	Beschreibung
Connect	Verbindet das virtuelle Dateisystem mit einem icoya Server.
Disconnect	Trennt die Verbindung mit dem Server.
FindFirst	Bestimmt den Inhalt eines Ordners, der einer Suchmaske entspricht (z.B. * . *) und liefert das erste Suchergebnis zurück.
FindNext	Gibt das nächste Suchergebnis zurück.
FindClose	Gibt den Speicher frei, der für die Suchergebnisse belegt wurde.
ChangeDirectory	Wechselt das aktuelle Verzeichnis auf dem Server.
CreateDirectory	Erstellt ein neues Verzeichnis.
OpenFile	Öffnet eine Datei vom Server und gibt das Handle auf den Stream dieser Datei zurück.
StoreFile	Lädt eine Datei auf den Server hoch und erstellt ggf. verschiedene Sprachen.
StoreDirectory	Lädt den Inhalt eines Verzeichnisses rekursiv auf den Server hoch.
DownloadFile	Speichert eine Datei vom Server auf der lokalen Festplatte.
Rename	Benennt eine Datei auf dem Server um.
CopyIDs	Kopiert Datei(en) / Verzeichnis(se) auf dem Server in ein anderes Verzeichnis.
DeleteIDs	Löscht Datei(en) / Verzeichnis(se) auf dem Server. Diese Datei(en) / Verzeichnis(se) werden nicht endgültig gelöscht und können wieder hergestellt werden.
FinalDeleteIDs	Löscht Datei(en) / Verzeichnis(se) endgültig vom Server.
GetVersions	Liefert ein XML Dokument, das Informationen über die Versionen einer Datei enthält.

5. icoyaDrive ClassicView



Um die Verzeichnisansicht möglichst leicht erweiterbar und änderbar zu machen, wurde angelehnt an die Web-Ansicht des Internet Explorers ein Webbrowser instantiiert, der in einem HTML Template ein eingebettetes ActiveX Objekt darstellt den icoyaDrive ClassicView.

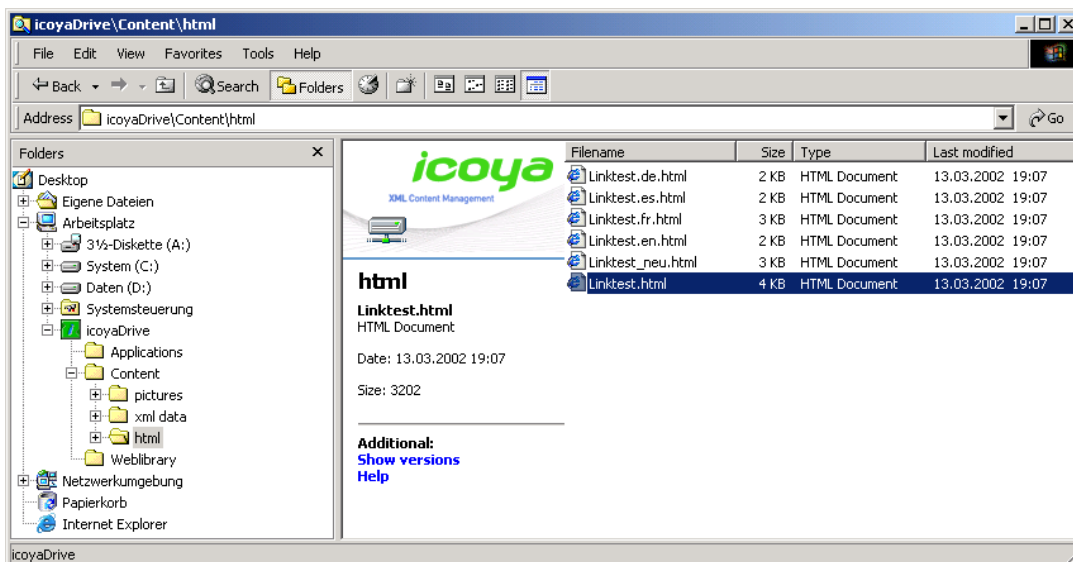
Grundsätzlich gibt es zwei Ansichten:

1. Die Dateien / Ordner als Liste
2. Die Versionen einer Datei

Das ActiveX Objekt stellt Callback Events bereit um dem Template mitzuteilen, dass sich die Auswahl geändert hat bzw. dass die Ansicht zwischen den

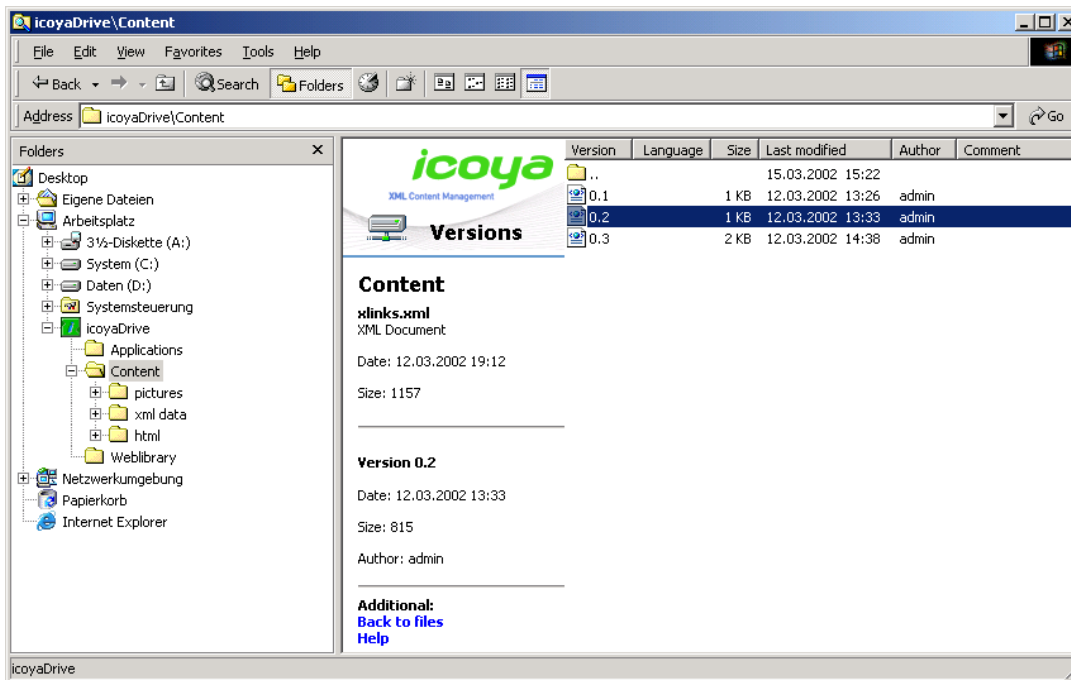
Dateien und den Versionen umgeschaltet wurde.

Jetzt können über JavaScript die Metadaten der ausgewählten Objekte abgefragt und per DHTML im Template angezeigt werden.



Screenshot 2 - Datei ausgewählt

Der ClassicView arbeitet intern mit zwei Listen (Dateien & Versionen), die je nach Ansicht umgeschaltet werden. Abhängig von der Ansicht und der ausgewählten Date bzw. Version werden vom Template Informationen wie Dateityp, Datum der letzten Änderung, Größe, etc. angezeigt.



Screenshot 3 - Version einer Datei ausgewählt

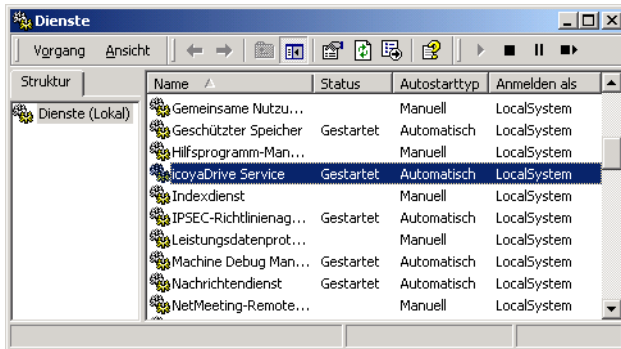
Methoden

Methode	Beschreibung
CreateView	Erstellt eine neue Instanz des <code>icoyaDriveClassicView</code> und zeigt sie in einem HTML Objekt an.
DestroyView	Gibt Speicher eines <code>icoyaDriveClassicView</code> frei.
Refresh	Zeichnet das Objekt neu.
GetWindow	Liefert das Windowhandle des Objekts zurück.
GetCurrentInfo	Gibt die Ansichtsparameter (Icons, Liste, Details) zurück.
SetCurrentInfo	Setzt die Ansichtsparameter (Icons, Liste, Details)
GetSelectedItems	Gibt eine Liste der markierten Einträge und die Position des ersten Eintrags zurück.
AddItem	Fügt der Liste einen Eintrag hinzu.
DeleteItem	Löscht einen Eintrag aus der Liste
EditItem	Markiert einen Eintrag zur Bearbeitung (d.h. der Benutzer kann den Namen ändern)
RenameItem	Benennt einen Eintrag um.
UpdateItem	Initialisiert einen Eintrag neu.
GetDetailsOf	Gibt Meta-Informationen wie Name, Größe, Datum, etc. eines Eintrags zurück.
SetVisibility	Schaltet die Ansicht um und zeichnet das Objekt neu.
SelectedSize	Gibt die Größe eines markierten Objekts zurück.
SelectedName	Gibt den Namen eines markierten Objekts zurück.
SelectedVersion	Gibt die Versionsnummer eines markierten Objekts zurück.
SelectedLanguage	Gibt die Sprache eines markierten Objekts zurück.
SelectedItem	Gibt ein markiertes Objekt zurück.
ResetItemStates	Setzt die Zustände der Einträge zurück (markiert / ausgeschnitten)
MarkItemAsCut	Markiert Einträge um dem Benutzer zu signalisieren, dass sie ausgeschnitten werden.
GetHelpURL	Gibt die URL der Hilfe zurück.

IsFolderSelected	Gibt an, ob mindestens ein Ordner markiert ist.
Get_SelectedItems	Liefert die Anzahl der markierten Elemente zurück.
Get_ItemCount	Liefert die Anzahl der Elemente insgesamt zurück.
Get_ / Set_DisplayType	Liefert / Setzt den Ansichtsmodus (Dateien / Versionen)

6. icoyaDrive Service

Da icoyaDrive nicht als richtiges Netzlaufwerk im System registriert ist, der Benutzer aber trotzdem direkt Dateien bearbeiten können soll, musste eine andere Lösung gefunden werden. Wenn doppelt auf eine Datei geklickt wird, lädt icoyaDrive diese Datei in ein temporäres Verzeichnis runter und öffnet die Datei mit dem zugehörigen Programm von dort. Ist dieser Datei keine Anwendung zugeordnet, wird der „Öffnen mit“ Dialog angezeigt und der Benutzer kann einen Editor auswählen.



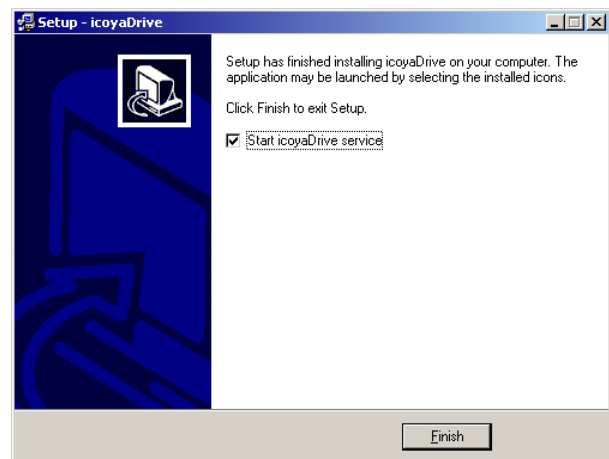
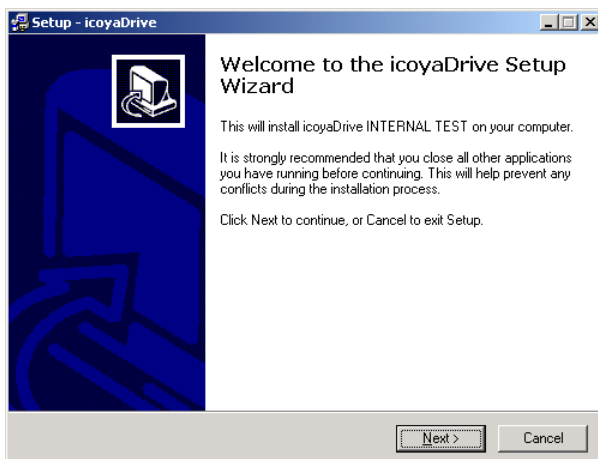
Der icoyaDrive Service ist ein Dienst, der automatisch gestartet wird und der den Inhalt des temporären Verzeichnisses von icoyaDrive auf Änderungen überwacht. Hat der Benutzer eine Datei aus icoyaDrive geöffnet, bearbeitet und speichert diese Datei jetzt ab, wird im Service ein Event ausgelöst. Der Service überprüft mit Hilfe einer Checksumme, ob sich der Inhalt der Datei geändert hat und lädt die Datei ggf. als neue Version wieder hoch.

Statt direkt eine Datei auszuwählen, kann der Benutzer auch in der Versionsansicht eine ältere Version oder nur eine bestimmte Sprache öffnen. Bei Änderungen wird in diesem Fall eine neue Version der ausgewählten Sprache erstellt.

icoyaDrive und der icoyaDrive Service kommunizieren über eine TCP Verbindung um Änderungen der Konfiguration mitzuteilen.

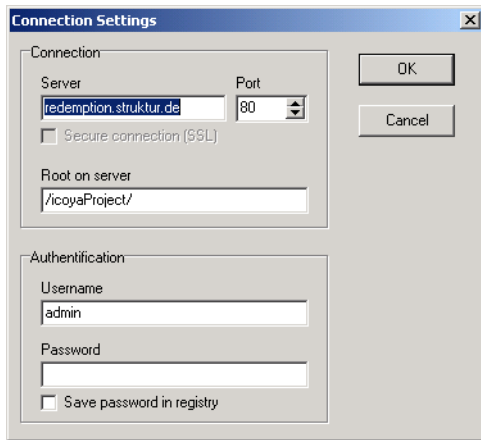
7. Installationsprogramm

Das Installationsprogramm wurde mit Inno Setup [4] erstellt, einem freien Installations-generator, der mit komplettem Sourcecode verfügbar ist. Neben einem Uninstaller bietet er die Möglichkeit, Dateien nach der Installation im System zu registrieren und Programme auszuführen.

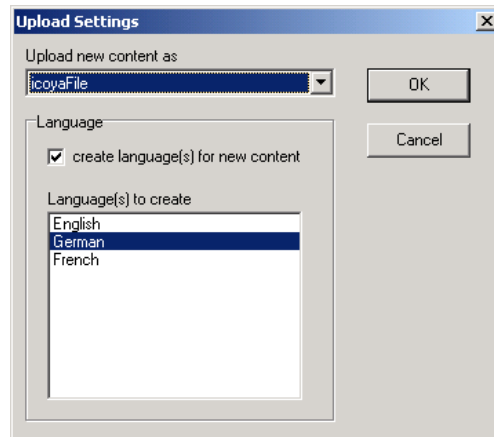


Screenshots 4 / 5 - Installationsprogramm

Konfigurationsformulare:

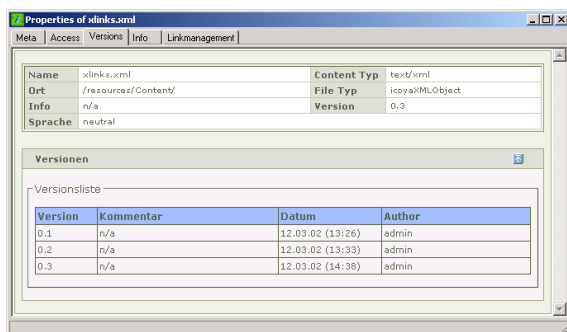


Screenshot 6 - Verbindungseinstellungen

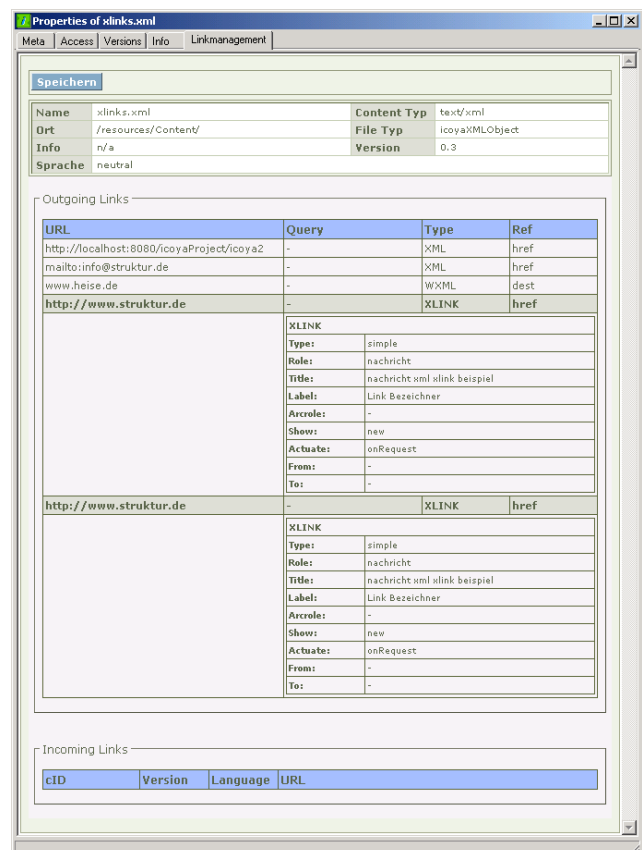


Screenshot 7 - Uploadeinstellungen

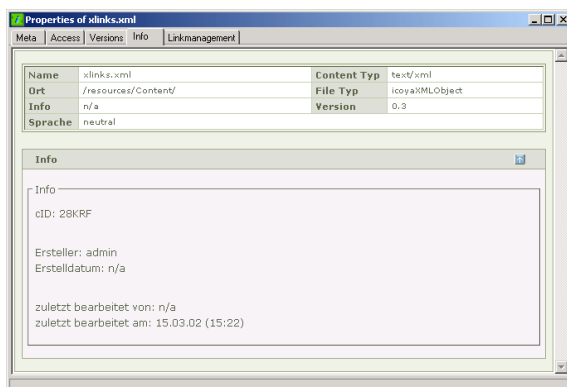
Eigenschaften einer Datei:



Screenshot 8 - Versionen



Screenshot 10 - ein- und ausgehende Links



Screenshot 9 - allgemeine Informationen

8. Zukünftige Arbeiten / Ausblick

- Die wichtigste fehlende Funktionalität ist die Implementierung als „echtes“ Laufwerk. Momentan kann nur über den Date Explorer auf icoyaDrive zugegriffen werden, nicht aber von beliebigen Anwendungen aus mit dem „Datei öffnen“ Dialog.
Hierfür ist ein Treiber notwendig, der über WebDAV das icoya Repository in das System einbindet. Allerdings ist die Programmierung eines Treibers aufwendig und muss auf Kernel-Ebene in C realisiert werden.
- Ein weiterer Punkt der fehlt, ist die Möglichkeit Dateien innerhalb des icoya Systems zu suchen. Da hier auf der Server-Seite die Implementierung fehlt, kann eine Suchfunktion in icoyaDrive erst später eingebaut werden.
- In einem Content Management System ist es außerdem wichtig, Dateien zu locken, d.h. vor Änderung durch andere Benutzer zu schützen während Daten bearbeitet werden.
- Um auch Clients hinter einer Firewall die Möglichkeit zu geben, auf das icoya Repository zugreifen zu können, muss eine Unterstützung für Verbindungen über Proxies eingebaut werden.

9. Referenz

9.1. Links

[1] Object Pascal Styleguide

<http://community.borland.com/article/0,1410,10280,00.shtml>

[2] IMalloc Interface

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wcedcom/htm/cerefIMalloc.asp>

[3] Shell Clipboard Formats

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/Shell/programmersguide/shell_basics/shell_basics_programming/transferring/clipboard.asp

[4] Inno Setup

<http://www.innosetup.com/>

9.2. Screenshots

Screenshot 1 - Geändertes Explorer Menü bei markiertem Ordner	9
Screenshot 2 - Datei ausgewählt.....	16
Screenshot 3 - Version einer Datei ausgewählt.....	17
Screenshots 4 / 5 - Installationsprogramm.....	19
Screenshot 6 - Verbindungseinstellungen.....	20
Screenshot 7 - Uploadeinstellungen.....	20
Screenshot 8 - Versionen.....	20
Screenshot 9 - allgemeine Informationen.....	20
Screenshot 10 - ein- und ausgehende Links.....	20