Universities Nice – Mannheim

Teleseminar on eCommerce

Sept 2001 – Feb 2002

Documentation of the

CDNOW Project

Advisors :

Michel Buffa
Claudia Schremmer

Participants :

Mannheim : Jörn Bachmeier
Aadrian Geursen
Thomas Wecker

Nice : Nicolas Falorni
Alexander Mladin
Sébastien Salazar

Summary

1. Introduction

The Universities of Mannheim and Nice in cooperation with accenture have given this year for the first time life to cooperation on eCommerce seminar of the duration of one semester. It began in September 2001 with a starting week in Nice and ended up with a concluding seminar in February 2002 in Mannheim. The time in between was foreseen for the work on the projects. Six different projects were proposed in September to the 24 participants. They had to build groups and choose one of the project themes. Our team decided to implement the cdnow project. The project consists of the implementation of a web page for the selling of music content over the web similar to the existing cdnow music portal.
This document describes the way our group worked on the project. First of all a short description of the project is given then the desired functionalities of the software are listed. After that the two groups – German and French – show how the implemented themselves the application. In the end the experiences on the collateral work between the two groups are sketched.

2. What was the Project about?

The aim of the project is to develop a site for selling music content on the web. This web site can be used by customers to search a CD database using different criteria (name of the artist, song, style of music, etc...) order some CDs or check their passed orders. Furthermore, customers can order directly digital content for download (mp3, wma or .waw files). The database hosts multimedia content (cd covers, digital music) as well as regular data concerning user and music information.

Customer profiles are updated so that accurate topics are displayed each time he connects to the site. When a customer finds some CDs after a search or an order, the site proposes what other customers that bought the same product liked.

An administrator interface is implemented as a secure web page for adding/ deleting/ modifying content on the database, as well as extracting information about the customer's profile, etc...

Used software technologies:

- XML + XSLT
- Java + JSP/Servlets
- DB Server connected through JDBC
- MySql Database
- Resin/caucho software
- Connectivity to FreeDB with PHP

3. How did we try to achieve the aims?

During the beginning seminar week in Nice our group consisting of 6 members 3from Mannheim and 3 from Nice had the first meeting. Then we partitioned the work into two main parts: at the one hand the web interface implementation and at the other the creation and management of the database behind the web application. The Mannheim group took the first task the Nice group cares about the second.

4. Functionalities

4.1 Features

4.1.1 The Internet site

Consultation of CD

The system offers the possibility to the customers to consult and to buy a CD among the list of products on sale or on promotion, helps columns concerning every music style.

Display by style

The system gives the possibility to the customer of research by choosing among the numerous different styles: "Blues", " Country ", "Jazz", " Rap ", "Soul", " Hip Hop ", " RnB ",… For each of the styles, the system will suggest an alphabetical research. Then user will have to select a letter, the system will show all the concerned artists. The user will have to choose the artist wished to obtain the list of all the albums of this one.

Display a list of albums

The system shows a list of albums concerning an artist, consisted of the reference, the title of the album, the jacket, the price TTC and a specifying wording if the album is available in listening.

Description of an album

A description of every album will be shown in zone 3. This zone will present the contents of the abum. This description will show when the user clicks on the title of the album. Every title of the abum will appear in the same way: the title, the authors, the duration then if the track is available in listening (if it is the case, one icon @@ will be in end of line).

4.1.2 Consultation of reception

Description of the GUI

The system shows the data by zones: the system shares the central window (or zone 8) in 4 zones: Left superior half is the zone 1 and the right-hand side is the zone 2, also left lower half is the zone 3 while that of the right-hand side is the zone 4; the menu which allows in an user to navigate, to change columns is in zone 5; a menu bar is high top of it zone8 allowing in an user of reach directly in column of the catalog, 61623; the bar at the top of the page is the zone 6: the system proposes there of make a search(research) and to the customer already registered to seize his login and its password, 61623; at the bottom of the page is the zone 7 in which the system proposes some link.

Resolutions

The system gives two possible resolutions: 800*600 and 1024*768 allowing so in the user to benefit of one better returned with regard to the configuration.

Contact

The system allows in user to contact the CdNow company by means of this option. The system shows a list of CdNow's various services with their corresponding e-mail.

Sending an e-mail in a service CdNow

The system suggests in user of sending an e-mail to a service of the company Cdnow by means of the links being in the page of contact. The system shows then a compound form of seizure of a zone of seizure allowing in an Internet user to seize its message, of a field "Name" and "First name", of a zone of text giving the possibility of seizing an object of the message then the text of an e-mail and of a button "Send" which sends the mail of an user in the wished service. Once the mail is sent, the system returns automatically on the page of the contacts.

Terms of sale

The system allows in an Internet user to acquaint with terms of sale of the company CdNow by means of the link "Terms of sale", the system shows then in the page the terms of sale.

Payment

The system allows in an Internet user to acquaint with the payment of the company Cdnow by means of the link "Payment", the system shows then in the page the payment.

Research

The system suggests in an Internet user in zone 6 of making a research that it is an artist or a title of album according to the choice of an user who will have select an option beforehand. The system will show in zone 8 the list resulting from the research made by an Internet user in an alphabetical order.

4.1.3 Consultation of the catalogue

Research of a definition by keyword

The system allows to the customer to seize a keyword on which will be made a research in the data base. All the appropriate information will be shown under the shape of list in which an Internet user can navigate with the buttons "Forward" and "Backward".

Research by alphabetical classification

The system allows the knowing customer an exact spelling of the word to do the research by selecting the first letter of the word. So the system will send back in result a list of all the words beginning with the selected letter.

Posting of the results

All the appropriate information will be shown under the shape of list in which an Internet user can navigate with the buttons "Forward" and "Backward".

4.1.4 Consultation and management of the supermarket trolley

The supermarket trolley appears in the form of an allowing table in an user to show all these purchases. The table decomposes as follows: Reference of the product, Name of the product, Quantity of the product, Price of the product TTC in Euros. If the customer makes modifications at the level of the quantities, he has the possibility of proceeding to a recalcul of the prices by means of a button "Modify" being at the bottom of the page of supermarket trolley.

Modification of the supermarket trolley

The system allows to the customer to modify the quantity of a product in its supermarket trolley: the customer can then increase or decrease the quantity of the selected product.

Deletion of a product

Having selected a product, the system gives the possibility to the customer to delete it simply by clicking the neighboring button "Delete".

Management of the redundancy

If the customer selects a product already contained in the supermarket trolley, the system shows in an screen the message an informing about the situation. The system proposes then to the customer of to cancel its second purchase and a redundancy in which case avoids so one would have found twice the same product in the supermarket trolley. But the system allows also to the customer to continue its purchase with in which case the quantity of this product again selected would be added to the already registered quantity.

Validation of the command

Registration of a customer during the first command

The system asks to the customer to supply some information concerning him (certain fields being compulsory marked of an asterisk and of other optional):
Last name and first name*
Birthday*
Address (Address, CP and City) *

E-mail*
Password*
Phone number 61623
Number of fax, number of mobile
Profession

Already registered customer

The system asks to the customer to seize identifying sound and its password allowing him of to reach its account. If the customer makes an error at the time of the seizure of his login or his password, the system will show then a message of alert. The system proposes to the customer having forgotten its login or password a button "Forget". The system will show then a message asking to the customer to seize its name, its first name and its place of birth. So the customer will reach his account thanks to the system which an aura recently identified.

Payment conditions

The system proposes the only one mode of payment to the customer: The payment by check card: the customer should supply the number of his check card and his date of expiration. This operation would be treated by IBS thanks to the software PAYBOX.

Place the order

The customer confirms then his command by selecting the button " To validate the command" .A message of confirmation will show then in an screen.

4.1.5 Consultation and management of an space out member

Consultation of the account

Consultation of the current command

The system proposes to the customer a state of promotion of its command so it will be informed as soon as possible of an arrival of its material or of a possible delay.

Consultation of the previous commands

The system will show the previous in order chronological list of the commands (from the most recent to the most ancient). Further to the selection of a command, a system will show the contents, the total amount of this one.

Statistics of purchases

The system will show the list of all the products bought by the customer. The system proposes to the customer to show a list according to the category of the product, so the customer can consult the list of all the accessories that he bought since he is a customer at Cdnow.

Modifications

Deletion of an space out member

The system proposes to the customer does not who want any more to deal with Cdnow to delete its account and the system erases then all the data concerning him.

Changes of the information (address, phones)

The system gives access to the customer in the certain information as he should supply at the time of his registration. So he can then modify this information for example address, telephone or password.

Mail

The system allows in an user to register to a newsletter to receive mails of information (on the last novelties, the promotions and more generally the information). Addition to the mail.

Addition on the List

The system allows the customer to add to the list of information by marking the corresponding compartment.

Deletion of the List

The system allows to the customer not to receive any more the mail of information.

5. How was the work distributed?

French Side

MySql

Database generalities :

The project uses a MySql database to store the user and music information. MySql is free database software downloadable from the internet. The MySql software is well designed for being used over the internet: besides from being accessible over the internet, it provides a web-based interface for administration.

The administration tool can be a PHP web interface. We use “PHPMyAdmin”. This is also free software downloadable from the internet. It supports login and password features as well as a full database configuration so it can be used as a complete administrative tool for the application database. With PHPMyAdmin the database can be adjusted manually. Tables and attributes can be added and deleted from the database. PHPMyAdmin is also a very helpful tool for the testing phase of the software.

The actual version of MySql allows all common SQL operations and requests so it can be used without reservation for our project.
The connection to the database is made by a Java application. The several implemented classes are described on the next pages.

Database :

The database is composed of four tables :

- **CUSTOMER :** contains the information concerning the users of the site.
- **ORDER :** contains the information concerning the orders (data on the customers and the products).
- **ALBUM :** contains the information concerning the albums of the various artists (updated with CDDB data).
- **TRACK :** contains the information about the samples of the CDs tracks.

CUSTOMER
In this table the information stored concerns the user. Besides the personal information of the customer there is also stored the preferred genre of the customer who allows to personalize the user account.

ID	bigint(10)	PRIMARY KEY
FirstName	varchar(30)	
LastName	varchar(30)	
Street	varchar(50)	
PostalCode	varchar(5)	
Town	varchar(30)	
Country	varchar(30)	
Phone1	varchar(20)	
Phone2	varchar(20)	
Fax	varchar(20)	
Email	varchar(30)	
PreferedGenre	varchar(30)	
Password	varchar(10)	

ORDERS
In this table there are stored all the commands passed by all users. For every ordered Article a new line will be created. Every order contains the ID of the customer so it can be at any time linked to a customer.

ID	bigint(10)	PRIMARY KEY
IDCustomer	varchar(10)	customer who passed the command
IDAlbum	varchar(10)	ordered Album
Date	date	'yyyy-mm-dd'
Type	varchar(20)	(CD or Download)

ALBUM
In this table there is stored the information about all the music offered by the site. This table is designed so that it can store all the information from the FreeDB format. The Administrator has the possibility to update this table. Using a PHP-script he can grab music information from a FreeDB-Server and write it in the ALBUM table. FreeDB is detailed later. It holds all the track names for each albums and so on…

The additional fields in this table are used for the economical management of the music. Prices are stored as well as the availability on stock.
Finally links are stored to the covers of the offered album.

ID	bigint(10)	PRIMARY KEY
FreedbID	varchar(10)	
Interpret	varchar(30)	
Title	varchar(30)	
Year	year	
Album	varchar(30)	
Single	boolean	LP or Single Album
Genre	varchar(30)	
CDQuantity	int(4)	quantity of CDs still available in the stock
CDPrice	float	prices can differ between cd and download
DownloadPrice	float	
CDEntrance	date	date of entrance in the selling list
Description	longtext	description of the Album; also possible links to critics or additional information
Lenght	time	full album length
TracksNb	int(2)	number of tracks in the Album
Picture	varchar(10)	link to cover picture
MP3	varchar(10)	download link

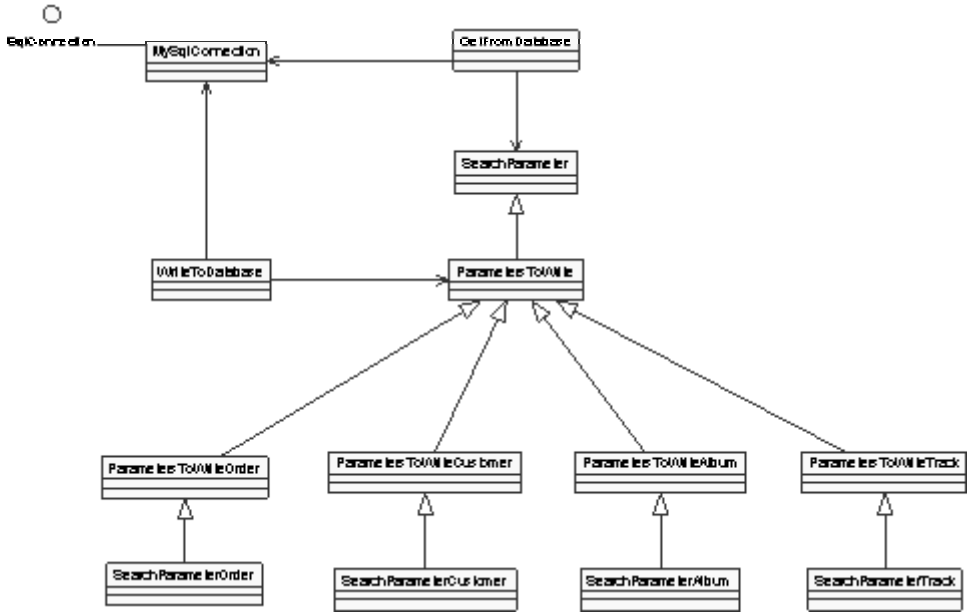
TRACK
In this table there are stored all tracks available

ID	bigint(10)	PRIMARY KEY
AlbumID	varchar(10)	
Title	varchar(30)	
Sample	varchar(30)	music sample that can be heard before buying

Access to the database :

The database access is made by a Java application. You can find the corresponding javadoc here.
The following class diagram represents the class dependencies of the classes described below:

The following classes are parameter classes. They are used as interfaces between the web interface and the database application for the transfer of data:



class diagram of the application classes

- **ParametersToWrite:**
  this class is the superclass of the ParametersToWrite… classes. These classes are used to get the information from the web interface and to write it into the database. An object of this class is handed over from the JSP servlet to the database application.

- **ParametersToWriteCustomer:**
  this class is inherited from the ParametersToWrite class. It contains the informations concerning a customer. These are the personal information (like name and address…) but also the prefered genre of the customer can be changed over this class. An instance of this class is created and handed over from the JSP servlet class when an entry is written to the customer table in the database.

- **ParametersToWriteOrder:**
  this class is inherited from the ParametersToWrite class. It contains the informations concerning an order. The customer ID is also included so that every order can be assigned to a customer. It also contains the information whether the order is a post order or a download. An instance of this class is created and handed over from the JSP servlet class when an entry is written to the orders table in the database.

- **ParametersToWriteAlbum:**
  this class is inherited from the ParametersToWrite class. It contains the informations concerning a music album. This is the identification information (like name, interpret…) but also the ordering information (price, availability in stack… can be changed. Finally the links for the download of the whole album or of the CD cover can be changed by this class. An instance of this class is created and handed over from the JSP servlet class when an entry is written to the album table in the database.

- **ParametersToWriteTrack:**
  this class is inherited from the ParametersToWrite class. It contains the informations concerning a single music track. The identification information (like ID and name) but also the download link for the whole track and for the small sample can be changed with this class. An instance of this class is created and handed over from the JSP servlet class when an entry is written to the album table in the database.

- **SearchParameter:**
  this is the superclass for all Parameter classes. These classes are used to get the query information from the web interface.

- **SearchParameterCustomer:**
  this class is inherited from the ParametersToWriteCustomer class. It contains the same fields like its superclass but this class is used to make the queries on customers. An instance of this class is created and handed over from the JSP servlet when a customer query is passed.

- **SearchParameterOrder:**
  this class is inherited from the ParametersToWriteOrder class. It contains the same fields like its superclass but this class is used to make the queries on orders. An instance of this class is created and handed over from the JSP servlet when an order query is passed.

- **SearchParameterAlbum :**
  this class is inherited from the ParametersToWriteAlbum class. It contains the same fields like its superclass but this class is used to make the queries on albums. An instance of this class is created and handed over from the JSP servlet when an album query is passed.

- **SearchParameterTrack:**
  this class is inherited from the ParametersToWriteTrack class. It contains the same fields like its superclass but this class is used to make the queries on tracks. An instance of this class is created and handed over from the JSP servlet when a track query is passed.

The following classes are used for the establishment of connections, the reading and writing/updating to the database. For the connection to the databases the com.caucho.jdbc.mysql.* package containing all the classes necessary for communication with MySQL databases have been imported.

- ***SqlConnection:*** <interface>
  this interface is used to model classes who will connect to MySql databases. It contains the methods open() and close() who shall be implemented.

- **MySqlConnection:**
  this class implements SqlConnection. It implements the open() method for the connection to the database using the JDBC for MySql driver from caucho. Therefore the connection URL, a login name and a password are required.

```
    // Load the jdbc driver for Mysql
Class.forName("com.caucho.jdbc.mysql.Driver");
// Get a connection to the database source using JDBC driver
String url = "jdbc:mysql-caucho://localhost/music";
//connection=DriverManager.getConnection(url);
connection = DriverManager.getConnection(url,"music","wolfgang");
```

- **GetFromDatabase:**
  used to make queries in the tables of the database and to return the results. The class gets a SearchParameter object. The main method in this class is getResult(). It returns a 2 dimensional Array of SearchParameter objects filled with the results found by the query. Depending on what SearchParameter was given to the GetFromDatabase class, the class transforms the SearchParameter into a SQL query. After that the method tries to establish a connection to the database using a MySqlConnection object. Depending on what SearchParameter was given the return Array is filled in one or two rows. For a track query in the first row there is the found SearchParameterTrack object, in the second there is the SearchParameterAlbum object corresponding to the track found. For the three other tables the results are stored in the first row of the array.
  The returned object Array can then be used from the JSP Servlet to extract the information needed to display.

  Example:
  Set the SearchParameter and call the GetFromDatabase class:

```
((SearchParameterAlbum)s).description="best of 2002";
((SearchParameterAlbum)s).interpret="dion";

    GetFromDatabase gfdb=new GetFromDatabase(s);
    SearchParameter[][] sp = gfdb.getResult();
```

  Read out the result objects:

```
for(int i=0;i<gfdb.i;i++){
//System.out.print(((SearchParameterAlbum)sp[0][i]).ID);
//System.out.println(((SearchParameterAlbum)sp[0][i]).interpret);
```

  Get the values:

```
//search is a SearchParameter created in the constructor. It is the //object that is handed over to the GetFromDatabase class
if (search instanceof SearchParameterAlbum){
// in this object we will store all the rows corresponding to the query
result=new SearchParameter[2][];

// get the information for the query. All the requested word are stored in the SearchParameter
ID=((SearchParameterAlbum)search).ID;
freedbID=((SearchParameterAlbum)search).freedbID;
interpret=((SearchParameterAlbum)search).interpret;
    .
```

```
        .
        .
 description=((SearchParameterAlbum)search).description;
 .
        .
        .

 // make the query with the key words

 // initial query
query="SELECT * FROM ALBUM WHERE ID2='0'";
//all the tables have the ID2 = 0, it is used for the simplification //of the making of the queries

 // if the key word is null for one parameter then "OR PARAMETER=" don't appear ine the query
 if (ID!=""){
   query+=" AND ID="'+ID+'"";
 }
 if (freedbID!=""){
  query+=" AND UCASE(FREEDBID) LIKE UCASE('"+freedbID+"')";
 }
 if (interpret!=""){
   query+=" AND (UCASE(TRIM(INTERPRET)) LIKE UCASE(TRIM('%"+interpret+"%'))";
   query+=" OR SOUNDEX(INTERPRET) LIKE SOUNDEX('"+interpret+'"))";
 }
        .
        .
        .
 if (description!=""){
   query+=" AND UCASE(DESCRIPTION) LIKE UCASE('%"+description+"%')";
 }
        .
        .
        .
 query+=" LIMIT "+start_row+","+number_of_rows;

        // create the connection corresponding to the query using a MySqlConnection
        MySqlConnection albumConnection=new MySqlConnection(query);
        try{
        // open the MySql Connection
        albumConnection.open();
        // get the result set of the query
        ResultSet rs = albumConnection.getResultSet();
        // use a count to insert elements in the table
        i=0;
        // result is a table of SearchParameterAlbum
        result[0]=new SearchParameterAlbum[30];
        //put the results in an array of SearchParameter. this parameter are set with the information of
        the stored rows
   while(rs.next()){
       result[0][i]=new SearchParameterAlbum();
       // we store all the parameters
       ((SearchParameterAlbum)result[0][i]).ID=rs.getString("ID");
        ((SearchParameterAlbum)result[0][i]).freedbID=rs.getString("FREEDBID");
        ((SearchParameterAlbum)result[0][i]).interpret=rs.getString("INTERPRET");
        .
        .
        .
        ((SearchParameterAlbum)result[0][i]).description=rs.getString("DESCRIPTION");
        .
        .
        .
```

```
        i++;
        }
     // close the resultSet and the MySqlConnection
     rs.close();
     albumConnection.close();
   }catch(Exception e){…}
 }
```

- **WriteToDatabase:**
  used to write and to update the tables of the database. The constructor of this class gets two parameters: the first is a ParameterToWrite object who contains the information from the web interface, the second is a function parameter who determines what to do with the data from the first parameter. Three possibilities are accepted:

  - **INSERT**: if this function is chosen, the ParametersToWrite object is added to the respective table. It gets a new ID Number and it is added at the end of the table as a new entry. Therefore the write() method is called. The method reads the ParametersToWrite object out and transforms it into a SQL “INSERT INTO TABLE…” operation.
  - **UPDATE**: if this function is chosen, the line in the corresponding table of the database with the same ID like the ID of the ParametersToWrite object is going to be updated. Therefore the update() method is called. The method reads the ParametersToWrite object and transforms it into a SQL “UPDATE…” operation.
  - **DELETE**: if this function is chosen, the line in corresponding table of the database will be deleted. Therefore the delete() method is called and the line corresponding to the ID of the ParametersToWrite object is deleted by a SQL “DELETE FROM TABLE…” operation.

  For the simplification of the usage of the WriteToDatabase class there have been introduced 3 final attributes: INSERT, UPDATE, DELETE. So the class can be called by handing over a ParanetersToWrite object and one of the 3 keywords.

  Example:
  Set the ParametersToWrite (customer):

```
ParametersToWrite p;
((ParametersToWriteCustomer)p).firstName="BILL";
((ParametersToWriteCustomer)p).lastName="GATES";
((ParametersToWriteCustomer)p).town="NEW YORK";
((ParametersToWriteCustomer)p).password="microsoft";
```

  inserting the line:

```
WriteToDatabase wtdb = new WriteToDatabase(p,WriteToDatabase.INSERT);
```

  Depending on what ParametersToWrite object is handed over the WriteToDatabase class establishes a connection to the database using a MySqlConnection object and then performs one of the three actions above:

```
customerConnection=new MySqlConnection(query);
        try{
        //open the connection
      customerConnection.open();
      //write the row
      customerConnection.writeToDB();
      //close the connection
      customerConnection.close();
}catch(Exception e){…}
```

Example for the passing of an request:

- The user fills the HTML form with query keywords.
- The JSP servlet of the webpage generates a **SeachParameter** object depending on the object that is searched (Customer, Order, Album or Track). The **GetFromDatabase** class is called from the servlet and the **SearchParameter** is handed over.
- Then the **GetFromDatabase.getResult()** method connects to the database over the **MySqlConnection** class. It passes the « select – from – where » query and returns the result in a 2 dimensional Array.
- After that the Array is parsed by the JSP Servlet and displayed in the web page.

Freedb

What is FreeDB ?

Freedb (free cd database) is an information database containing artist, disc title, track titles, and other information for digital audio compact discs. It is based on the CDDB server software, which was created originally to support Xmcd, a CD-audio player software package for many computer platforms running the X11 window system. It uses the OSF/Motif toolkit for its graphical user interface. The database archives are periodically updated, incorporating new user submissions.

Cddb/Freedb ID :

Freedb access requires that the software computes a « disc ID » which is an identifier that is used to access the freedb. The disc ID is a 8-digit hexadecimal number.

Each CD entry is a separate file. The files are organized in several directories, each directories is a category of music. Currently the categories are listed as follows : blues, classical, country, folk, jazz, newage, reggae, rock, soundtrack and misc. The individual database files have a file name that is the 8-digit disc ID. For example, under the blues directory there may be the following files : 0511c012, 060e7314, …, fb0f8814, fd0e6013.

Freedb file format :

The beginning of the first line in a database entry should consist of the string « #xmcd ». This string identifies the file as an xmcd format CD database file. All the lines that begin with by # are comments. The comments should also contain the string « #Track frame offsets » followed by the List of track offsets. After the offset list, the following strinf should appear : « # Disc length : xxx seconds ». Following the comments is the disc data. Each line of disc data consists of the format « KEYWORD=data ».

Valid keywords are as follows :

- DISCID
- DTITLE : the artist and the disc title are separated by a « / »
- DYEAR : 4-digit year in which the CD was released
- DGENRE : exact genre of the disc in a textual form
- TTITLExxx : the track number should be substituted for the « xxx » starting with 0. This field contains the title of the xxxth track of the CD.
- EXTD : this field contains the extended dta for the CD
- EXTTxxx : this field contains the extended track data for the xxxth track.
- PLAYORDER : this field contains a comma-separated List of track numbers which represent a programmed track play order

Example for a FreeDB formatted data file:

```
#xmcd
#
#Track frame offsets :
#       150
….
#       210627
#
#Disc length : 2952 seconds
#
DISCID=270b8617
DTITLE=Franske Stemninger / Con Spirito
DYEAR=1981
DGENRE=Classical
TTITLE0= Mille regrets de vous abandonner
….
TTITLE22=L’arche de Noe
EXTD=Copyright © 1981 MCA Records Inc.\nManufactured
EXTD=for MCA Records Inc.
EXTT0=Des Prez \nYez
….
EXTT22=Schmitt : A contre-voix
PLAYORDER=
```

German Side

JSP & Servlets

Java Server Pages (JSP) are similar to HTML-documents in structure. But they consist of two parts: one HTML-text and as many ingrained instructions to the JSP-server as you like. To insert JSP-instructions, there are used special tags, which are similar to the well known HTML-tags. HTML-tags keep their function to format the displayed document. JSP-tags have in contrast to HTML-tags not always effect on the displayed document. They are not sent to client, but server side actions can have influence to the output. Most of these actions are used to implement the application logic. Mainly access authorization or database transactions.

Server side applications offer dynamic pages adjusted to user's needs instead of static pages. HTML provides the functionality to communicate with the user. Inquiries can be done by HTML-forms. The used <form>-tag applies as destination a JSP-application. This application can take over the data enclosed in the form, process the data and generate a suitable reply.

So, by using JSP-pages it is possible to separate surface components from the program. As a result specialists can be applied to develop the programming part and the web design independently. Because of this strict separation it is easy to maintain the program logic. By using JAVA as programming language the code is also portable. In fact every JSP-page is finally compiled into a Servlet by the JSP engine. Compiled Servlet is used by the server to serve the user requests.

There is also the opportunity to use finished components, so-called Beans. These beans are small program components which are re-useable. They can be easily integrated in a web application. JSP represents the presentation layer. So in complex applications the reusability of the components is granted.

In our project CDNow, we only use one JSP-page:

```
<jsp:useBean id="pg" class="cdnow.Core"/>
<jsp:useBean id="param" class="cdnow.Parameter"/>
<jsp:setProperty name="param" property="*" />
<%
  if ((request.getParameter("login") != null)
        && request.getParameter("login").equalsIgnoreCase("YES")))
  {
    session.setAttribute("LOGIN", "YES");
  }

  pg.setSession(session);

  param.setSid(session.getId());

  Object tmp = session.getAttribute("LOGIN");
  if ((tmp != null) && (tmp instanceof String)
```

```
        && (((String)tmp).equalsIgnoreCase("YES")))
 {
   pg.setLoginState("YES");
 }

%>
<%=
        pg.generatePage(param)
%>
```

This page includes two beans:
Core and *Parameter*. The first bean *Core* is used to generate the displayed document as a function of the transferred user data. This data is enclosed in the second bean *Parameter*.

JSP offers the action *getProperty* to adopt the parameters of the URL's query-string. This functionality is used by the bean *Parameter*. *GetProperty* sets the property values in the *Parameter-Bean*.

The JSP-server automatically creates important objects, so-called implicit objects. In our JSP-page we outreach the implicit object *session* to the bean *Core*. The session object is used for session management. It shares information for one user across multiple pages while visiting the web site, for instance if the user is logged in or not.

As we have only one JSP-page, it is very easy to maintain the portal. Also testing gets very easy as all pages can be easily invoked by just typing the desired parameter.

XML & XSLT

What is XML?

XML is the abbreviation for e**X**tensible **M**ark-up **L**anguage. A mark-up language in general is used, as the name says, to mark-up data. It is a methodology for encoding data with some information. In so far **XML** has some parallels to **HTML** (**H**yper**T**ext **M**ark-up **L**anguage). But **XML** is even more than a typical mark-up language that defines a set of tags each of which has some associated meaning. **XML** gives the programmer the possibility to name the tags in a semantic useful way.

HTML vs. XML

In fact **HTML** is the most popular mark-up language at this time. The defined set of tags is used to present data. The strength of **HTML** is the very easy way of implementation because of its small numbers of tags, the very forgiving syntax-checking and the simple relationship of the tags.
But the advantage of being not so complex is also a disadvantage because it is not user extensible. In so far **HTML** is not suited for handling large and

complex data, data that must be used in different ways or data with long life-cycle.
Out of this, the motivation for a new mark-up language was born. The new language should be straightforwardly usable over the internet, support a wide variety of applications, compatible with **SGML** (**S**tandard **G**eneralized **M**ark-up **L**anguage; **SGML** is an international standard for the definition of device-independent, system-independent methods of representing texts in electronic form.), human readable, easy to implement etc...
While **HTML** is used to present data, **XML** is only used to structure it. An **XML** document has the structure of a tree with a root element and a theoretical infinite number of own tags that define the branches and leafs. According to this, an **XML** document has to be well-formed. That means that every tag, that is opened, has to be closed. Otherwise the file will not be represented. The representation related data is stored in an external **XSL**-file.

What is XSL?

XSL stands for e**X**tensible **S**tylesheet **L**anguage. XSL is a language for expressing stylesheets. It consists of three parts: XSL Transformations (XSLT): a language for transforming XML documents, the XML Path Language (XPath), an expression language used by XSLT to access or refer to parts of an XML document. (XPath is also used by the XML Linking specification). The third part is XSL Formatting Objects: an XML vocabulary for specifying formatting semantics. An XSL stylesheet specifies the presentation of a class of XML documents by describing how an instance of the class is transformed into an XML document that uses the formatting vocabulary.
The XSL-file is based on templates. A template describes what to do if a tag or an attribute in the XML-file matches the condition of the template.

A short example:

XML-File

```
…
<example style=”test”>
        <test>
                <…>
        </test>
</example>
...
```

XSL-File

```
…
<xsl:template match=”example”>
        do …
</xsl:template>
<xsl:template match=”@test”>
        do …
</xsl:template>
…
```

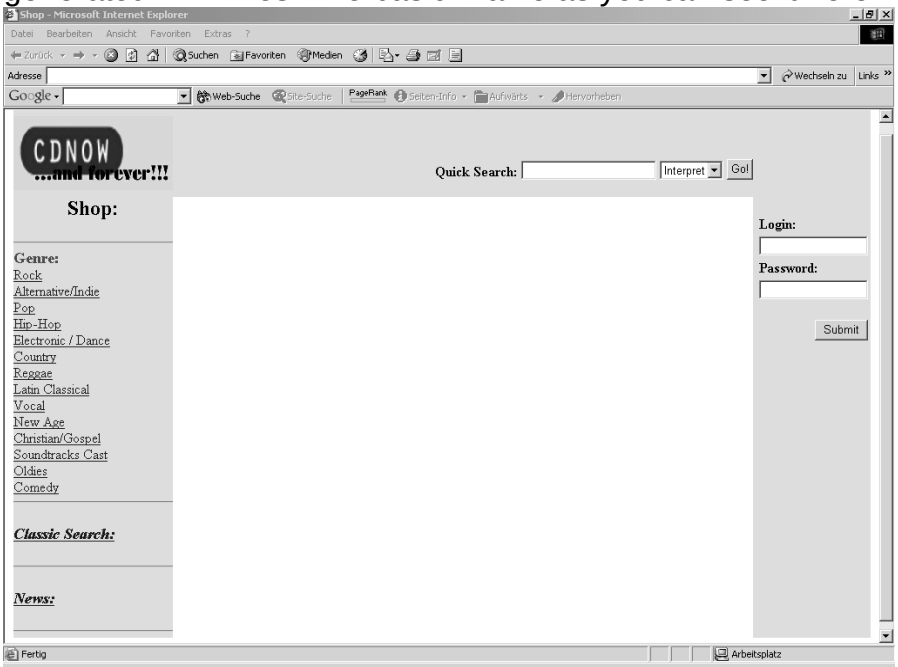
The parser (like Xalan, AxKit or Cocoon) associates each XML file with an XSL file and generates for example an HTML-file or a PDF. The association is realized by a line at the beginning of the XML-file like <?xml-stylesheet type="text/xsl" href="cd-now-stylesheet.xsl"?>.

Why and how we used XML

We decided to use XML, because of its positive features according to semantically structuring of data. In fact XML is kind of a database, but static and not dynamical. In order to appreciate XML, it is important to understand why it was created. XML was created so that richly structured documents could be used over the web. The only viable alternatives, HTML and SGML, are not practical for this purpose. HTML, as we've already discussed, comes bound with a set of semantics and does not provide arbitrary structure. SGML provides arbitrary structure, but is too difficult to implement just for a web browser. Full SGML systems solve large, complex problems that justify their expense. Viewing structured documents sent over the web rarely carries such justification.

So we use the XML-Technology to transfer the data from the database to an HTML file. The Parsing takes place at the serverside. If a search request is send from the client to the Servlet, the JSP connects the database, get the data and generates an XML-file. This is much easier than to create an HTML-file because there is no representation relating information needed. If modifications of the representation have to be done, the only file that has to be changed is the XSL-file.

Each XML-file has a root element. It does not matter how the tag is defined, it is just important that one exists. One big stylesheet is used that matches all generated XML-files. The basic Frame as you can see it here…



…is always generated. The data will be represented in the middle part of the table depending on the information.

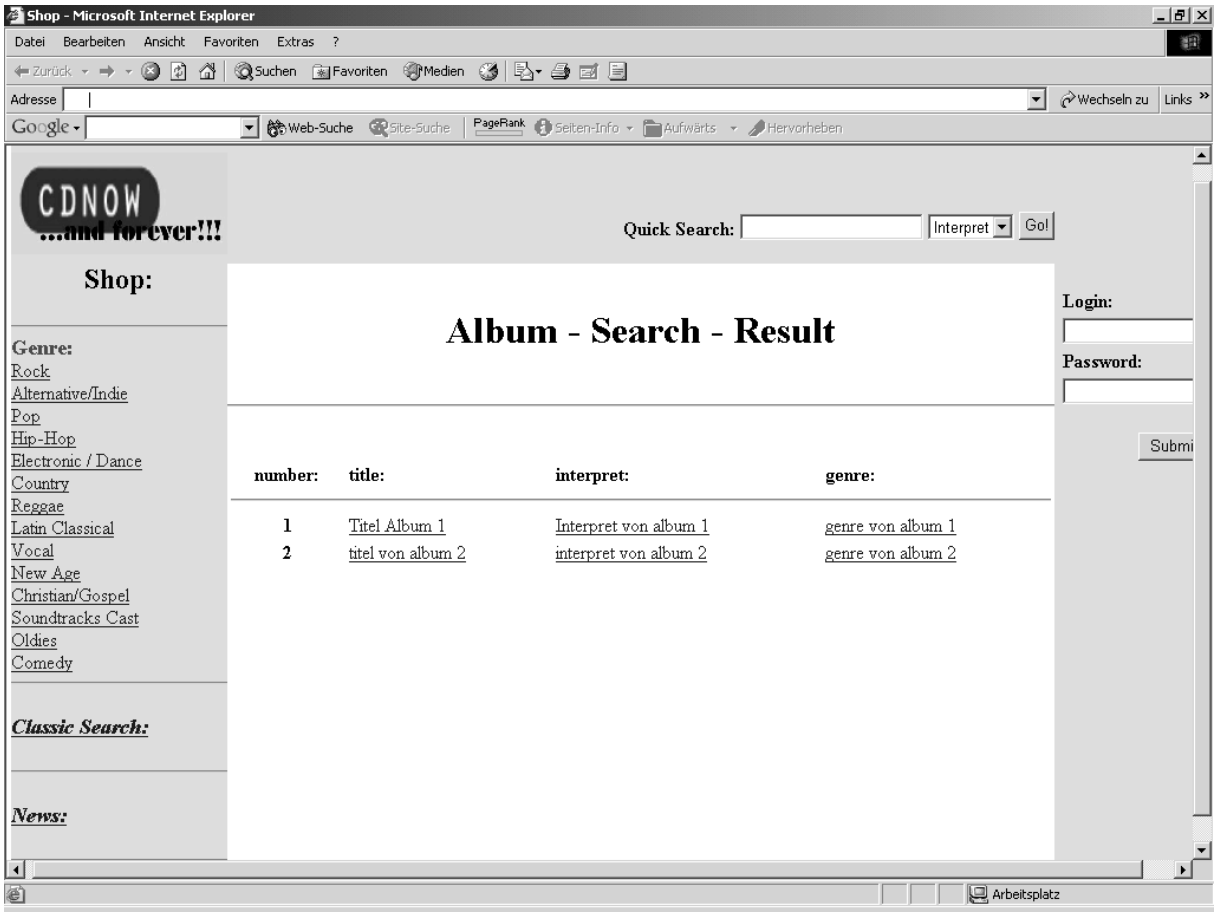
There are several templates implemented to visualize the different features of the website. The templates are named:

- classic_search (represents the classic search)
- interpret_search (represents all matches for interpret)
- album_search (represents all matches for album)
- track_search (represents all matches for track)
- user_profile (represents the account data of a user)
- message (represents different types of messages)
- logincheck (checks if the user is logged in)
- welcome (short welcome page with logincheck)
- news (represents the news)
- download (represents the download menu)
- buy (represents the bought title)

For example the XML-file that is produced by an “album_search”-request looks like this:

```
<album_search parameter="d=a&amp;amp;f=f&amp;amp;s=d&amp;amp;" loggedIn=”Yes”>
    <result>
        <album>
            <title>Titel of album 1</title>
            <interpret>Interpret of album 1</interpret>
            <genre>Genre of album 1</genre>
        </album>
        <album>
            <title>Titel of album 2</title>
            <interpret>Interpret of album 2</interpret>
            <genre>Genre of album 2</genre>
        </album>
    </result>
</album_search>
```

The XSL delivers all the layout information and the result of the parsing is an HTML-file that looks like this:


Shop - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten Medien
Adresse
Wechseln zu Links
Google Web-Suche Site-Suche PageRank Seiten-Info Aufwärts Hervorheben
CDNOW
...and forever!!!
Quick Search: Interpret Go!
Shop:
Genre:
Rock
Alternative/Indie
Pop
Hip-Hop
Electronic / Dance
Country
Reggae
Latin Classical
Vocal
New Age
Christian/Gospel
Soundtracks Cast
Oldies
Comedy
Classic Search:
News:
Album - Search - Result
number: title: interpret: genre:
1 Titel Album 1 Interpret von album 1 genre von album 1
2 titel von album 2 interpret von album 2 genre von album 2
Login:
Password:
Submi
Arbeitsplatz

6. What kinds of Problems occurred?

Different problems occurred during the implementation phase. Due to the fact that we have an ALBUM table that is filled with over 400000 entries there occurred parsing problems while the parsing of album titles in foreign typesets (like Chinese…)
There were also problems in the passing of SQL queries. Because of we had very much keywords to search for (artist, title, year of appearance…) it was complicated to filter the really wanted data.

7. Conclusion

The project was for all participants very interesting and knowledge gaining. We learned to use several techniques as XML/XSL , JSP, … as well as we learned to manage a project over distance. In the end we can all say this was a very interesting experience where we had fun and learned much from each other.