VII. Anhang

a. Webshop Web Service Code

i. Klasse FileHandler

```
import java.io.File;
import java.util.ArrayList;
import java.util.List;
import java.util.Stack;


public class FileHandler {
  public List<File> find( String start, String... extensions )
      {
        final List<File> files = new ArrayList<File>( 1024 );
        final Stack<File> dirs = new Stack<File>();
        final File startdir = new File( start );
        if ( startdir.isDirectory() )
          dirs.push( startdir );
        while ( dirs.size() > 0 )
        {
          for ( File file : dirs.pop().listFiles() )
          {
            if ( file.isDirectory() )
              dirs.push( file );
            else
              if ( match(file.getName(), extensions) )
                files.add( file );
          }
        }
        return files;
      }
```

```
    private static boolean match( String s, String... suffixes )
    {
      for ( String suffix : suffixes )
        if ( s.length() >= suffix.length() && s.substring(s.length() - suffix.length(),s.length()).equalsIgnoreCase(suffix) )
          return true;
      return false;
    }
  }
```

Abbildung 19: Klasse FileHandler

ii. Klasse Fachdokument

```
import java.util.HashMap;

import java.util.Map;


public class Fachdokument {

    String id;

    String titel;

    String kurzbeschreibung;

    String[] thematischeZuordnung;

    String[] funktionaleKlassifikation;

    String[] schlagworte;

    String[] fachobjektbezug;

    String urlMetaBeschreibungsseite;

    String[] fachdokumentenDatei;

    String datumLetzteAenderung;

    Map metaData = null;


    /**
     * Gibt das Datum der letzten Änderung zurück
     * @return   String
     */
    public String getDatumLetzteAenderung() {

            return datumLetzteAenderung;

    }
```

```
/**
 * Setzt das Datum der letzten Änderung
 * @param  _datumLetzteAenderung
 */
public void setDatumLetzteAenderung(String _datumLetzteAenderung) {
	this.datumLetzteAenderung = _datumLetzteAenderung;
}

/**
 * Gibt Fachobjektbezug zurück
 * @return   String []
 */
public String[] getFachobjektbezug() {
	return fachobjektbezug;
}

/**
 * Setzt Fachobjektbezug
 * @param  _fachobjektbezug
 */
public void setFachobjektbezug(String[] _fachobjektbezug) {
	this.fachobjektbezug = _fachobjektbezug;
}
```

```
/**
 * Setzt funktionale Klassifikation
 * @return   String []
 */
public String[] getFunktionaleKlassifikation() {
        return funktionaleKlassifikation;
}

/**
 * Setzt funktionale Klassifikation
 * @param  _funktionaleKlassifikation
 */
public void setFunktionaleKlassifikation(String[] _funktionaleKlassifikation) {
        this.funktionaleKlassifikation = _funktionaleKlassifikation;
}

/**
 * Gibt Id der Publikation zurück
 * @return   String
 */
public String getId() {
        return id;
}
```

```
/**
 * Setzt die Id
 * @param  _id
 */
public void setId(String _id) {
        this.id = _id;
}

/**
 * Gibt Kurzbeschreibung zurück
 * @return   String
 */
public String getKurzbeschreibung() {
        return kurzbeschreibung;
}

/**
 * Setzt Kurzbeschreibung
 * @param  _kurzbeschreibung
 */
public void setKurzbeschreibung(String _kurzbeschreibung) {
        this.kurzbeschreibung = _kurzbeschreibung;
}
```

```
/**
 * Gibt Schlagworte zurück
 * @return   String []
 */
public String[] getSchlagworte() {
        return schlagworte;
}

/**
 * Setzt Schlagworte
 * @param  _schlagworte
 */
public void setSchlagworte(String[] _schlagworte) {
        this.schlagworte = _schlagworte;
}

/**
 * Gibt thematische Zuordnung zurück
 * @return   String []
 */
public String[] getThematischeZuordnung() {
        return thematischeZuordnung;
}
```

```
/**
 * @param  _thematischeZuordnung
 */
public void setThematischeZuordnung(String[] _thematischeZuordnung) {
	this.thematischeZuordnung = _thematischeZuordnung;
}

/**
 * Gibt Titel zurück
 * @return   String
 */
public String getTitel() {
	return titel;
}

/**
 * Setzt Titel
 * @param  _titel
 */
public void setTitel(String _titel) {
	this.titel = _titel;
}
```

```
/**
 * Gibt Url der Metadaten - Beschreibungsseite zurück
 * @return   String
 */
public String getUrlMetaBeschreibungsseite() {
        return urlMetaBeschreibungsseite;
}

/**
 * Setzt Url der Metadaten - Beschreibungsseite
 * @param  _urlMetaBeschreibungsseite
 */
public void setUrlMetaBeschreibungsseite(String _urlMetaBeschreibungsseite) {
        this.urlMetaBeschreibungsseite = _urlMetaBeschreibungsseite;
}

/**
 * Gibt Metadaten gepackt als Map zurück
 * @return   Map
 */
public Map getMetaData() {
        return metaData;
}
```

```
/**
 * Packt alle Metadaten in eine HashMap
 */
public void setAllMetaData() {
	this.metaData = new HashMap();
	this.metaData.put("id", getId());
	this.metaData.put("titel", getTitel());
	this.metaData.put("kurzbeschreibung", getKurzbeschreibung());
	this.metaData.put("thematischeZuordnung", getThematischeZuordnung());
	this.metaData.put("funktionaleKlassifikation", getFunktionaleKlassifikation());
	this.metaData.put("schlagworte", getSchlagworte());
	this.metaData.put("fachobjektbezug", getFachobjektbezug());
	this.metaData.put("urlMetaBeschreibungsseite", getUrlMetaBeschreibungsseite());
	this.metaData.put("fachdokumentDateien", getFachdokumentenDatei());
	this.metaData.put("datumLetzteAenderung", getDatumLetzteAenderung());

}

/**
 * Gibt Metadaten von Publikation zurück
 * @return Map
 */
public Map getAllMetaDaten() {
	return this.metaData;
}
```

```
    /**
     * @param  _fachdokumentenDatei
     */
    public void setFachdokumentenDatei(String[] _fachdokumentenDatei) {
            this.fachdokumentenDatei = _fachdokumentenDatei;
    }

    /**
     * @return   Returns the fachdokumentenDatei.
     */
    public String[] getFachdokumentenDatei() {
            return fachdokumentenDatei;
    }
}
```

Abbildung 20: Klasse Fachdokument

iii. Klasse WebShopWS

```
import EDispatch.Remote.ServerConnection;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.net.MalformedURLException;

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.util.Enumeration;

import java.util.HashMap;

import java.util.Map;

import java.util.Properties;

import javax.xml.rpc.ServiceException;


public class WebShopWS {


    /* ++++ Werte werden aus Properties Datei geholt (service.properties) ++++ */

    Properties myProperties = new Properties();

    String shopId = null;

    String user = null;

    String pwd = null;

    String serverURL = null;

    String serverPort = null;

    String[] idList = null;

    Map themeMap = new HashMap();

    String fachObjectBezug = null;

    String entryCategory = null;

    String funcionalClassification = null;

    String shopName = null;

    String functionId = null;
```

```
String docId = null;
String myPropFile = null;
Boolean communicationType = null; // RMI(true) oder SOAP(false); Default SOAP (false)

/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
// ArrayList FachdokumentenListe = new ArrayList();
Fachdokument specialFachdokument = new Fachdokument();

public WebShopWS() {
}

/**
 * Gibt alle Metadaten einer bestimmten Publikation zurück. Ist eine Web
 * Service Methode
 *
 * @param docId
 *            Id der Publikation
 * @return Map
 */
public Map getMetaDataFromId(String _docId, String _shopName) {
        // speichert Metadaten aus Fachdokument mit bestimmter Id (docId) in
        // specialFachdokumentObjekt
        checkParams(_docId, _shopName);
        this.functionId = "metaDataFromId";
        readProperties(this.functionId);
        try {
                ServerConnection con = getConnectionToServer(serverURL, serverPort,
                                user, pwd);
                setFachdokument(this.docId, specialFachdokument, con);
                con.disconnect();
```

```
                specialFachdokument.setAllMetaData();
                return specialFachdokument.getAllMetaDaten();
        } catch (RemoteException e) {
                e.printStackTrace();
        } catch (MalformedURLException e) {
                e.printStackTrace();
        } catch (NotBoundException e) {
                e.printStackTrace();
        } catch (ServiceException e) {
                e.printStackTrace();
        }
        return null;
}

/**
 * Gibt alle Publikations Id's zurück. Ist eine Web Service Methode
 *
 * @return String []
 */
public String[] getAllId(String _shopName) {
        this.shopName = _shopName;
        this.functionId = "allId";
        readProperties(this.functionId);
        System.out.println(shopName);
        try {
                ServerConnection con = getConnectionToServer(serverURL, serverPort,
                                user, pwd);
                setIdList(con);
                con.disconnect();
                return idList;
```

```
        } catch (RemoteException e) {
                e.printStackTrace();
        } catch (MalformedURLException e) {
                e.printStackTrace();
        } catch (NotBoundException e) {
                e.printStackTrace();
        } catch (ServiceException e) {
                e.printStackTrace();
        }
        return null;
}

/**
 * Überprüft welcher String die docId und welcher der Shop Name ist.
 *
 * @param a
 *          Stringparameter von Aufruf der Web Service Methode
 *          getAllMetaDataFromId
 * @param b
 *          Stringparameter von Aufruf der Web Service Methode
 *          getAllMetaDataFromId
 */
private void checkParams(String a, String b) {

        try {
                Integer.valueOf(a).intValue();
                this.docId = a;
                this.shopName = b;
        } catch (Exception e) {
                this.docId = b;
```

```
                this.shopName = a;
        }
}

/**
 * Wandelt String in Boolean Wert um.
 *
 * @param _boolValue
 *          String
 * @return Boolean
 */
private Boolean getBoolValue(String _boolValue) {

        if (_boolValue.equals("1") || _boolValue.toLowerCase().equals("true")) {
                return true;
        } else if (_boolValue.equals("0") || _boolValue.toLowerCase().equals("false")) {
                return false;
        } else {
                System.err.println("keinen richtigen Booleanwert angegeben. '0', 'false', '1', 'true' verwenden");
                return null;
        }
}

/**
 * Liest service.properties Datei und setzt die Attribute
 * (serverURL,ServerPort usw.)
 */
private void readProperties(String functionId) {
        Properties prop = new Properties();
    String path = new File(System.getProperty("user.dir")).getAbsolutePath();
```

```
 // System.out.println( "Looking in path: " + path );
FileHandler fh = new FileHandler();
List<File> files = fh.find( path,  ".properties" );
 for ( File f : files ){
    //System.out.println( f.getAbsolutePath() );
    //System.out.println(f.getName());
    if (f.getName().equalsIgnoreCase(this.shopName.toLowerCase() + "_"+ "service.properties")) {
                        this.myPropFile = f.getAbsolutePath();
                        break;
                }
 }
    try {
                prop.load(new FileInputStream(this.myPropFile));
                System.out.println(prop.getProperty("$communicationType"));
                System.out.println(prop.getProperty("$user"));
                this.user = prop.getProperty("$user");
                this.pwd = prop.getProperty("$pwd");
                this.serverPort = prop.getProperty("$serverPort");
                this.serverURL = prop.getProperty("$serverUrl");
                this.shopId = prop.getProperty("$shopId");
                this.entryCategory = prop.getProperty("$entryCategory");
                this.communicationType = getBoolValue(prop
                                .getProperty("$communicationType")); // defaul SOAP (false)
                if (functionId.equals("metaDataFromId")) {
                        // findet automatisch die Zuordnungen der Themen aus Properties - Datei
                        Enumeration e = prop.propertyNames();
                        String[] propertiesElements = new String[prop.size()];
                        int i = 0;
                        while (e.hasMoreElements()) {
                                propertiesElements[i] = (e.nextElement()).toString();
```

```
                        if (!(propertiesElements[i].startsWith("$"))) {
                                i++;
                        }
                }
                for (int j = 0; j < propertiesElements.length; j++) {
                        if (!(propertiesElements[j] == null)) {
                                this.themeMap.put(propertiesElements[j], prop.get(propertiesElements[j]));
                        }
                }
                this.fachObjectBezug = prop.getProperty("$Fachobjektbezug");
                this.funcionalClassification = prop.getProperty("$Funktionale_Klassifikation");
        }

    } catch (FileNotFoundException e) {
            e.printStackTrace();
            getDefaultProperties(prop);
    } catch (IOException e) {
            e.printStackTrace();
            getDefaultProperties(prop);
    }
}

/**
 * Wenn Fehler beim lesen der Properties - Datei auftreten werden
 * Standartwerte gesetzt
 *
 * @param prop
 */
private void getDefaultProperties(Properties prop) {
    System.out.println("Durch einen Lesefehler der Propertie-Datei, wurden Defaultwerte gesetzt.");
```

```
        this.user = prop.getProperty("$user", "lesen-shop");
        this.pwd = prop.getProperty("$pwd", "lesen-shop");
        this.serverPort = prop.getProperty("$serverPort", "80");
        this.serverURL = prop.getProperty("$serverUrl",          "www.lubw.baden-wuerttemberg.de");
        this.shopId = prop.getProperty("$shopId", "6638");
        this.communicationType = getBoolValue(prop.getProperty("$communicationType", "0")); // defaul SOAP(false)
        this.entryCategory = prop.getProperty("$entryCategory", "19");
        // findet automatisch die Zuordnungen der Themen aus Properties - Datei
        Enumeration e = prop.propertyNames();
        String[] propertiesElements = new String[prop.size()];
        int i = 0;
        while (e.hasMoreElements()) {
                propertiesElements[i] = (e.nextElement()).toString();
                if (!(propertiesElements[i].startsWith("$"))) {
                        i++;
                }
        }

        for (int j = 0; j < propertiesElements.length; j++) {
        if (!(propertiesElements[j] == null)) {
        this.themeMap.put(propertiesElements[j], prop.getProperty(propertiesElements[j], "ist keinem Thema zugeornet"));
                }

        }
        this.fachObjectBezug = prop.getProperty("$Fachobjektbezug", "null");
        this.funcionalClassification = prop.getProperty("$Funktionale_Klassifikation", "null");
}

/**
 * Castet String aus service.properties - Datei in String Array
```

```
 *
 * @param propString
 * @return String []
 */
private String[] castToStringArray(String propString) {
        String[] tempStringArray = propString.split(";");
        return tempStringArray;
}


/**
 * Setzt die Attribute des Fachdokumentenobjekts
 *
 * @param docId
 *          Id der Publikation
 * @param fd
 *          Fachdokumentenobjekt
 * @param con
 *          ServerConnectionobjekt
 */
private void setFachdokument(String docId, Fachdokument fd,ServerConnection con) {
        try {
                HashMap params = new HashMap();
                Map result = con.execute("entry", docId, "get", params);
                // set metadata
                fd.setId(docId);
                fd.setTitel(result.get("title").toString());
                fd.setKurzbeschreibung(result.get("Abstract").toString());
                fd.setFachdokumentenDatei(getFileURI(docId, con));
                fd.setSchlagworte(getKeyWords(result, params, con));
                fd.setDatumLetzteAenderung(result.get("modification").toString());
```

```
                fd.setThematischeZuordnung(getThemeAsscociate(result, params, con));
                fd.setFachobjektbezug(getFachObjektBezug());
                fd.setFunktionaleKlassifikation(getFunctionaleClassification());
                fd.setUrlMetaBeschreibungsseite(getUrlMetaDataDescriptionSite(docId));
        } catch (RemoteException e) {
                e.printStackTrace();
        }
}



/**
 * Knüpft die WebShop Themen an die Portal-U Themen
 *
 * @param result
 *          Entry Objekt mit allen Entry attributen (title, parent usw.)
 * @param params
 *          Map mit den Parametern des Entry Object
 * @param con
 *          ServerConnection
 * @return String[]
 */
private String[] getThemeAsscociate(Map result, HashMap params,
                ServerConnection con) {
        try {
                String parentID = ((Integer) (result.get("parent"))).toString();
                result = con.execute("entry", parentID, "get", params);
                String shopCategory = (result.get("title")).toString();
```

```
                return castToStringArray(this.themeMap.get(shopCategory.replace(" ", "_")).toString());
        } catch (RemoteException e) {
                e.printStackTrace();
                return null;
        }
}
/**
 * Setzt idList
 *
 * @param con
 *          ServerConnection
 */
private void setIdList(ServerConnection con) {
        try {
                HashMap params = new HashMap();
                params.put("categories", this.entryCategory);
                params.put("subTree", Integer.valueOf(shopId));
                params.put("useSubTree", Boolean.valueOf(true));
                Map result;
                result = con.execute("entry", "", "search", params);
```

```
                // get list of publication. The Id's from the publications are
                // stored resultsSE. resultSE returned an int[]
                // result.get() returned an Object. This Object must be cast.
                // Description about entry methods include in the developerhandbook from WebGenesis
                int[] myIdList = (int[]) (result.get("resultsSE"));
                // cast the list of int [] to String []
                this.idList = new String[myIdList.length];
                if (myIdList != null) {
                        for (int i = 0; i < myIdList.length; i++) {
                                this.idList[i] = (Integer.valueOf(myIdList[i])).toString();
                        }
                } else {
                        this.idList = null;
                }
        } catch (RemoteException e) {
                e.printStackTrace();
        }
}

/**
 * Gibt Liste mit "Fachobjektbezügen" zurück
 *
 * @return String []
 */
private String[] getFachObjektBezug() {

        if (castToStringArray(this.fachObjectBezug) == null
                        || castToStringArray(this.fachObjectBezug)[0].equals("")) {
                return null;
        } else {
```

```
                return castToStringArray(this.fachObjectBezug);
        }
}

private String getUrlMetaDataDescriptionSite(String docId) {
        return "http://" + this.serverURL + "/servlet/is/" + docId + "/";
}

/**
 * Gibt Liste mit "funktionalen Klassifikationen" zurück
 *
 * @return String []
 */
private String[] getFunctionaleClassification() {

if (castToStringArray(this.funcionalClassification) == null || castToStringArray(this.funcionalClassification)[0].equals(""))
        {
                return null;
        } else {
                return castToStringArray(this.funcionalClassification);
        }
}

/**
 * Gibt "Schlagworte" zurück
 *
 * @param result
 *          Entry Objekt mit allen Entry attributen (title, parent usw.)
 * @param params
 *          Map mit den Parametern des Entry Object
```

```
 * @param con
 *            ServerConnection
 * @return String []
 * @throws RemoteException
 */
private String[] getKeyWords(Map result, HashMap params, ServerConnection con) throws RemoteException {
        int[] schlagwortListeId = (int[]) (result.get("allDescriptors"));
        if (schlagwortListeId == null) {
                return null;
        } else {
                String schlagwort = new String();
                String[] schlagwortListe = new String[schlagwortListeId.length];
                for (int i = 0; i < schlagwortListeId.length; i++) {
                        String descriptionId = Integer.toString(schlagwortListeId[i]);
                        Map result1 = con  .execute("entry", descriptionId, "get", params);
                        schlagwortListe[i] = result1.get("term").toString();
                }
                return schlagwortListe;
        }
}


/**
 * Gibt Dateinamen von "Fachdokumentendateien" einer Publikation zurück
 *
 * @param docId
 * @param con
 * @return String []
 * @throws RemoteException
 */
private String[] getFileURI(String docId, ServerConnection con) throws RemoteException {
```

```
            HashMap params = new HashMap();
            Map result = con.execute("entry", docId, "getContentFileInfos", params);
            String[] dateiListe = (String[]) (result.get("name"));
            if (dateiListe == null || dateiListe.equals("") || dateiListe.length == 0) {
                        return null;
            } else {
                        // konkatinieren der einzelnen Name aus dateiListe[] zur aufrufbaren URI
                        String[] dateiListenArray = new String[dateiListe.length];
                        String newDateiURI = "http://" + serverURL + "/servlet/is/" + docId
                                    + "/" + dateiListe[0].toString()
                                    + "?command=downloadContent&filename="
                                    + dateiListe[0].toString();
                        dateiListenArray[0] = newDateiURI;
                        for (int i = 1; i < dateiListe.length; i++) {
//http://localhost/servlet/is/272/angebote_lfu_landesverwaltung.pdf?command=downloadContent&filename=angebote_lfu_landesver
waltung.pdf
                                    newDateiURI = "http://" + serverURL + "/servlet/is/" + docId
                                                + "/" + dateiListe[i].toString()
                                                + "?command=downloadContent&filename="
                                                + dateiListe[i].toString();
                                    dateiListenArray[i] = newDateiURI;
                        }
                        return dateiListenArray;
            }
      }

      /**
       * Gibt ServerConnection zurück
       * @param _serverUrl
       * @param _port
```

```
     * @param _userName
     * @param _pwd
     * @return ServerConnection
     * @throws RemoteException
     * @throws MalformedURLException
     * @throws NotBoundException
     * @throws ServiceException
     */
    private ServerConnection getConnectionToServer(String _serverUrl, String _port, String _userName, String _pwd)
                    throws RemoteException, MalformedURLException, NotBoundException, ServiceException {
        ServerConnection con = new ServerConnection(_serverUrl + ":" + _port, this.communicationType);
        con.connect(_userName, _pwd);
        return con;
    }
}
```

Abbildung 21: Klasse WebShopWS

b. Testclient

```
#!perl -w

use strict;

use SOAP::Lite;

#+++ IP ist zu Testzwecken auf einen Testrechner gesetzt +++#

#+++ SOAP::Lite holt sich mit der 'uri' die Web Service Adresse ++#

#+++ und mit 'service' die WSDL - Datei, damit die Web Service Methoden ++#

#+++ bekannt werden ++#

my $soap = SOAP::Lite

        ->uri('http://lfu.server.de/servlet/is/services/WebshopWS')

        ->service ('http://lfu.server.de/servlet/is/services/WebShopWS?wsdl');

# ++++++++++ Ruft getAllId() vom WebShopWS auf und gibt alle +++++++++++++++#

# ++++++ Id's vom WebShop auf www.lubw.baden-württemberg.de zurück ++++++++#

printf "Show all Id from Shop on www.lubw.baden-württemberg.de. Please wait.\n\n";

my @idList = $soap->getAllId("lubwShop");

foreach my $id (sort (@idList))

{

  printf "Id: %s ||",$id;

}

print "\n\n\n";

# ++++++++++++ Ruft getMetaDataFromId(String docId) vom WebShopWS ++++++++++#

# +++++++ auf und gibt alle Metadaten der speziellen Publikation zurück +++#

print "Get Metadata from ID: \n";

my $docId = <STDIN>;

chomp($docId);

print "Show now the Metadata from the Publication with the Id ".$docId." \n";

print "Get Infomation from ID. Please wait.\n";
```

```
#+++++ setzt eine Referenz vom Rückgabewert der Methode getMetaDataFromId.+#

#++++++++++++++++++++++ Gibt einen Hash zurück ++++++++++++++++++++++++++++#

my $hash = new $soap->getMetaDataFromId($docId,"lubwShop");

#+++++ listet alle Schlüsselwerte auf ++++#

#print "size of hash:  " . keys( %{$hash} ) . ".\n";

#my @keys = keys(%{$hash});

#foreach(@keys)

#{

#  print "$_ |";

#}

#+++++ gibt Schlüssel - Wert Paare auf Konsole aus ++++#

print "\n\n\n\n";

my $k;

my $j;

my $v;

#++++ Schleife geht alle Schlüssel - Wert Paare durch +++++#

foreach $k (sort(keys %{$hash}) )

{

  if ($hash->{$k}){

     #+ einige Rückgabewerte sind String Arrays und beginnen mit 'Hash'+#

     #+ dadurch muss auf zweites internes Schlüsselpaar zugegriffen werden +#

     if (!($hash->{$k} =~ /^HASH/)){

               printf "key: %s\nvalue: %s\n", $k,$hash->{$k};

               print "\n";

     }else{

               foreach $j ( sort(keys %{$hash->{$k}}) ) {

               printf "key: %s\n", $k;

               printf "value: %s\n\n", $hash->{$k}{$j};

               }

     }
```

```
#++++ Wenn ein Hashwert 'null' ist +++++#
  }else{
    printf "key: %s\nvalue: null\n", $k;
    print "\n";
  }
}
```

Abbildung 22: Perl Testclient

c. Inhalt der Zusatzdateien „server-config.wsdd“ und “lubws-hop_service.properties”

```
<service name="WebShopWS" provider="java:RPC" style="wrapped" use="literal">
<operation name="getMetaDataFromId" qname="ns1:getMetaDataFromId" returnQName="ns1:getMetaDataFromIdRe
turn" returnType="ns2:Map" soapAction="" xmlns:ns1="http://DefaultNamespace" xmlns:ns2="http://xml.apache.org/xml-soap">
            <parameter qname="ns1:docId" type="xsd:string" xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
</operation>
<operation name="getAllId" qname="ns3:getAllId" returnQName="ns3:getAllIdReturn" returnType="xsd:strin
g" soapAction="" xmlns:ns3="http://DefaultNamespace" xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
            <parameter name="allowedMethods" value="getAllId getMetaDataFromId"/>
            <parameter name="typeMappingVersion" value="1.2"/>
            <parameter name="wsdlPortType" value="WebShopWS"/>
            <parameter name="className" value="WebShopWS"/>
            <parameter name="wsdlServicePort" value="WebShopWS"/>
            <parameter name="schemaQualified" value="http://xml.apache.org/xml-soap,http://DefaultNamespace"/>
            <parameter name="wsdlTargetNamespace" value="http://DefaultNamespace"/>
            <parameter name="wsdlServiceElement" value="WebShopWSService"/>
</service>
```

Abbildung 23: Anpassung an server-config.wsdd

```
#Properties-Datei für den WebShop Web Service

#!!!!!+++++++++WICHTIG+++++++++!!!!!!!#

# Wenn ein Schlüssel mehr als einen Wert hat, bitte ein ";" zwischen den Werten schreiben

#++++++++++ConnectionProperties+++++++++#

# Das "$" Zeichen ist für das automatische Finden der Themenassoziationen nötig

$user=lesen-shop

$pwd=lesen-shop

$serverUrl=www.lubw.baden-wuerttemberg.de

$serverPort=80

# RMI(1)-> Port:1099 ist der Defaultwert von WebGenesis, SOAP(0)-> Port:80

$communicationType=0

#++++++++++ShopProperties+++++++++++++++#

#Id des WebShops in www.lubw.baden-wuerttemberg.de

$shopId=6638

# die Kategorie 19 entspricht in WebGenesis den Publikationen.

$entryCategory=19

#++++++++++Themenassoziation++++++++++++++#

#WebShop Thema=PortalU Thema

Abfall=Abfall

Altlasten=Altlasten

Boden=Boden

Chemikalien_und_Arbeitsschutz=Chemikalien;Gesundheit

Lärm=Lärm und Erschütterungen

Klima=Luft und Klima

Luft=Luft und Klima

    Luft_-_Emissionskataster=Luft und Klima

    Luft_-_Luftqualität=Luft und Klima

    Luft_-_Luftreinhalteplanung=Luft und Klima

    Luft_-_Verkehr=Luft und Klima
```

Natur_und_Landschaft=Natur und Landschaft

Natur_und_Landschaft_-_Artenschutz=Natur und Landschaft

Natur_und_Landschaft_-_Flächenschutz=Natur und Landschaft

Natur_und_Landschaft_-_Natura_2000=Natur und Landschaft

Natur_und_Landschaft_-_Eingriffsregelung_/_Landschaftsplanung=Natur und Landschaft

Natur_und_Landschaft_-_Naturschutz-Info=Natur und Landschaft

Natur_und_Landschaft_-_Rechtliche_Grundlagen=Natur und Landschaft

Natur_und_Landschaft_-_Verlagsveröffentlichungen=Natur und Landschaft

Radioaktivität=Strahlung

Elektromagnetische_Felder=Strahlung

Allgemeine_Umweltfragen=Umweltinformationen

Betrieblicher_Umweltschutz=Umweltwirtschaft

Betrieblicher_Umweltschutz_-_Nachhaltiges_Wirtschaften=Umweltwirtschaft

Betrieblicher_Umweltschutz_-_Stoffstrommanagement=Umweltwirtschaft

Betrieblicher_Umweltschutz_-_Technische_Luftreinhaltung=Umweltwirtschaft

Betrieblicher_Umweltschutz_-_weitere_Publikationen=Umweltwirtschaft

Wasser=Wasser

Wasser_-_Seen=Wasser

Wasser_-_Fließgewässer=Wasser

Wasser_-_Grundwasser=Wasser

Wasser_-_Abwasser=Wasser

Wasser_-_Wasserhaushalt=Wasser

Wasser_-_Institut_für_Seenforschung=Wasser

Lokale_Agenda_21=Verkehr;Umweltwirtschaft;Luft und Klima;Gesundheit;Energie

Lokale_Agenda_21_-_Leitfäden=Verkehr;Umweltwirtschaft;Luft und Klima;Gesundheit;Energie

Lokale_Agenda_21_-_Aktionsbörse=Verkehr;Umweltwirtschaft;Luft und Klima;Gesundheit;Energie

Lokale_Agenda_21_-_Arbeitsmaterialien=Verkehr;Umweltwirtschaft;Luft und Klima;Gesundheit;Energie

Lokale_Agenda_21_-_weitere Publikationen=Verkehr;Umweltwirtschaft;Luft und Klima;Gesundheit;Energie

```
#+++++++++Fachobjektbezug+++++++++++++++#

# ist optional; zu diesen Zeitpunkt Stand 05.07.2006 gibt es noch keine Fachobjektbezüge im WebShop

$Fachobjektbezug:

#+++++++++funktionale Klassifikation+++++++++++++++#

# zu diesem Zeitpunkt Stand 05.07.2006 gibt es noch keine funktionalen Klassifikationen im WebShop

$Funktionale_Klassifikation:
```

Abbildung 24: Properties-Datei – lubwshop_service.properties

d. WebGenesis-Befehle

Get - Informationen eines Eintrags lesen

Aufruf	REMOTE		
Zugriffsrechte	**READ**		
Parameter	*keine*		
Rückgabewerte	String	**name**	Name des Eintrags
	String	**title**	Titel des Eintrags
	String	**comment**	Die Kurzbeschreibung des Eintrags
	String	**Abstract**	Abstract des Eintrags
	Integer	**category**	ID der Eintragskategorie
	Integer	**parent**	ID des Parenteintrags
	int []	**children**	Array mit den Ids der Untereinträge des Eintrags
	Integer	**owner**	ID des Besitzers des Eintrags
	Integer	**creator**	ID des Erstellers des Eintrags
	String	**creation** (ab 7.00)	Erstellungsdatum in ISO-Format: yyyy-MM-dd'T'HH:mm:ss.SS
	Integer	**modifier**	ID des letzten Bearbeiters des Eintrags
	String	**modification** (ab 7.00)	Änderungsdatum in ISO-Format: yyyy-MM-dd'T'HH:mm:ss.SS
	Integer	**layout**	ID des Layouts des Eintrags
	String	**displayPage**	Alternative Darstellungsdatei
	String	**extraDescriptors**	Zusätzliche Schlagworte des Eintrags
	int []	**allDescriptors**	Ids aller zugewiesenen Schlagworte
	Integer	**int1**	Id des Zieleintrages wenn es sich um einen Querverweis handelt
	Integer	**seqNo**	Sortierindex des Eintrags im übergeordneten Eintrag
	String	**childOrder**	Reihenfolge der Attribute nach denen die Untereinträge sortiert werden.
Beschreibung	Liefert die Attribute eines Eintrags. Wird ein Attribut nicht zurückgegeben, so ist es nicht gesetzt (null). **Hinweis:** Versionen vor 6.34 liefern nur die Attribute *name* und *title*.		

Abbildung 25: Befehl zum Objekttyp Entry – get

Search - Nach Einträgen suchen

Aufruf	HTTP / REMOTE		
Zugriffsrechte	*keine*		
Parameter	String	**terms** (o)	Begriffe, die im Eintrag enthalten sein müssen, getrennt durch Komma oder Leerzeichen. Wird ein Leerstring angegeben, wird dieser Parameter ignoriert.
	String	**fields** (o)	Attribute, in denen nach den Begriffen gesucht wird, getrennt durch Komma oder Leerzeichen. Mögliche Attribute sind *descriptors*, *name*, *title*, *commentC* und *abstract*. Wird dieser Parameter nicht angegeben, wird in allen (Standard-) Feldern gesucht.
	String	**categories** (o)	Ids der Kategorien, denen die Einträge angehören dürfen, getrennt durch Komma.
	Integer	**owner** (o) (ab 6.35)	Id des Benutzers, dem der Eintrag gehört.
	Integer	**creator** (o)	Id des Benutzers, der den Eintrag erstellt hat.
	Integer	**modifier** (o) (ab 6.35)	Id des Benutzers, der den Eintrag geändert hat.
	String	**from** (o) (ab 6.50)	Änderungsdatum muss größer (oder gleich) als dieser Wert sein. Ab Version 7.00 wird als Format nicht mehr DateAccessor.DATE sondern DateAccessor.ISO (yyyy-MM-dd'T'HH:mm:ss.SS) erwartet.

	String	**till** (o) (ab 6.50)	Änderungsdatum muss kleiner (oder gleich) als dieser Wert sein. Ab Version 7.00 wird als Format nicht mehr DateAccessor.DATE sondern DateAccessor.ISO (yyyy-MM-dd'T'HH:mm:ss.SS) erwartet.
	Integer	**subTree**	Id eines Eintrags. Als Ergebnis werden nur solche Einträge geliefert die unterhalb dieses Eintrags liegen. *Hinweis*: Dieser Parameter wirkt als Filter, damit man ein Ergebnis erhält muss mindestens bei *terms, categories* oder *creator* ein Wert angegeben werden.
	Boolean	**useSubTree** (o) (ab 6.60)	True um den subTree Parameter zu verwenden.
	String	**orderBy** (o)	Attribute, nach denen sortiert wird. Beispiel: „name desc, modification"
	String	**template** (ab 6.34)	Name eines Templates, dass anstatt des Standardtemplates „Search.Detailed" bzw. „Search.Results" verwendet werden soll.
	Boolean	**doIt** (o) (HTTP)	Führt die Suche aus bei *true*, ansonsten wird die Suchmaske angezeigt Bei **REMOTE** immer true.
	Integer	**permissions** (o) (ab 7.10)	Filtered die Einträge nach der angegebenen Rechte Kombination. Es wird immer mindestens nach Sehenrechten (Grant.SEE) gefiltert.

Rückgabewerte	**HTTP:** List [Entry] **REMOTE:** int []	**resultsSE**	Gefundene Einträge
	Integer	**count**	Anzahl der gefundenen Einträge
	Grantee	**user** (HTTP)	Der angemeldete Benutzer
	String	**dateFormat**	Das verwendete Datumsformat.
	Search	**criterion** (HTTP)	Suchobjekt mit Suchparametern
Beschreibung	Sucht nach Einträgen. Beim Aufruf über HTTP werden vollständige Entry-Objekte geliefert, bei REMOTE nur die IDs der gefundenen Einträge.		

Abbildung 26: Befehl zum Objekttype Entry – search

GetContentFileInfos – Dateiinformationen lesen

Aufruf	REMOTE		
Zugriffsrechte	**READ**		
Parameter	*keine*		
Rückgabewerte	String []	**name**	Namen der Dateien
	int []	**size**	Größe der Dateien in Byte
	String []	**formatted-Size**	Größe der Dateien formatiert als String. Je nach Größe der Datei wird gerundet und die Endung „Bytes", „KB" oder „MB" angefügt.
	String []	**modification**	Änderungsdatum der Dateien. Formatiert als ISO-Datum.
Beschreibung	Liest Informationen über die Dateien im Inhaltsbereich eines Eintrags.		

Abbildung 27: Befehl zum Objekttype Entry – getContentFileInfos