Database Raport

1. Mutual classes

1.1. BasePage.java

BasePage.java class gets connecetion between wicket id and java's dynamic object. HomePage.java ArtistHomePage.java ... etc all classes extends BasePage.java.

```
public class BasePage extends WebPage {

	public BasePage() {
		// TODO Auto-generated constructor stub
		this(null);
	}

	public BasePage(IModel<?> model) {
		super(model);
		Date saat = new Date();
		Label etiket = new Label("zaman", saat.toString());
		this.add(etiket);
		NavigationPanel panelObject = new NavigationPanel("navigation");
		this.add(panelObject);
	}

}
```

Any class that extends BasePage can be considered as above codes allready written in their .java files. It makes connection with "zaman" wicket id and "navigation" wicket id. Using "zaman" wicket id and

```
Date saat = new Date();
Label etiket = new Label("zaman", saat.toString());
this.add(etiket);
above codes prints screen time.

Using "navigation" wicket id and
NavigationPanel panelObject = new NavigationPanel("navigation");
this.add(panelObject);
above code places navigation panel to screen
```

1.2. HomePage.java & HomePage.html

Application Home page has has navigation panel to directs user each class Artist Person ... etc. Since it extends BasePage it shows time.

1.3. NavigationPanel.java & NavigationPanel.html

Navigation Panels allow user to click tabs and directs to related classes home page. Our program has 12 class Threfore navigation panel has 12 tabs and recover database tab

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">

<title>NavigationPanel</title>
<link rel="stylesheet" href="style1.css" />
<!-- wicket id html dekini java da ulasmak icindir -->
</head>
<body>

      <wicket:panel>
      <div id="wrapper">
       <div id="header">
    <div>
      <p>OK</p>
      <p class="more"><a href="#"><img src="images/more_red.jpg" width="71"
height="14" alt="More" /></a></p>
    </div>
  </div>
  <div id="nav">
                  <a href="#" wicket:id="home"> Home</a>
                    <a href="#" wicket:id="artist_class">Artists</a>
                          <a href="#" wicket:id="person_class">Person</a>
                        <a href="#" wicket:id="studios_class">Studios</a>
                          <a href="#" wicket:id="location_class">Location</a>
                          <a href="#" wicket:id="song_class">Song</a>
                          <a href="#" wicket:id="url_class">URL</a>
                          <a href="#" wicket:id="genre_class">Genre</a>
                          <a href="#" wicket:id="instrument_class">Instrument</a>
                          <a href="#" wicket:id="award_class">Award</a>
                          <a href="#" wicket:id="Album">Album</a>
                          <a href="#" wicket:id="Festival">Festival</a>
                          <a href="#" wicket:id="Lyrics">Lyrics</a>
                          <a href="#" wicket:id="recover_db">Recover DataBase</a>
                          <a href="#" wicket:id="user_class">User</a>

 </div>
  </div>
      </wicket:panel>

</body>
</html>
```

On above each class placeholder and their wicket id is given on NavigationPanel.html. Their wicket id precessed in WicketApplication.java

```
Link targetArtistClass = new Link("artist_class") {

                  @Override
                  public void onClick() {
                        // TODO Auto-generated method stub
                        setResponsePage(new ArtistHomePage());

                  }

            };
```

```
this.add(targetArtistClass);
```

Link class provides to redirect each realated classes home page. Wicket id is used Link class constructor. Also Link class constructor has onClick() method. onClick() method executes when user clicks related tab on NavigationPanel.html. SetResponsePage() method redirects user home pages. In order to provide Link between panel tabs, Link object of related navigation panel tab added to program with following code.

this.add(targetArtistClass).

1.4. Start.java

Start.java class has main method of program. In java initially public static void main(String arg[]) method starts. Also crucial methods that arrenges our localhost and connection values in Start.java file.

1.5. Wicket Application.java

WicketApplication is one of the significant class of our program it generates collections of each class. JDBC methods adds, deletes or updates this collections. Wicket application class is reachable for every class in this program.

For every collection wicket application class has following codes.

```
private IPersonCollection personCollection;
```

```
person collection declared globally as an interface type.
```

```
personCollection = new PersonCollectionJDBC(dbFilePath, context);
```

```
personCollection initialized with JDBC methods to process. Also Context of program
and file directory of database sent as parameter to PersonCollectionJDBC method.
```

```
public IPersonCollection getPersonCollection() {
        return personCollection;
}
```

```
getPersonCollection() method returns collections. Since WicketApplication.class
can be reached from all program. Every class can add, delete or update their
collections in their classes
```

Following codes show how to use WicketApplication object and how to reach collections

```
Person person = (Person) this.getModelObject();
WicketApplication app = (WicketApplication) this.getApplication();
IPersonCollection personCollection = app.getPersonCollection();
```

2. Artist Class
2.1.
3. Person Class
3.1. Person.java
3.1.1. Usage

In order to generate a class, initially attributes, constructors, setters and getters must be generated.

3.1.2. Attributes

In this Person.java file fallowing attributes are declared

```
private Integer _id = null;
private String name;
private String surname;
private Integer age;
```

3.1.3. Methods

Person.java file contains undermentioned methods.

```
	public Integer get_id() {
		return _id;
	}

	public void set_id(Integer _id) {
		this._id = _id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getSurname() {
		return surname;
	}

	public void setSurname(String surname) {
		this.surname = surname;
	}

	public Integer getAge() {
```

```
		return age;
	}

	public void setAge(Integer age) {
		this.age = age;
	}

	public Person() {
		// TODO Auto-generated constructor stub
	}

	public Person(String isim) {
		// TODO Auto-generated constructor stub
		this.setName(isim);
	}
```

3.2. IpersonCollection.java

IpersonCollection.java file is an interface that contains prototypes of methods. IpersonCollection.java provides our program polymorphism flexibility. Other classes of our program **implements** IpersonCollection interface.

IpersonCollection.java declared as interface keyword and contains following prototype of methods

```
public interface IPersonCollection {

	public void addPerson(Person person);

	public void deletePerson(Person person);

	public void updatePerson(Person person);

	public List<Person> getPerson();

}
```

3.3. Person CollectionJDBC

PersonCollectionJDBC file provides database connection and necessary methods. It implements IpersonCollection interface

3.3.1. Methods

`public PersonCollectionJDBC(String dbFilePath, ServletContext context): provides database connections.`

`public void addPerson(Person person): adds person to PERSONTABLE`

public void deletePerson(Person person): delete selected person infromation from PERSONTABLE

public void updatePerson(Person person): updates person

public List <Person> getPerson(): returns all rows of PERSONTABLE

3.3.2. Codes & Explanations

Each try catch blocks used for catching probable exceptions

```
public PersonCollectionJDBC(String dbFilePath, ServletContext context) {

		try
			Class.forName("org.sqlite.JDBC");

		} catch (ClassNotFoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

		try {
			String jdbcUrl = "jdbc:sqlite:" + dbFilePath;// databases are

			// referenced using

			// JDBC URLs.
			conn = DriverManager.getConnection(jdbcUrl);
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
```

String dbFilePath: This parameter holds our database file directory.

ServletContext context: This parameter holds context of program

Class.forName("org.sqlite.JDBC"):In this line program registers JDBC driver. Since "SQLite" is used in this project our driver name "org.sqlite.JDBC"

String jdbcUrl = "jdbc:sqlite:" + dbFilePath:This line is used at second try-catch block. Databases are referenced with their urls. Since we use SQLite, we add "jdbc:sqlite:" before dbFilePath.

3.3.3. Add

```
	public void addPerson(Person person) {
```

```
            // TODO Auto-generated method stub
            String query = "INSERT INTO PERSONTABLE (NAME, SURNAME, AGE) VALUES
(?, ?, ?)";

            try {
                  PreparedStatement statement = conn.prepareStatement(query);
                  statement.setString(1, person.getName());
                  statement.setString(2, person.getSurname());
                  statement.setInt(3, person.getAge());
                  statement.executeUpdate();
                  statement.close();

            } catch (SQLException e1) {
                  // TODO Auto-generated catch block
                  e1.printStackTrace();
            }

      }
```

This methods adds person to PERSONTABLE.

First of all we initialized query that inserts data to our table.

```
String query = "INSERT INTO PERSONTABLE (NAME, SURNAME, AGE) VALUES (?, ?, ?)";
```

Then statement is declared, When developers deal with databases statement must be used. Basically statement uses given query and functions helps us to handle databases.

```
PreparedStatement statement = conn.prepareStatement(query);
```

Statement is generated using Connection object “conn”. Conn object defined in constructor it identifies which database will be processed.

```
statement.setString(1, person.getName());
statement.setString(2, person.getSurname());
statement.setInt(3, person.getAge());
```

above codes sets question marks of queries using statements

```
statement.executeUpdate();
statement.close();
```

Statement is executed. After necessary insertions have done statement is closed

3.3.4. Delete

```
   public void deletePerson(Person person) {
            // TODO Auto-generated method stub

            String query = "DELETE FROM PERSONTABLE WHERE (ID = ?)";
            PreparedStatement statement;
```

```
            try {
                  statement = conn.prepareStatement(query);
                  statement.setInt(1, person.get_id());
                  statement.executeUpdate();
                  statement.close();

            } catch (SQLException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
}
```

In delete method same algorithm progressed with addPerson() method

- Query defined
- Appropriate statement is generated
- Question Mark is setted
- Statment is executed

Only difference is `person.get_id()) method is used to identify which row will be deleted.`

3.3.5. Update

```
public void updatePerson(Person person) {
    // TODO Auto-generated method stub
            String query = "UPDATE PERSONTABLE SET NAME=?, SURNAME=?, AGE=? WHERE
(ID=?)";
            try {

                  PreparedStatement statement = conn.prepareStatement(query);
                  statement.setString(1, person.getName());
                  statement.setString(2, person.getSurname());
                  statement.setInt(3, person.getAge());
                  statement.setInt(4, person.get_id());
                  statement.executeUpdate();// executeUpdate returns nothing
                  statement.close();

            } catch (SQLException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }

    }
```

In update method same algorithm progressed

- Query defined
- Appropriate statement is generated
- Question Mark is setted
- Statment is executed

In order to identify which row is updated, “WHERE ID=?” clause is used in query

```
public List<Person> getPerson() {
		// TODO Auto-generated method stub
		List<Person> personLinkedList = new LinkedList<Person>();

		String query = "SELECT ID, NAME, SURNAME, AGE FROM PERSONTABLE";
		Statement statement;
		try {
			statement = conn.createStatement();

			ResultSet results = statement.executeQuery(query);

			while (results.next()) {
				Integer ID = results.getInt("ID");
				String name = results.getString("NAME");
				String surName = results.getString("SURNAME");
				Integer age = results.getInt("AGE");

				Person person = new Person(name);
				person.set_id(ID);
				person.setSurname(surName);
				person.setAge(age);

				personLinkedList.add(person);

			}
			results.close();
			statement.close();

		} catch (SQLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

		return personLinkedList;
	}
```

getPerson() method returns list of all people. Initially linked list is declared then. When developers tries to get set of rows from database, developer must use **ResultSet** class. ResultSet object generates new table from our database.

While(results.next) loop is signifiant it traverse all ResultSet object, which is our new generated table from database.

İnside of this loop each row each data will be inserted on linked list. Finally genaretd linked list returned.

Except the indicated details, same algorithm progressed

- Query defined
- Appropriate statement is generated
- Question Mark is setted
- Statment is executed

3.3.6. s

3.4. PersonDisplayPage

3.4.1. PersonDisplayPage.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Person Display Page</title>
</head>
<nav wicket:id="navigation" />
<body>
	<table>
		<tr>
			<th>Person Name:</th>
			<td wicket:id="name">James</td>
		</tr>
		<tr>
			<th>Person Surname:</th>
			<td wicket:id="surname">Hetfield</td>
		</tr>
		<tr>
			<th>Person Age:</th>
			<td wicket:id="age">51</td>
		</tr>
	</table>
	<p>
		<a href="#" wicket:id="person_edit_link"> Edit Person</a>
	</p>


	<footer>
		<div id="zaman" wicket:id="zaman">asd</div>
	</footer>
</body>
</html>
```

PersonDisplay page contains 3 component to show name, surname and age of the Person. Top of the page navigation panel is placed, bottom of the page it time is shown.

3.4.2. PersonDisplayPage.java

```
public class PersonDisplayPage extends BasePage {

	private Person _person;

	public PersonDisplayPage(Person person) {
```

```
		this.add(new Label("name", person.getName()));
		this.add(new Label("surname", person.getSurname()));
		this.add(new Label("age", person.getAge().toString()));
		_person = person;
		Link editLink = new Link("person_edit_link") {
			@Override
			public void onClick() {
				// TODO Auto-generated method stub
				PersonDisplayPage parent = (PersonDisplayPage) this.getParent();
				this.setResponsePage(new PersonEditPage(parent.getPerson(),
						false));
			}
		};
		this.add(editLink);
	}
	public Person getPerson() {// parent Person display pagedir
		return _person;
	}
}
```

In order to use navigation panel and time PersonDisplayPage.java extends BasePage.java .

this.add(new Label("name", person.getName())). In this line html components added and filled with Labels that represent person information.

PersonDisplayPage contains edit button. Edit button directs user to edit page in order to update person information using given wicket id.

getParent() method gets parent information in terms of java hiearcy. This is key feature of our program if PersonEditPage opened from PersonListPage program knows this person will be updated and edit page started with fallowing constructor.

PersonDisplayPage parent = (PersonDisplayPage) this.getParent();

this.setResponsePage(new PersonEditPage(parent.getPerson(), false));

3.5. PersonDisplayPageLink

This class is used in PersonListForm in order to display person information and directs person display page.

3.6. PersonEditPage

This page is used updating or adding information about person.

3.6.1. PersonEditPage.html

Person edit page has 3 textfields and Save button inside of a form. User wil enter information about person these texfields, then He or She will press save button in order to save to database

3.6.2. PersonEditPage.java

PersonEditPage executes PersonEditForm.java and decides person is updated or inserted.

3.7. PersonEditForm.java

There is exist person_edit_form in PersonEditPage.html and PersonEditForm.java file in our program. In PersonEditPage.html person_edit_form identifed with wicket id. On the other hand PersonEditForm.java file is the file that crucial proceses is done.

```
public class PersonEditForm extends Form {// formda submit var
	private boolean _newPersonKey;

	public PersonEditForm(String id, Person person, boolean NewPersonKey) {
		super(id);
		// TODO Auto-generated constructor stub

		CompoundPropertyModel model = new CompoundPropertyModel(person);
		this.setModel(model);

		this.add(new TextField("name"));
		this.add(new TextField("surname"));
		this.add(new TextField("age"));

		_newPersonKey = NewPersonKey;
	}

	@Override
	protected void onSubmit() {
		// TODO Auto-generated method stub
		super.onSubmit();

		Person person = (Person) this.getModelObject();
		WicketApplication app = (WicketApplication) this.getApplication();
		IPersonCollection personCollection = app.getPersonCollection();
```

```
            if (this._newPersonKey)
                  personCollection.addPerson(person);
            else
                  personCollection.updatePerson(person);

      }
```

```
CompoundPropertyModel model = new CompoundPropertyModel(person);
this.setModel(model);
```
Above codes sets model type.Now Object types and names arrenged.

```
this.add(new TextField("name"));
this.add(new TextField("surname"));
this.add(new TextField("age"));
```

Above codes inserts textfields to edit page, in order to provide fields to enter.

```
_newPersonKey = NewPersonKey;
```
This assignment is crucial for our program. _newPersonKey is a flag that shows values retrieved from PersonEditPage is will be updated or will be inserted.

If user press save button in person_edit_form, onSubmit() method is executes. OnSubmit method decides person whether person is inserted or deleted based on _newPersonKey.

3.8. PersonHomePage

Person Home Page has a list that redirects user List, Edit or Search Pages.

3.8.1. PersonHomePage.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<link rel="stylesheet" href="style1.css">
<!-- close tag yok -->
<title>Person Home Page</title>
</head>
<body>
   <nav wicket:id="navigation">asf</nav>

   <div class="menu">
         <ul>
               <li><a href="#" wicket:id="add_person">Add Person</a></li>
               <li><a href="#" wicket:id="list_person">List Person</a></li>
               <li><a href="#" wicket:id="search _person">Search</a></li>


         </ul>
   </div>

   <h1>Person Home Page</h1>
```

```
        <footer>
                <div id="zaman" wicket:id="zaman">asd</div>
        </footer>
    </body>
</html>
```

In PersonHomePage.html clicked items redirected related pages.

3.8.2. PersonHomePage.java

```
public class PersonHomePage extends BasePage {

        public PersonHomePage() {

                // TODO Auto-generated constructor stub

                Link targetEditPage = new Link("add_person") {// link classı yapmadim

                        @Override

                        public void onClick() {

                                // TODO Auto-generated method stub

                                Person person = new Person();

                                setResponsePage(new PersonEditPage(person));

                        }

                };

                this.add(targetEditPage);

                Link targetListPage = new Link("list_person") {

                        @Override

                        public void onClick() {

                                // TODO Auto-generated method stub

                                setResponsePage(new PersonListPage());

                        }

                };

                this.add(targetListPage);          }          }
```

In this class depends on wicket id of clicked operation PersonHomePage redirects user using Link Class and setResponsePage() method.

3.9. PersonListPage

3.9.1. PersonListPage.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<link rel="stylesheet" href="style1.css" />
<title>Person List Page</title>
</head>
<body>
	<nav wicket:id="navigation">asd</nav>
	<form action="#" wicket:id="person_list_form">
		<div wicket:id="selected_person">
			<table>
			<tr>
			<th>Name Surname Age </th>
			</tr>
				<tr wicket:id="person_list_tr">
					<td><input type="checkbox"
wicket:id="selected_checkbox" /></td>
					<td>
					<a href="#" wicket:id="person_link">
					<span wicket:id="name">Person Name</span>
					<span wicket:id="surname">Person Surname</span>
					<span wicket:id="age">Person Age</span>
					</a>
					</td>
				</tr>
			</table>
		</div>

		<input type="submit" value="DELETE" name="delete_submit_button"/>
	</form>

	<footer id="zaman" wicket:id="zaman">ghj</footer>
</body>
	</html>
```

PersonListPage.html has a form that holds checkboxes, tables, spans, placeholders, submit button.

Checkboxes and submit button used for deleting items from database. In order to update person each person in table connected to link to PersonEditPage.

3.9.2. PersonListPage.java

PersonListPage java just generates form and adds it to page.

3.9.3. PersonListForm.java

PersonListPage.java class is relatively complex class. So codes explain step by step.

```
selected_persons_list = new LinkedList<Person>();
		// check edilenler gruplandı
		CheckGroup<Person> selectedPersonGroup = new CheckGroup<Person>(
				"selected_person", this.selected_persons_list);
		this.add(selectedPersonGroup);
```

In this part Checked items grouped in linked list.

```
WicketApplication app = (WicketApplication) this.getApplication();
IPersonCollection personCollection = app.getPersonCollection();
List<Person> allPeople = personCollection.getPerson();
```

We get collection above codes.

Following codes helps us to fill PersonListPage to person informations. First of all PropertyListView is used to connect html table row and our linked list data.

Pouplate item is populates all item in linked list.

PersonDisplayPageLİnk provides Link for every row of table.

After inserting checkbox for every list item. PropertyListView object is added to selectedPersonGroup hieracy.

```
PropertyListView personListView = new PropertyListView(
                        "person_list_tr", allPeople) {

                @Override
                protected void populateItem(ListItem arg0) {
                        // TODO Auto-generated method stub
                        Person person = (Person) arg0.getModelObject();

                        PersonDisplayPageLink link = new PersonDisplayPageLink(
                                        "person_link", person);

                        link.add(new Label("name"));
                        link.add(new Label("surname"));
                        link.add(new Label("age"));

                arg0.add(new Check("selected_checkbox", arg0.getModel()));

                        arg0.add(link);

                }

        };

        selectedPersonGroup.add(personListView);
    }
```

OnSubmit button deletes selected persons from collection and database.

```
    protected void onSubmit() {
        // TODO Auto-generated method stub
        super.onSubmit();

        WicketApplication app = (WicketApplication) this.getApplication();
        IPersonCollection koleksiyon = app.getPersonCollection();
        for (Person person : this.selected_persons_list)
            koleksiyon.deletePerson(person);
```

```
        this.setResponsePage(new PersonListPage());
    }
```

3.10. SearchPackage

In order to implement serch method for person class we grouped .html and .java files into a package.

3.10.1. PersonSearchPage.html

```
<h>Person Search Page</h>

<p>Enter Person Name to Search</p>

<form action="#" wicket:id="person_search_form">
      <input type="text" wicket:id="searchWord">
      <input type="submit" value="Search">

</form>
```

PersonSearchPage has texfield that user enters text to search and submit button inside of a form.

3.10.2. PersonSearchPage.java

This page generates form and adds it PersonSearchPage.

3.10.3. PersonSearchListPage.html

```
<title>Person Search List Page</title>
</head>
<body>
      <nav wicket:id="navigation">asd</nav>
      <h1>Person search list page</h1>

      <form action="#" wicket:id="person_search_list_form">
            <p>
                  Searched string:<span wicket:id="girilen">SPAN</span>
            </p>

            <table>

                  <th>Name Surname Age</th>
```

```
                <tr wicket:id="founded_person_list_tr">
                      <td><span wicket:id="name">Person Name</span>
                      <span  wicket:id="surname">Person Surname</span>
                       <span wicket:id="age">Person Age</span>
                      <span wicket:id="band">Person Band</span>
                </tr>

          </table>
    </form>
```

This page has a table inside a form table holds search results.

3.10.4. PersonSearchListForm.java

```
public class PersonSearchListForm extends Form {
      private List<Person> searched_persons_list;

      public PersonSearchListForm(String id, String girilenKisi) {
            super(id);
            // TODO Auto-generated constructor stub

            this.add(new Label("girilen", girilenKisi));

            searched_persons_list = new LinkedList<Person>();

            WicketApplication app = (WicketApplication) this.getApplication();
            IPersonCollection personCollection1 = app.getPersonCollection();
            List<Person> allPeople1 =
personCollection1.searchPerson(girilenKisi);

            PropertyListView foundPersonListView = new PropertyListView(
                        "founded_person_list_tr", allPeople1) {

                  @Override
                  protected void populateItem(ListItem arg0) {
                        // TODO Auto-generated method stub
                        Person person = (Person) arg0.getModelObject();

                        arg0.add(new Label("name"));
                        arg0.add(new Label("surname"));
                        arg0.add(new Label("age"));
                        arg0.add(new Label("band"));
                        arg0.add(new Label("location"));
                        arg0.add(new Label("instrument"));

                  }
            };
            this.add(foundPersonListView);
```

```
	}
}
```

3.10.5. sad

4. Location Class

4.1. Location.java

4.1.1. Usage

In order to generate a class, initially attributes, constructors, setters and getters must be generated.

4.1.2. Attributes

```
	Integer id;

	String continent;

	String country;

	String town;
```

4.1.3. Methods

```
public Integer getId() {

		return id;

	}


	public void setId(Integer id) {

		this.id = id;

	}


	public String getContinent() {

		return continent;
```

```
}

public void setContinent(String continent) {
	this.continent = continent;
}

public String getCountry() {
	return country;
}

public void setCountry(String country) {
	this.country = country;
}

public String getTown() {
	return town;
}

public void setTown(String town) {
	this.town = town;
}

public Location() {
	// TODO Auto-generated constructor stub
}
```

4.1.4. E

4.2. Ee

```
public interface ILocationCollection {
	public void addLocation(Location location);
	public void deleteLocation(Location location);
	public void updateLocation(Location location);
	public List<Location> getLocation();

}
```

4.3. LocationCollectionJDBC

4.3.1. Methods

```
public void addLocation(Location location) {
		// TODO Auto-generated method stub

		String query = "INSERT INTO LOCATIONTABLE (CONTINENT, COUNTRY, TOWN)
VALUES (?, ?, ?)";

		try {
			PreparedStatement statement = conn.prepareStatement(query);
			statement.setString(1, location.getContinent());
			statement.setString(2, location.getCountry());
			statement.setString(3, location.getTown());
			statement.executeUpdate();
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
```

}

CLASSES

1. Studios

This class can carry add, list, delete, update and search operations out. The importance studios class is connections with other classes.

1.1. Attributes

Studios class has 6 attributes. Attributes are shown below with their variable types.

```
private String name;

private String location;

private String owner;

private Integer foundationDate;

private String artist;

private String link;
```

name attribute is used for name of a studio. location attribute is used for location of studio. Using location attribute, we will connect studios class and location class. owner attribute is used for owner of studio. foundationDate shows foundation date of a studio. artist attribute is used to show artists of studio. Using artist attribute, we will connect studios class and artist class. link attribute is used to show web connections of studio. Using link attribute, we will connect studios class and URL class.

1.2. Connections

We use alter table property for connections. Firstly alter table additions is written in init.sql file. After that we supply neccessary changes on StudiosCollectionJDBC file.

1.2.1. init.sql

```
ALTER TABLE STUDIOSTABLE ADD COLUMN FK_LOCATION INTEGER REFERENCES
LOCATIONTABLE(ID);
ALTER TABLE STUDIOSTABLE ADD COLUMN FK_ARTIST INTEGER REFERENCES ARTISTTABLE(ID);
ALTER TABLE STUDIOSTABLE ADD COLUMN FK_LINK INTEGER REFERENCES URLTABLE(ID);
```

1.2.2. StudiosCollectionJDBC

A query is created. This query is used to connect searched id of alter table. For instance, location table and studios table is connected with this query:

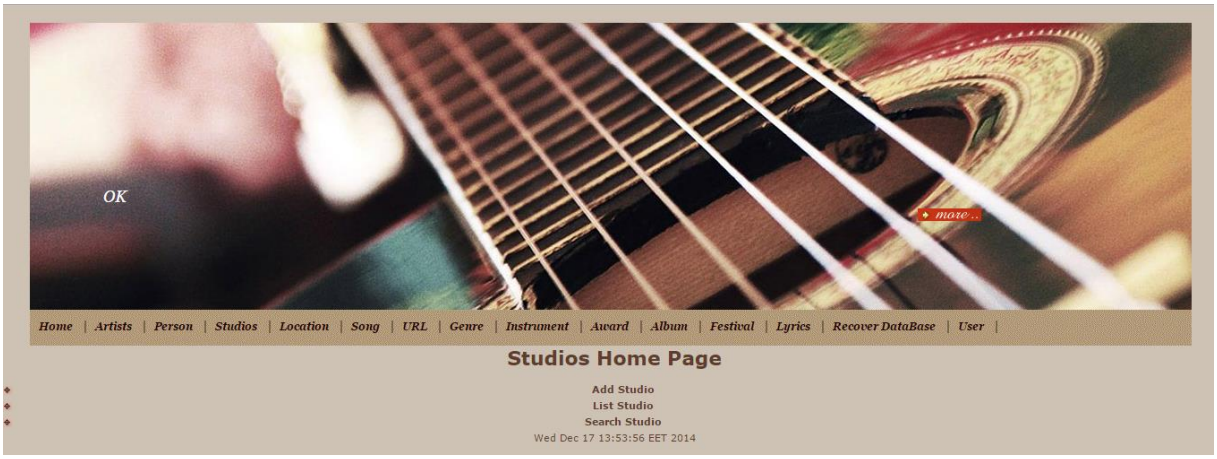
```
String query0 = "SELECT ID FROM LOCATIONTABLE WHERE TOWN ='%" +
studio.getLocation() + "%'";
```

As seen in sql code, location attribute of studios class is compared with town attribute in location class, if it is pair, connection is done.

ID	NAME	LOCATION	OWNER	FOUNDATIONDATE	ARTIST	LINK	FK_LOCATION
1	Smart Studios	Wisconsin	Steve Marker	1983	Nirvana	http://en.wikipedi...	1

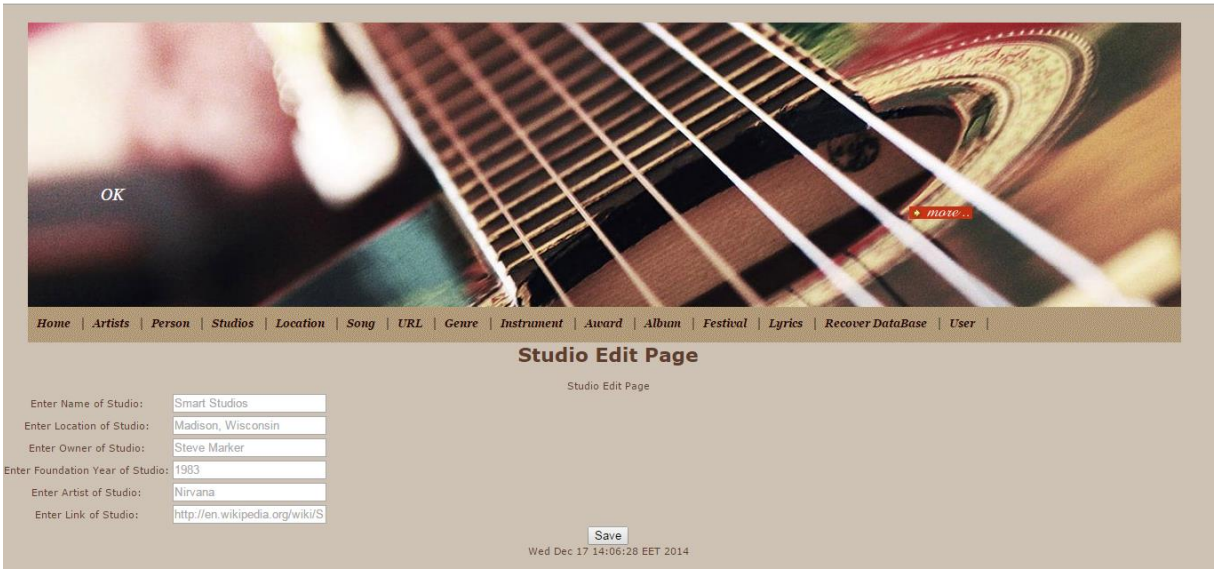
1.3. Operations

For realizing operations, firstly we click on Studios tab on our website. It send user to Studios Home Page.



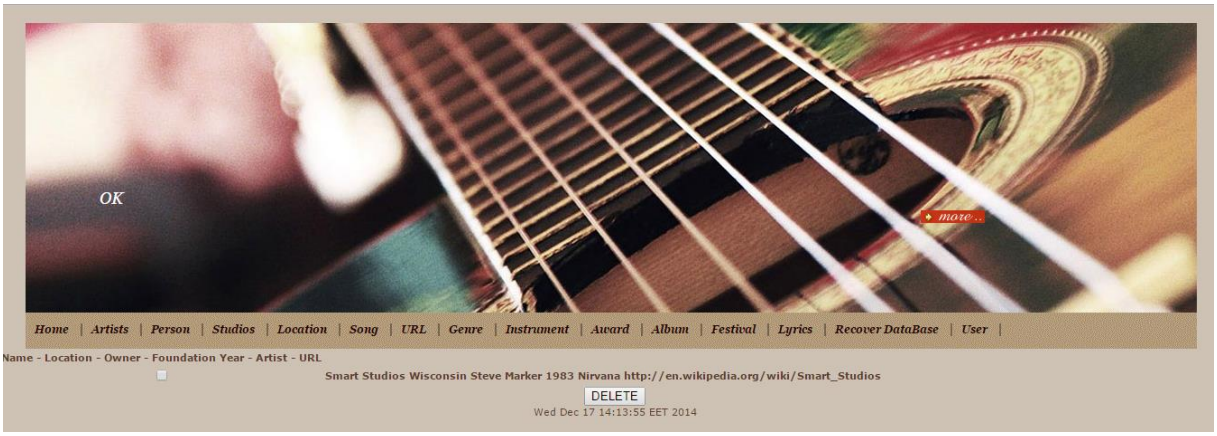
1.3.1. Add Operation

In add operation, we add one row to our sql table. User clicks on Add Studio link. Then user fills the blanks in this form page and click on “Save” button. After clicked on “Save” button, list operation is realized.



1.3.2. List Operation

In list operation user can see studios in our sql table. User clicks on “List Studio” link.

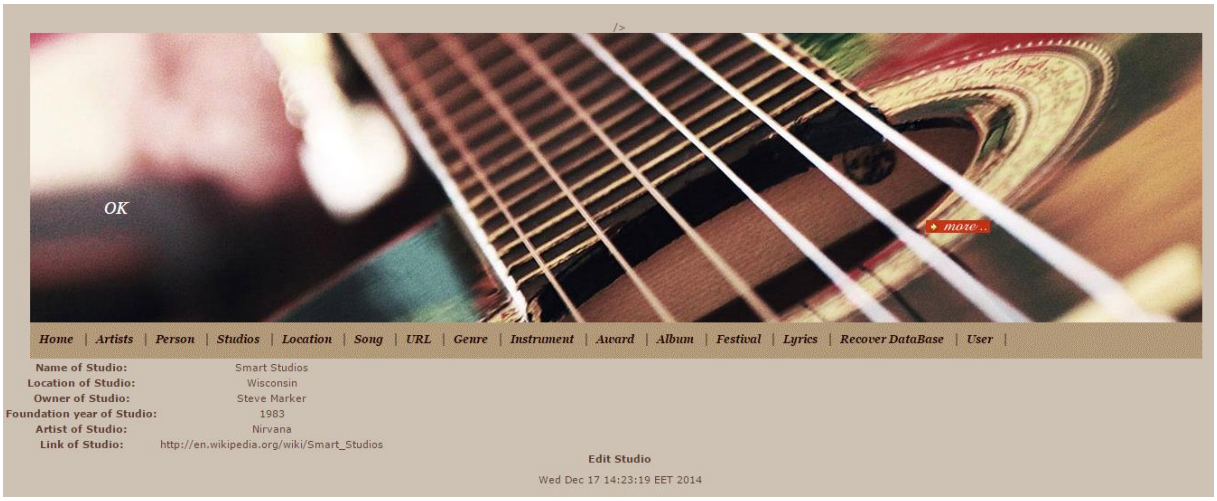


1.3.3. Delete Operation

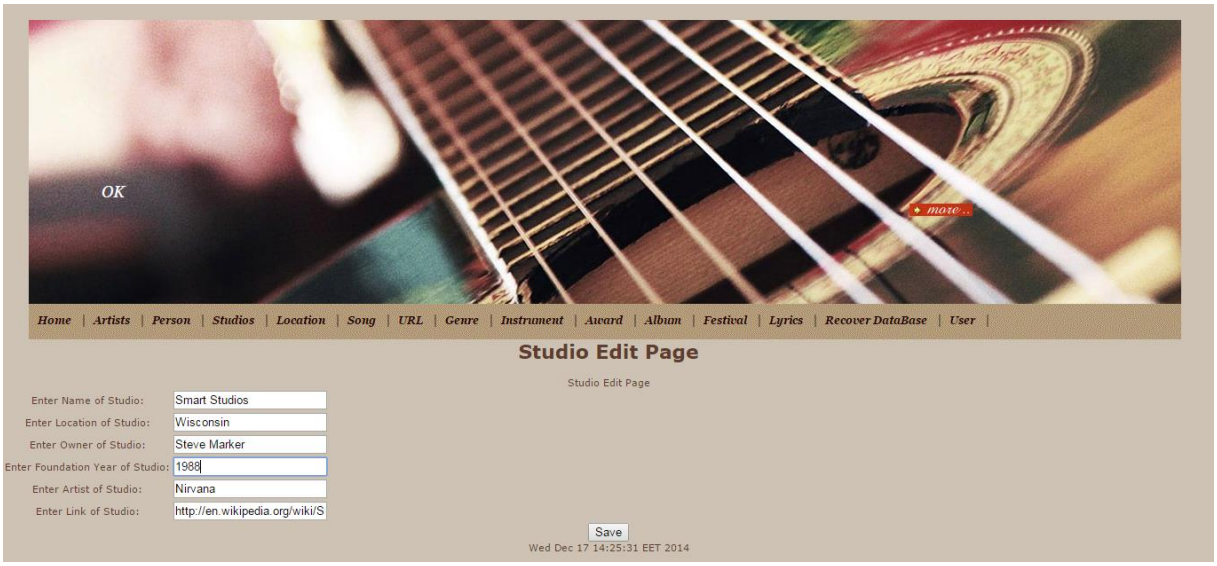
In delete operation, user can delete one row from our sql table. User clicks on check box next to listed compenent which will be deleted. Then there is a “delete” button on bottom. When clicked this button, it is deleted.

1.3.4. Update Operation

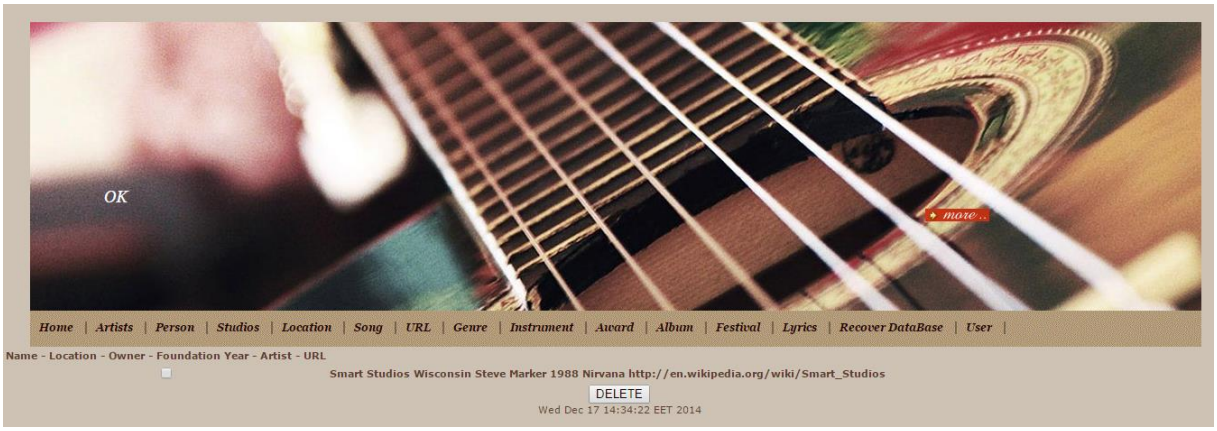
In update operation, user can update one row from our sql table. User clicks on listed compenent, Studio Display page is shown on screen.



If user wants to update this informations , “Edit Studio” button is clicked then edit form page is shown on screen

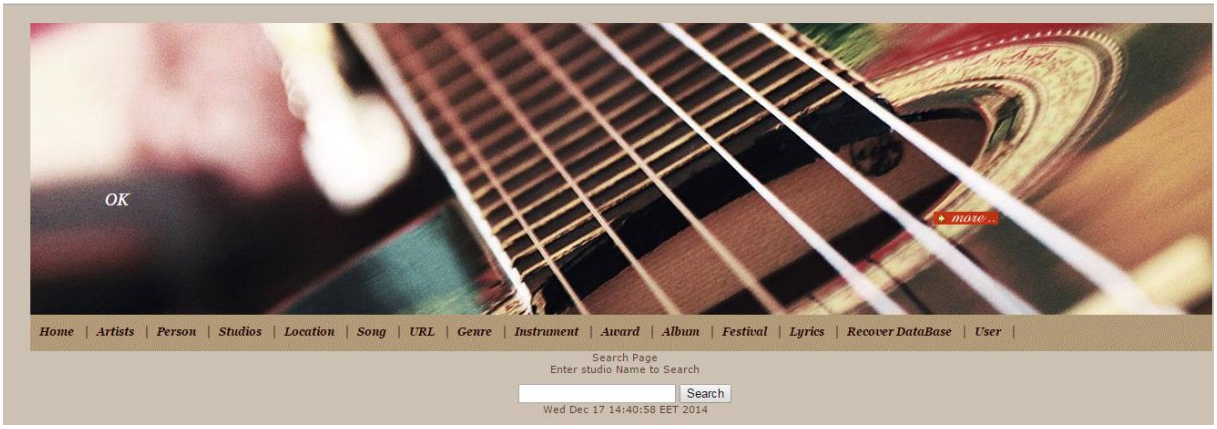


Then user fills parts which will be updated in this form page and click on “Save” button. After clicked on “Save” button, list operation is realized.

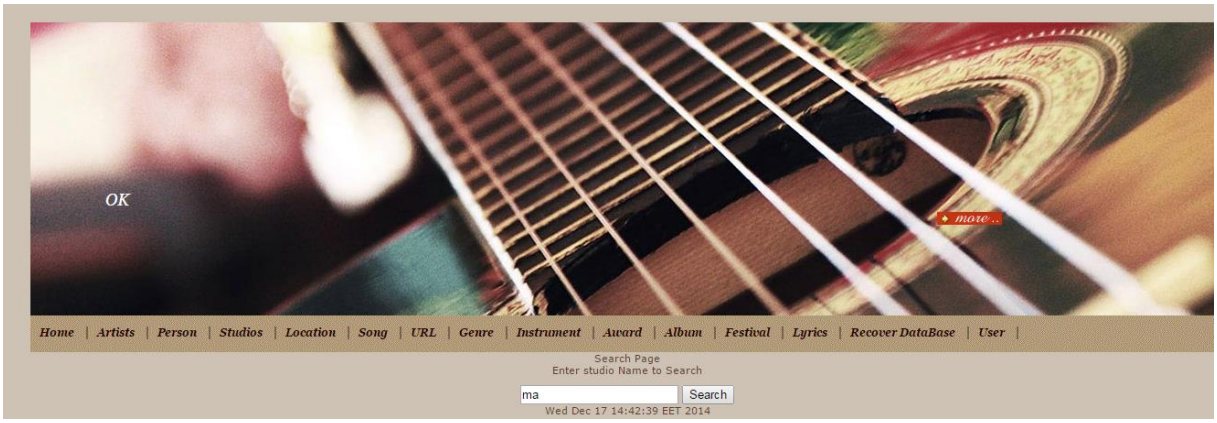


1.3.5. Search Operation

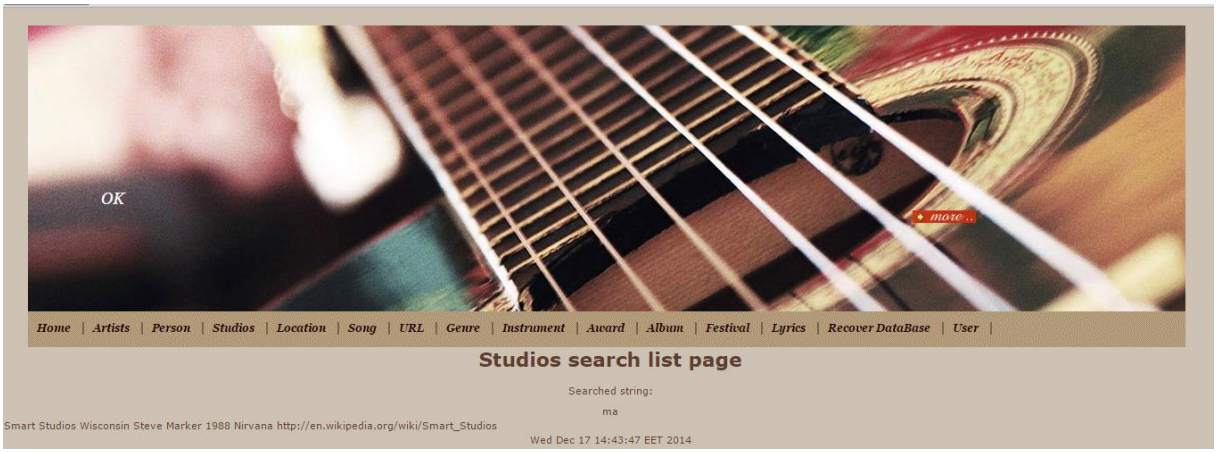
In search operation, any word can be searched in name of studio then all found rows from sql table is listed. To realize this operation, firstly, “Search Studio” link is clicked. Then Studio Search Page is shown.



Then searched word is fill in blank than click on “Search” button.



Then all found rows from sql table is listed.



2. Song

This class can carry add, list, delete, update and search operations out. Song class is one of the most important part of our website. Songs are first component of music.

2.1. Attributes

Studios class has 4 attributes. Attributes are shown below with their variable types.

private String name;

private String lyric;

private String link;

private String album;

name attribute is used for name of a song. lyric attribute is used for lyric of song. Using lyric attribute, we will connect song class and lyric class. link attribute is used to show web connections of songs. Using link attribute, we will connect song class and URL class.

album attribute is used for album of song. Using album attribute, we will connect song class and album class.

2.2. Connections

We use alter table property for connections. Firstly alter table additions is written in init.sql file. After that we supply neccessary changes on StudiosCollectionJDBC file.

2.2.1. init.sql

```
ALTER TABLE SONGTABLE ADD COLUMN FK_ALBUM INTEGER REFERENCES ALBUMTABLE(ID);
ALTER TABLE SONGTABLE ADD COLUMN FK_LYRIC INTEGER REFERENCES LYRICSTABLE(ID);
ALTER TABLE SONGTABLE ADD COLUMN FK_LINK INTEGER REFERENCES LINKTABLE(ID);
```

2.2.2. StudiosCollectionJDBC

A query is created. This query is used to connect searched id of alter table. For instance, album table and song table is connected with this query:

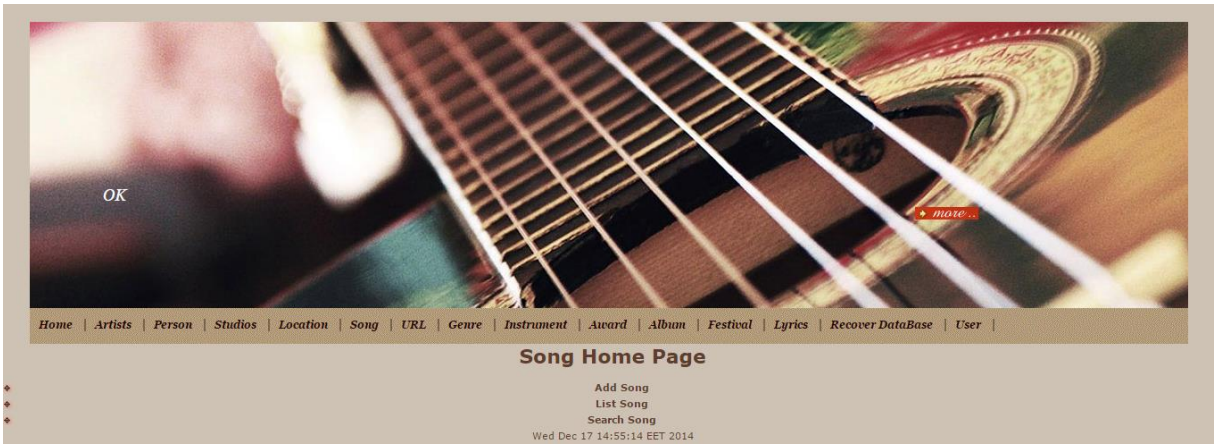
```
String query0 = "SELECT ID FROM ALBUMTABLE WHERE NAME LIKE '%"+ song.getAlbum() +
"%'";
```

```
As seen in sql code, album attribute of song class is compared with name attribute
in album class, if it is pair, connection is done.
```

ID	NAME	LYRIC	LINK	ALBUM	FK_ALBUM	FK_LYRIC	FK_LINK
1	Fear of the Dark	When the light be...	http://www.yout...	Album	1		
2	Here Comes The R...	I want to walk in th...	http://www.youtub...	Album	2		
3	Litaliano	Lasciatemi cantare,...	http://www.youtub...	Album	3		

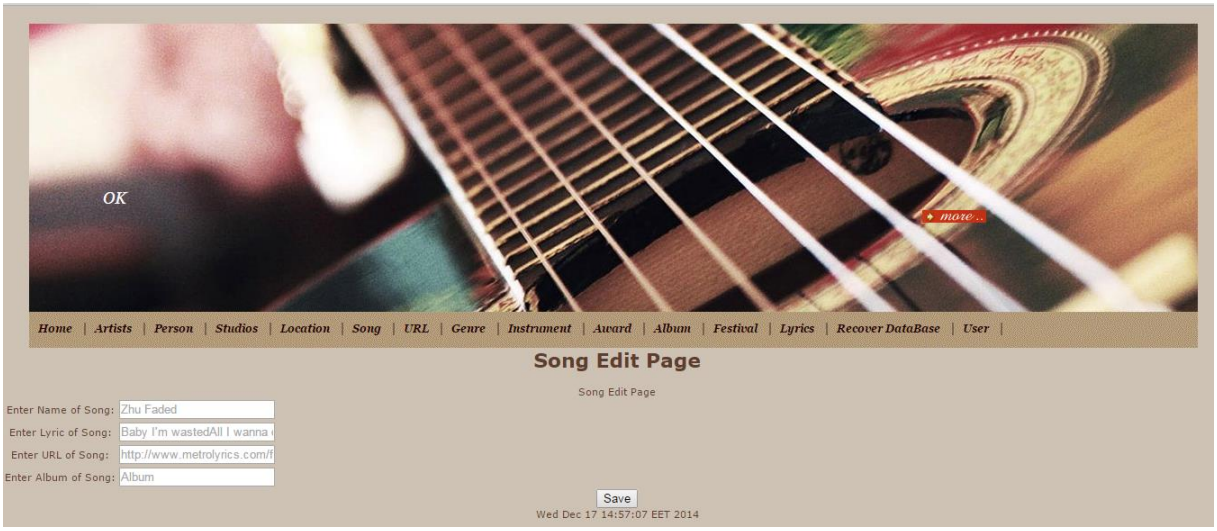
2.3. Operations

For realizing operations, firstly we click on Song tab on our website. It send user to Song Home Page.



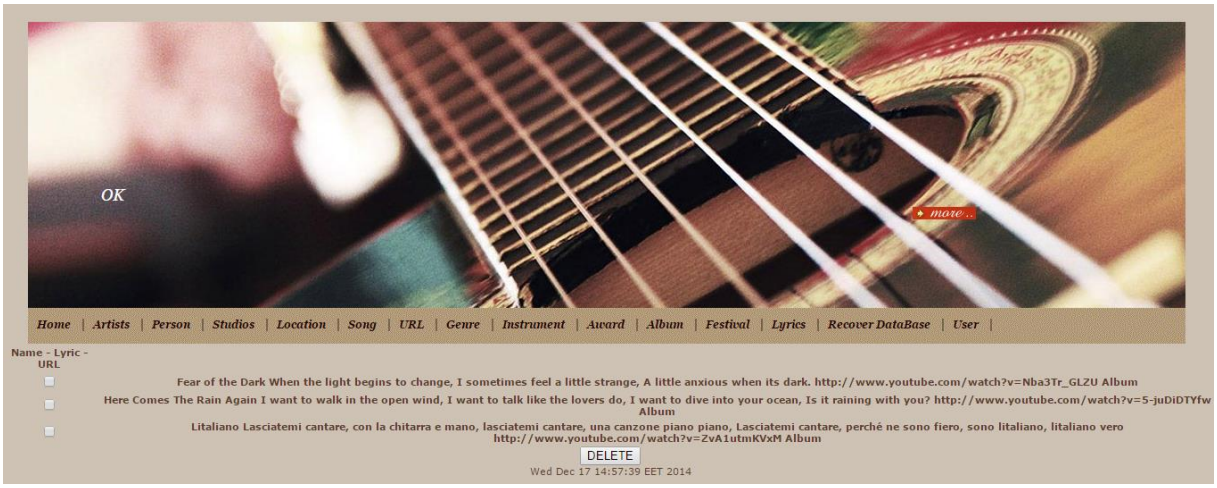
2.3.1. Add Operation

In add operation, we add one row to our sql table. User clicks on Add Song link. Then user fills the blanks in this form page and click on “Save” button. After clicked on “Save” button, list operation is realized.



2.3.2. List Operation

In list operation user can see studios in our sql table. User clicks on “List Studio” link.

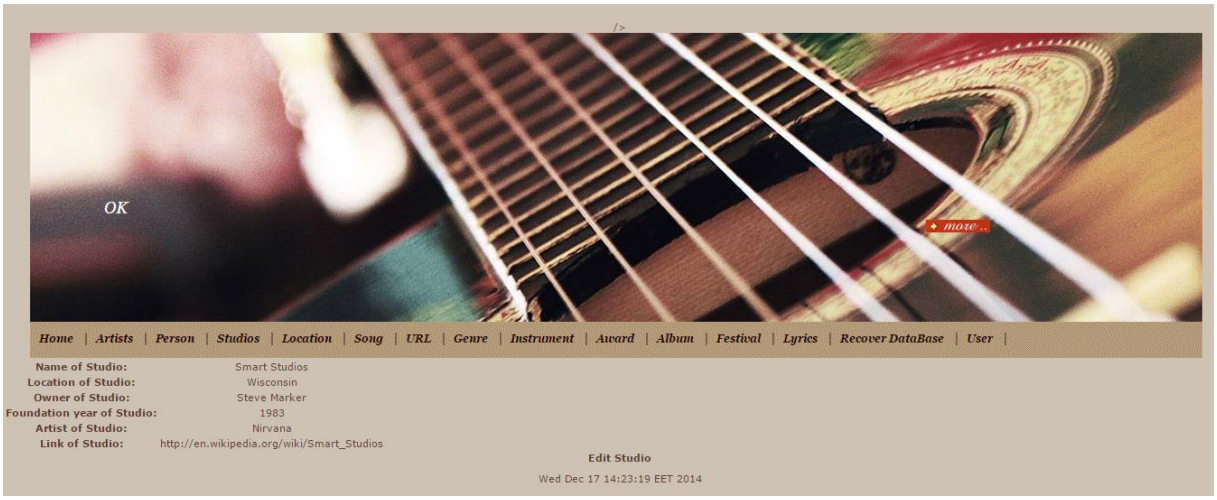


2.3.3. Delete Operation

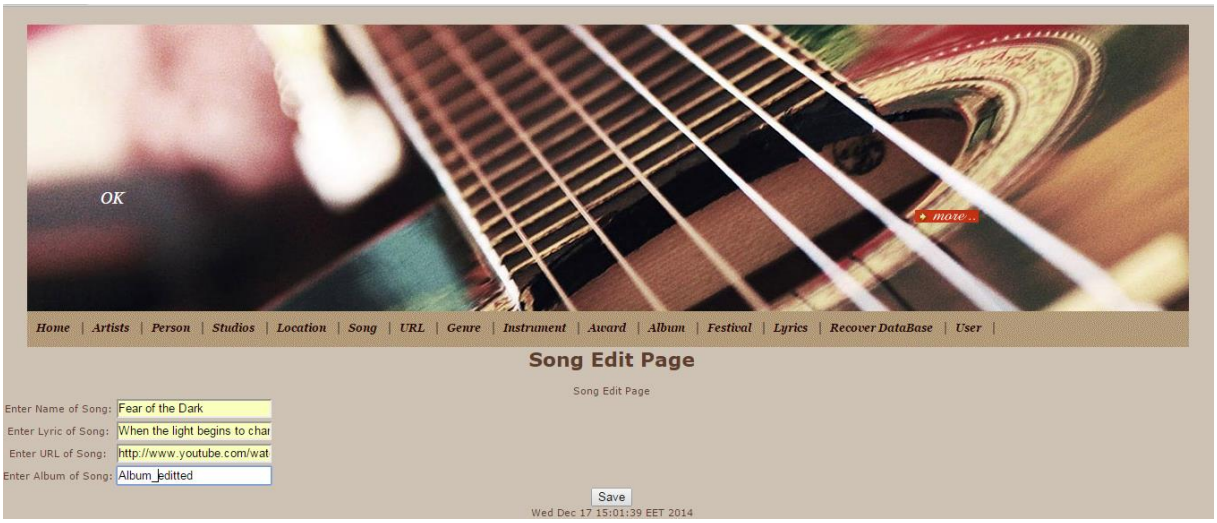
In delete operation, user can delete one row from our sql table. User clicks on check box next to listed compenent which will be deleted. Then there is a “delete” button on bottom. When clicked this button, it is deleted.

2.3.4. Update Operation

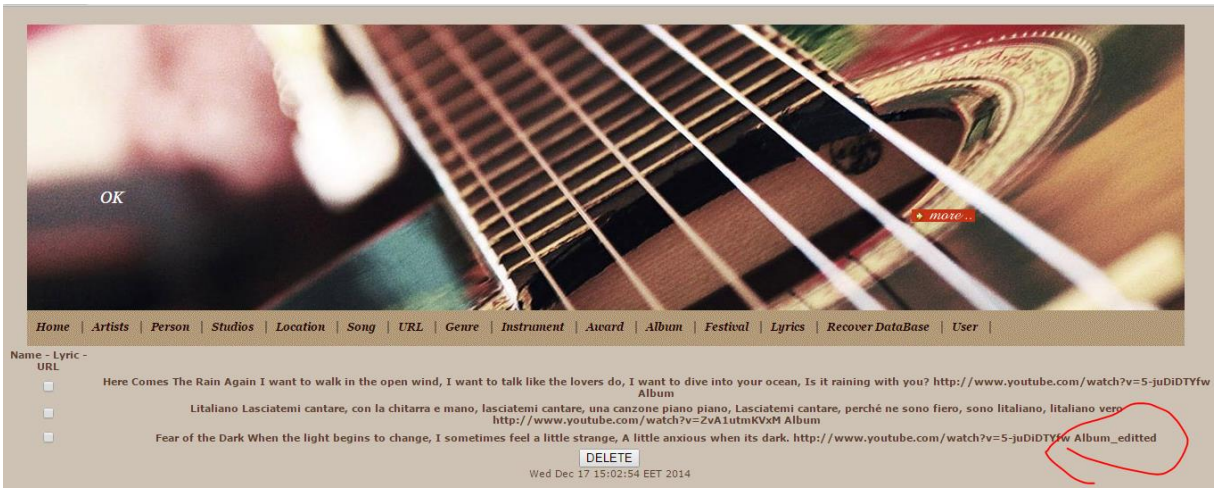
In update operation, user can update one row from our sql table. User clicks on listed compenent, Song Display page is shown on screen.



If user wants to update this informations , “Edit Songs” button is clicked then edit form page is shown on screen

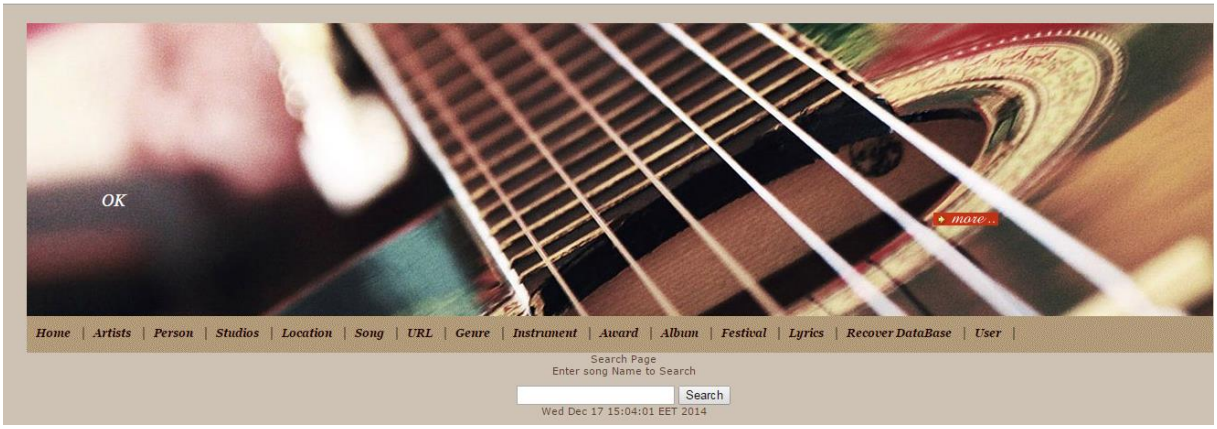


Then user fills parts which will be updated in this form page and click on “Save” button. After clicked on “Save” button, list operation is realized.



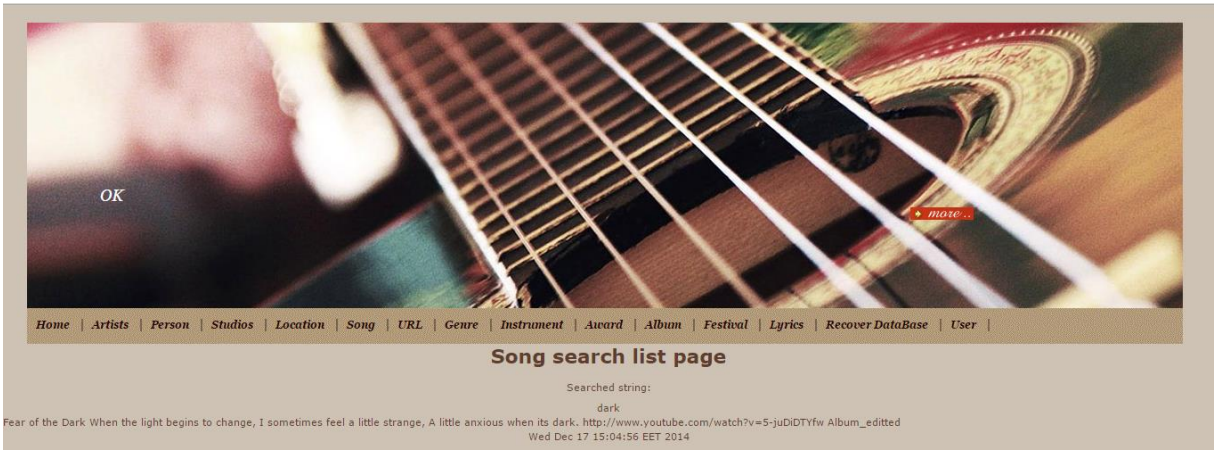
2.3.5. Search Operation

In search operation, any word can be searched in name of studio then all found rows from sql table is listed. To realize this operation, firstly, “Search Song” link is clicked. Then Song Search Page is shown.



Then searched word is fill in blank than click on “Search” button.

Then all found rows from sql table is listed.



3. URL

This class can carry add, list, delete and update operations out. URL class supply the web services of other classes.

3.1. Attributes

Studios class has 5 attributes. Attributes are shown below with their variable types.

private String name;

private String wiki;

private String twitter;

private String facebook;

private String youtube;

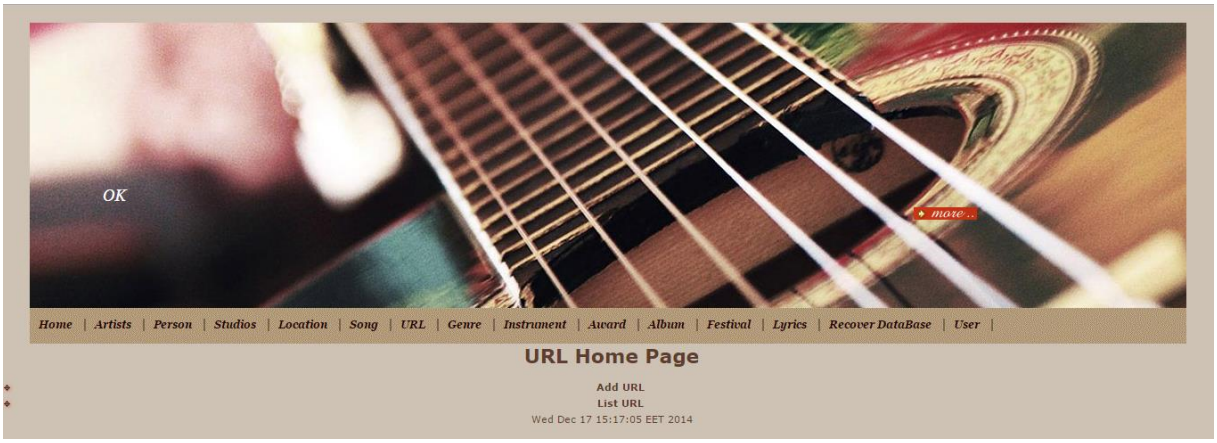
name attribute is used for name of URL. wiki attribute is used for wikipedia URL. twitter attribute is used for twitter URL. facebook attribute is used for facebook URL. youtube attribute is used for youtube URL.

SQL Table

ID	NAME	WIKI	FACEBOOK	TWITTER	YOUTUBE
1	name	http://en.wikipedia.org/...	https://www.facebook.c...	https://twitter.com/jtim...	http://www.youtube.com
2	name1	http://en.wikipedia.org/wi...	-	-	http://www.youtube.com
3	name2	http://en.wikipedia.org/wi...	-	-	http://www.youtube.com

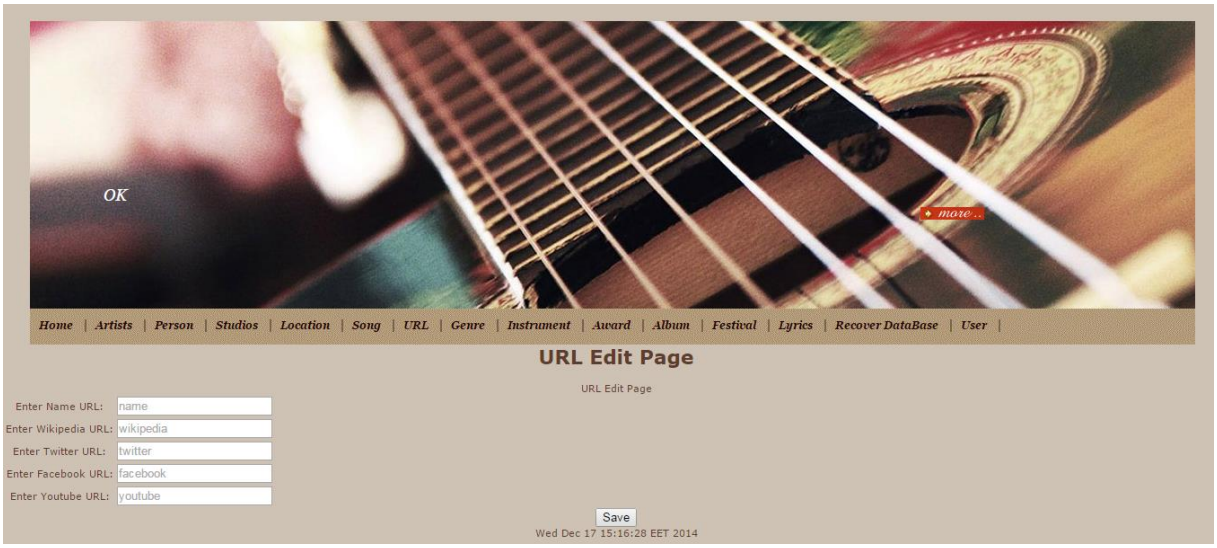
3.2. Operations

For realizing operations, firstly we click on Song tab on our website. It send user to Song Home Page.



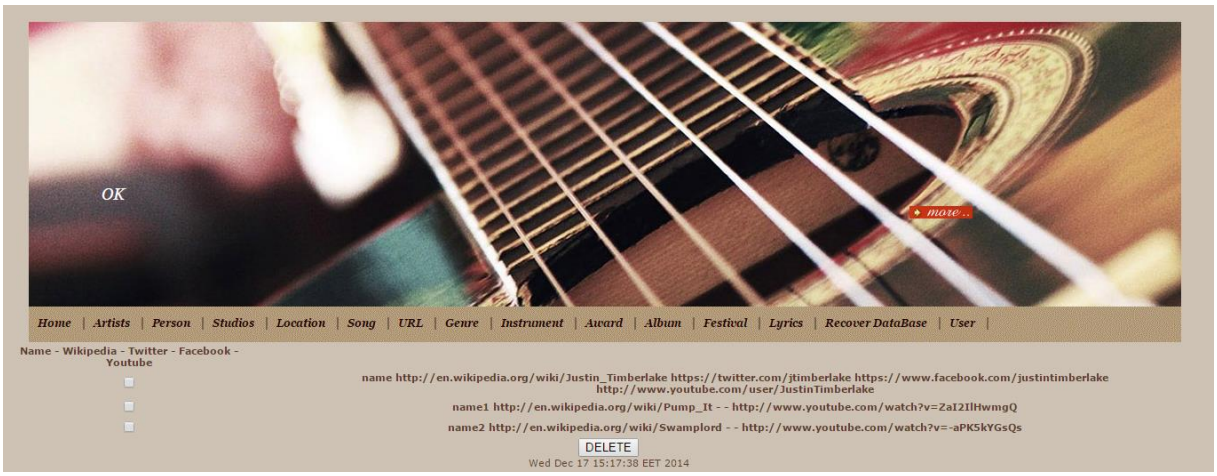
3.2.1. Add Operation

In add operation, we add one row to our sql table. User clicks on Add URL link. Then user fills the blanks in this form page and click on “Save” button. After clicked on “Save” button, list operation is realized.



3.2.2. List Operation

In list operation user can see studios in our sql table. User clicks on “List URL” link.

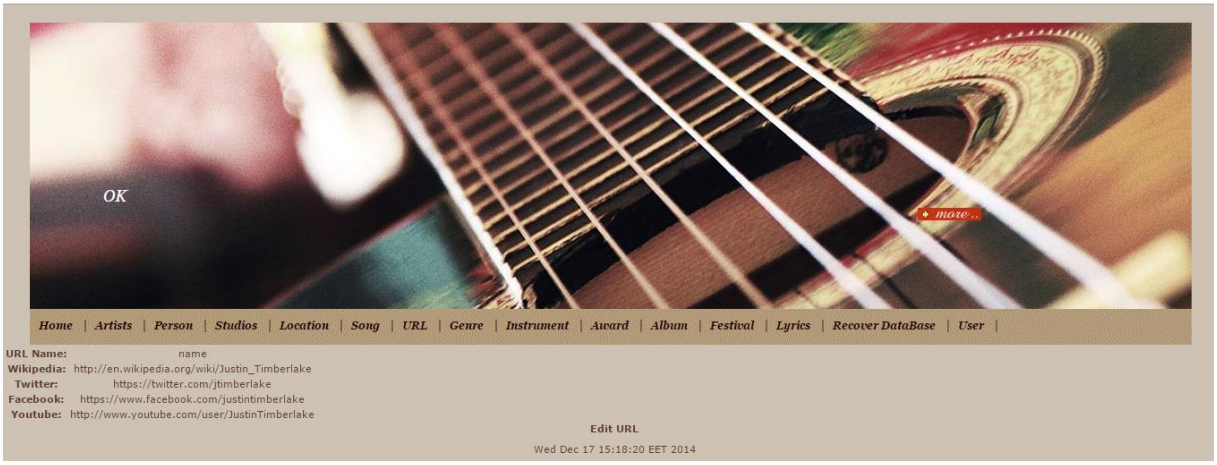


3.2.3. Delete Operation

In delete operation, user can delete one row from our sql table. User clicks on check box next to listed compenent which will be deleted. Then there is a “delete” button on bottom. When clicked this button, it is deleted.

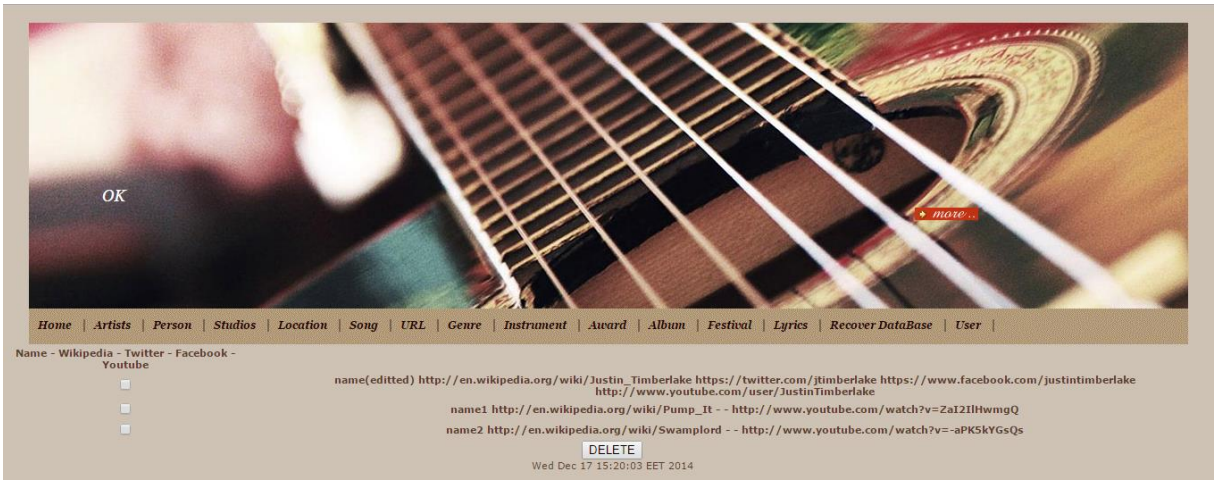
3.2.4. Update Operation

In update operation, user can update one row from our sql table. User clicks on listed compenent, URL Display page is shown on screen.



If user wants to update this informations , “Edit URL” button is clicked then edit form page is shown on screen.

Then user fills parts which will be updated in this form page and click on “Save” button. After clicked on “Save” button, list operation is realized.



4. Person

This class can carry add, list, delete, update and search operations out. Person class holds people which are artists, owner of studios…

4.1. Attributes

Studios class has 6 attributes. Attributes are shown below with their variable types.

private String name;

private String surname;

private Integer age;

private String band;

private String location;

private String instrument;

name attribute is used for name of a person. surname attribute is used for surname of a person. age shows age of a person. band attribute is used to show band of a person. Using band attribute, we will connect person class and artist class. location attribute is used for hometown of a person. Using location attribute, we will connect person class and location class. instrument attribute is used to show instrument which a person play. Using instrument attribute, we will connect person class and instrument class.

4.2. Connections

We use alter table property for connections. Firstly alter table additions is written in init.sql file. After that we supply neccessary changes on StudiosCollectionJDBC file.

init.sql

```
ALTER TABLE PERSONTABLE ADD COLUMN FK_LOCATION INTEGER REFERENCES
LOCATIONTABLE(ID);
ALTER TABLE PERSONTABLE ADD COLUMN FK_INSTRUMENT INTEGER REFERENCES
INSTRUMENTTABLE(ID);
ALTER TABLE PERSONTABLE ADD COLUMN FK_BAND INTEGER REFERENCES ARTISTTABLE(ID);
```

PersonCollectionJDBC

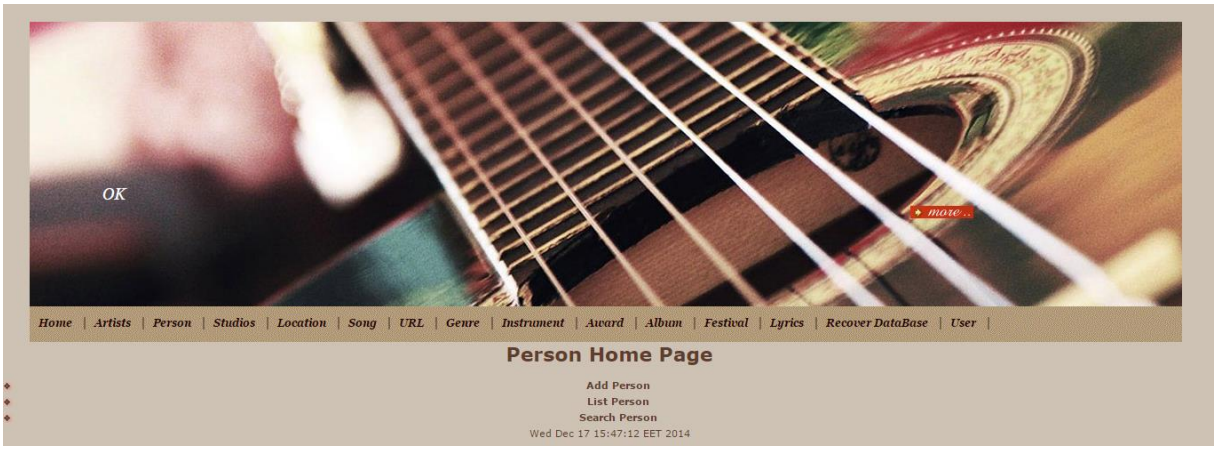
A query is created. This query is used to connect searched id of alter table. For instance, location table and studios table is connected with this query:

```
String query0 = "SELECT ID FROM LOCATIONTABLE WHERE TOWN ='%" +
person.getLocation() + "%'";

As seen in sql code, location attribute of person class is compared with town
attribute in location class, if it is pair, connection is done.
```

ID	NAME	SURNAME	AGE	BAND	LOCATION	INSTRUMENT	FK_BAND	FK_LOCATION	FK_INSTRUME.
1	James	Hetfield	51	Metallica	A	B			
2	Kurt	Kobain	0	Nirvana	C	d			
3	Saul Hudson	SLASH	49	Guns N Roses	E	f			

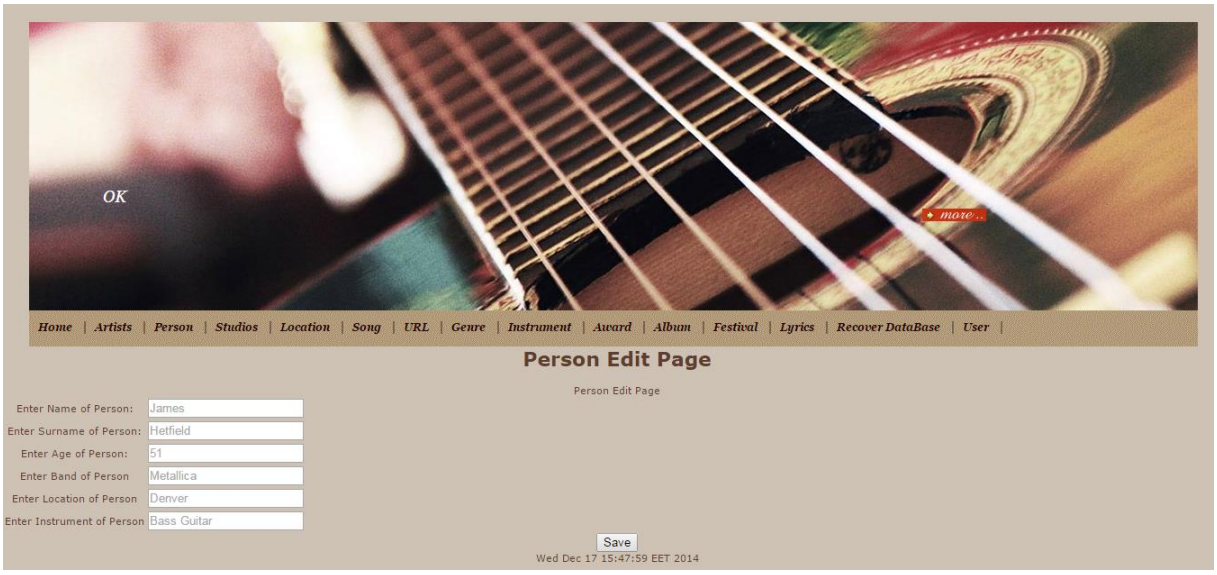
4.3. Operations

For realizing operations, firstly we click on Person tab on our website. It send user to Person Home Page.



4.3.1. Add Operation

In add operation, we add one row to our sql table. User clicks on Add Person link. Then user fills the blanks in this form page and click on “Save” button. After clicked on “Save” button, list operation is realized.



4.3.2. List Operation

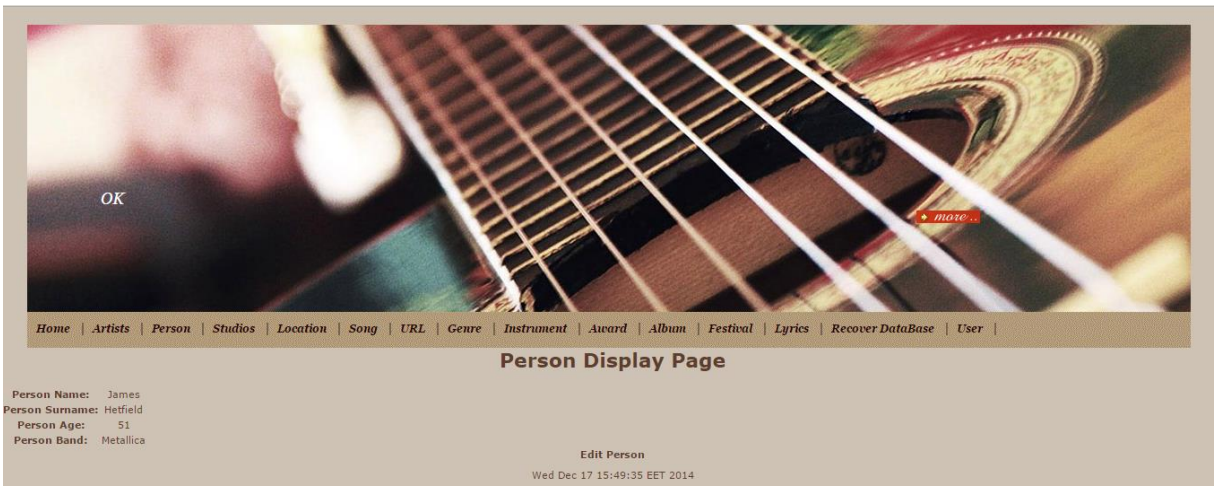
In list operation user can see studios in our sql table. User clicks on “List Person” link.

4.3.3. Delete Operation

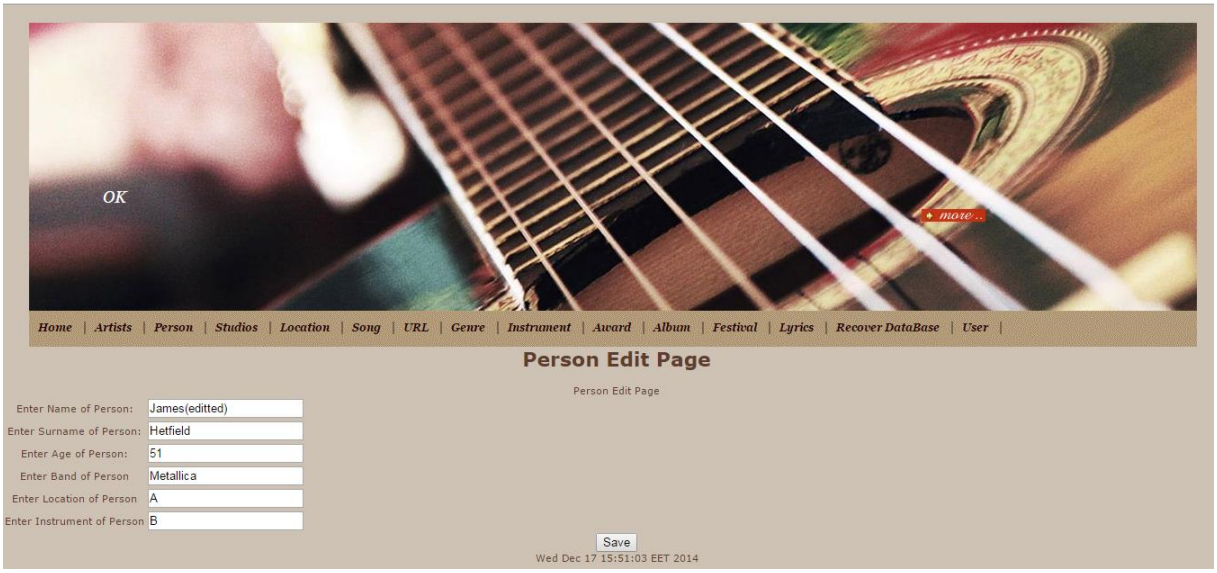
In delete operation, user can delete one row from our sql table. User clicks on check box next to listed compenent which will be deleted. Then there is a “delete” button on bottom. When clicked this button, it is deleted.

4.3.4. Update Operation

In update operation, user can update one row from our sql table. User clicks on listed compenent, Person Display page is shown on screen.



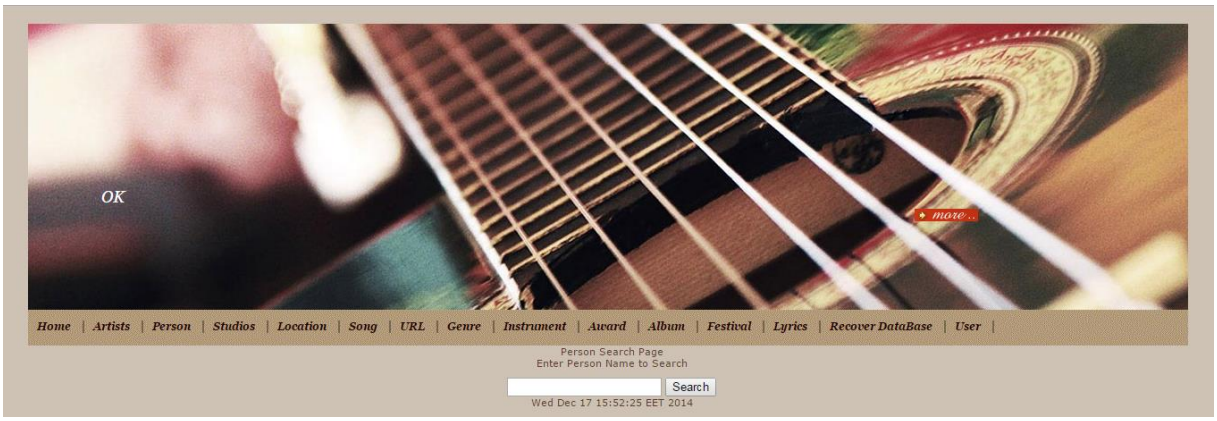
If user wants to update this informations , “Edit Studio” button is clicked then edit form page is shown on screen



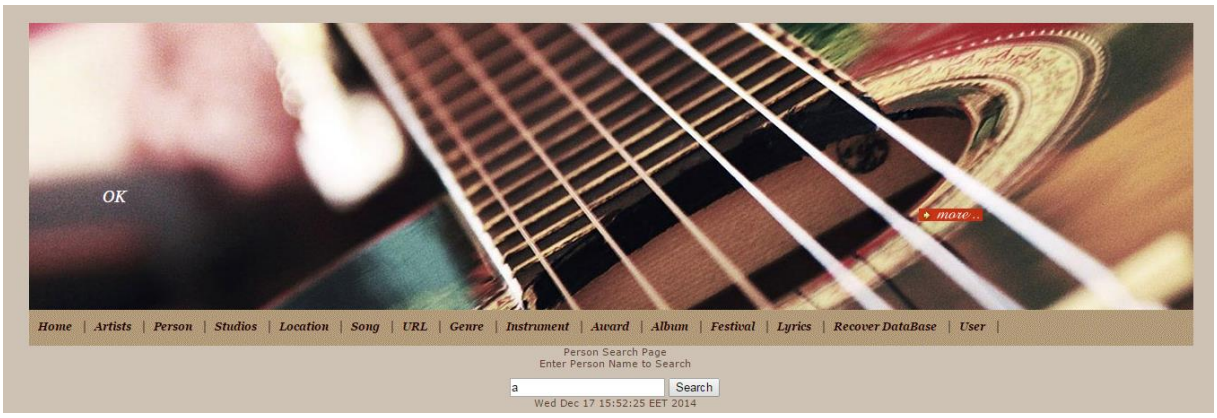
Then user fills parts which will be updated in this form page and click on “Save” button. After clicked on “Save” button, list operation is realized.

4.3.5. Search Operation

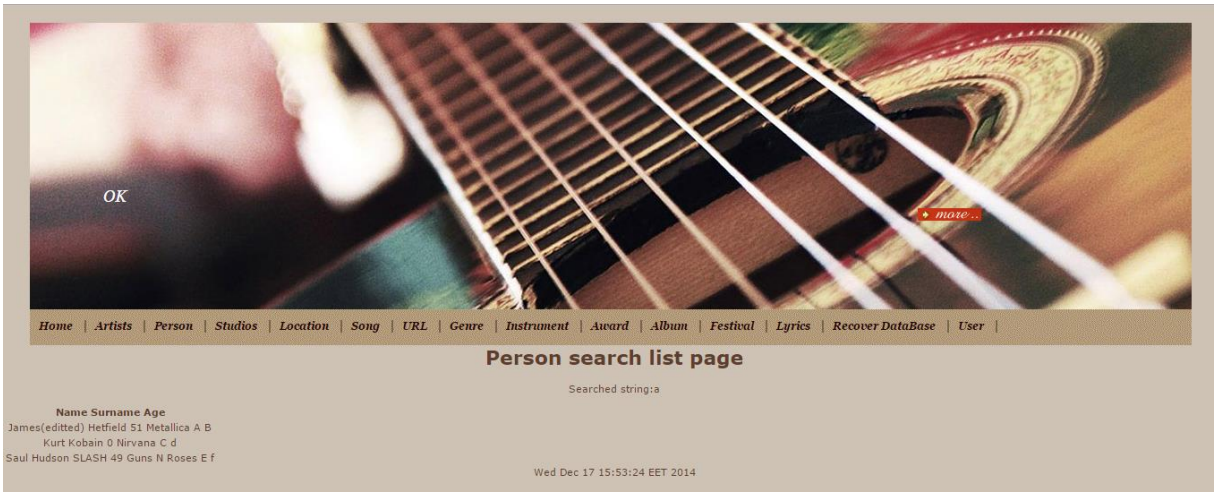
In search operation, any word can be searched in name of person then all found rows from sql table is listed. To realize this operation, firstly, “Search Person” link is clicked. Then Person Search Page is shown.



Then searched word is fill in blank than click on “Search” button.



Then all found rows from sql table is listed.



5. Artist

This class can carry add, list, delete, update and search operations out. Artist class is one of the most important part of our website. Artist informations can be reached.

5.1. Attributes

artist class has 4 attributes. Attributes are shown below with their variable types.

private String _name;

private String _genre;

private Integer _memberNumber;

private String url;

_name attribute is used for name of a artist or band. _genre attribute is used for genre of artist or band. Using _genre attribute, we will connect genre class and artist class. _memberNumber attribute is used to show number of members of a band. url attribute is used for web connections of band or artist. Using url attribute, we will connect artist class and URL class.

5.2. Connections

We use alter table property for connections. Firstly alter table additions is written in init.sql file. After that we supply neccessary changes on ArtistCollectionJDBC file.

init.sql

```
ALTER TABLE ARTISTTABLE ADD COLUMN FK_URL INTEGER REFERENCES URLTABLE(ID);
ALTER TABLE ARTISTTABLE ADD COLUMN FK_GENRE INTEGER REFERENCES GENRETABLE(ID);
```

ArtistCollectionJDBC

A query is created. This query is used to connect searched id of alter table. For instance, album table and song table is connected with this query:

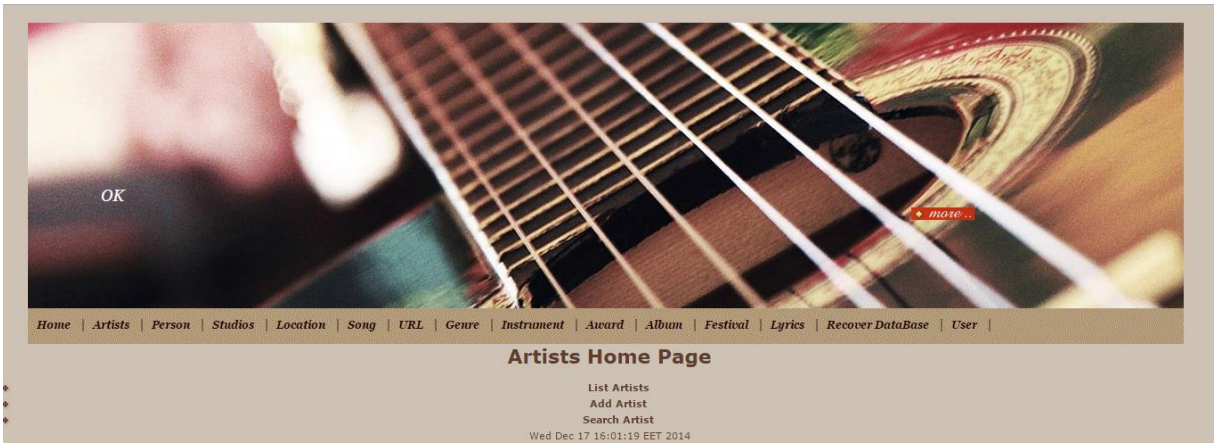
```
String query0 = "SELECT ID FROM GENRETABLE WHERE NAME LIKE '%"+ song.getGenre() +
"%'";
```

```
As seen in sql code, genre attribute of Artist class is compared with name
attribute in genre class, if it is pair, connection is done.
```

ID	NAME	GENRE	MEMBERNUMBER	NAME2	SURNAME2	URL	FK_URL	FK_GENRE
1	Metallica	Heavy Metal	4					
2	Nirvana	Alternative Rock	5					
3	Guns N Roses	Hard Rock	5					

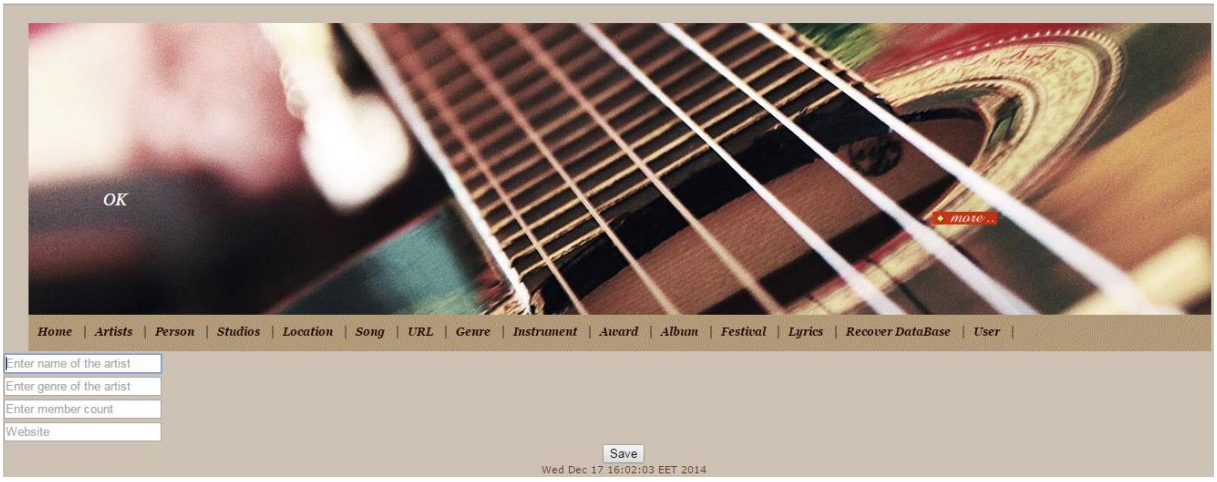
5.3. Operations

For realizing operations, firstly we click on Song tab on our website. It send user to Artist Home Page.



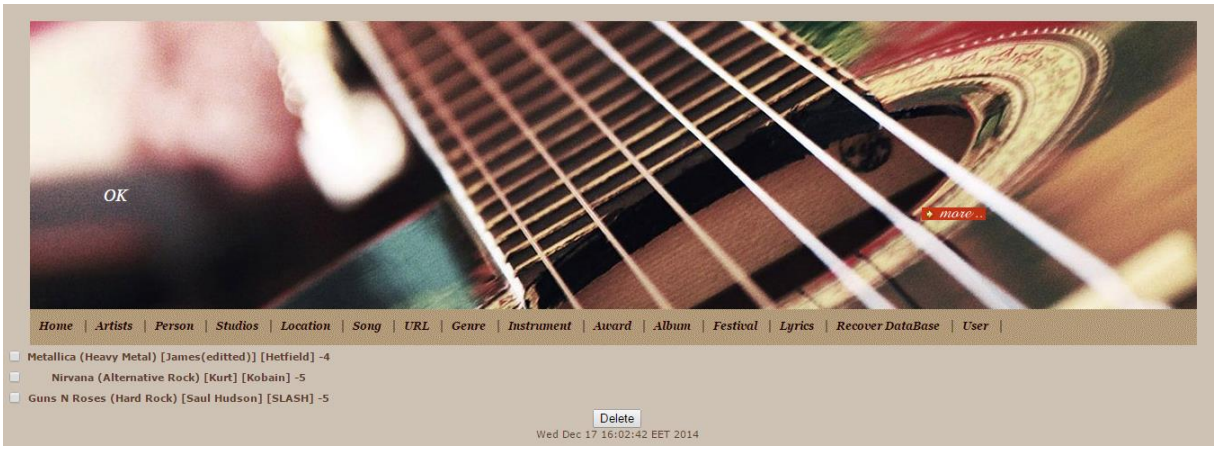
5.3.1. Add Operation

In add operation, we add one row to our sql table. User clicks on Add Artist link. Then user fills the blanks in this form page and click on “Save” button. After clicked on “Save” button, list operation is realized.



5.3.2. List Operation

In list operation user can see studios in our sql table. User clicks on “List Artist” link.

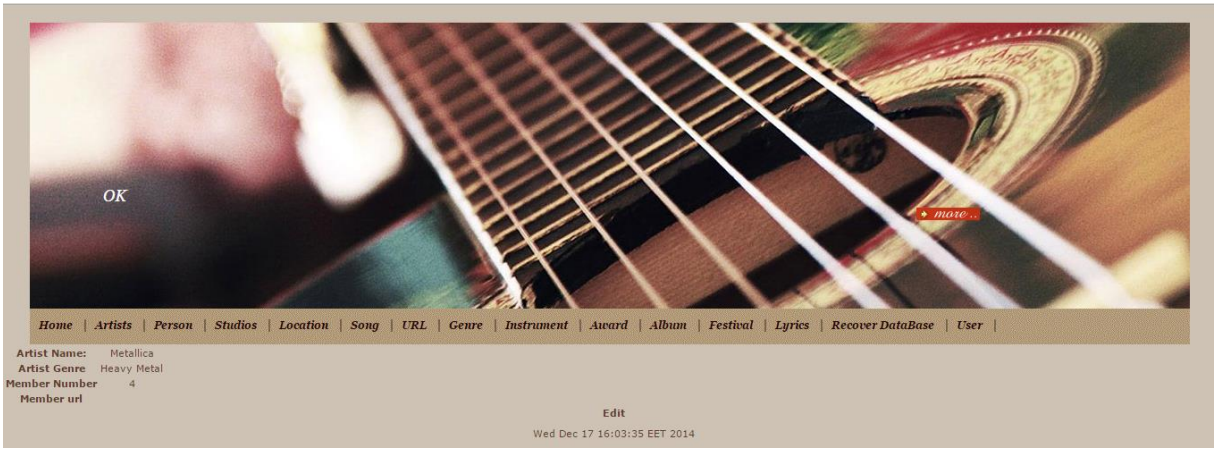


5.3.3. Delete Operation

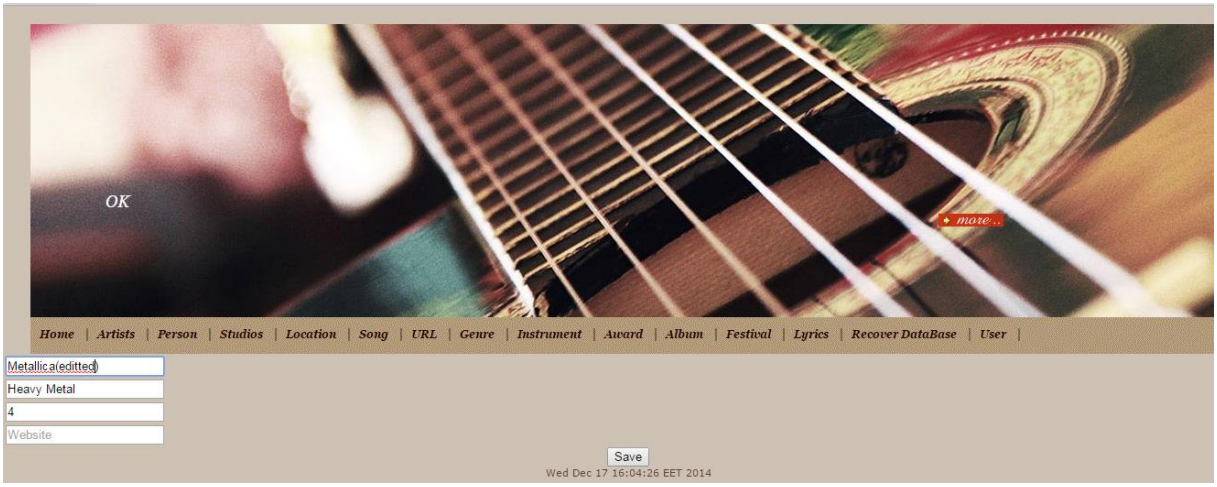
In delete operation, user can delete one row from our sql table. User clicks on check box next to listed compenent which will be deleted. Then there is a “delete” button on bottom. When clicked this button, it is deleted.

5.3.4. Update Operation

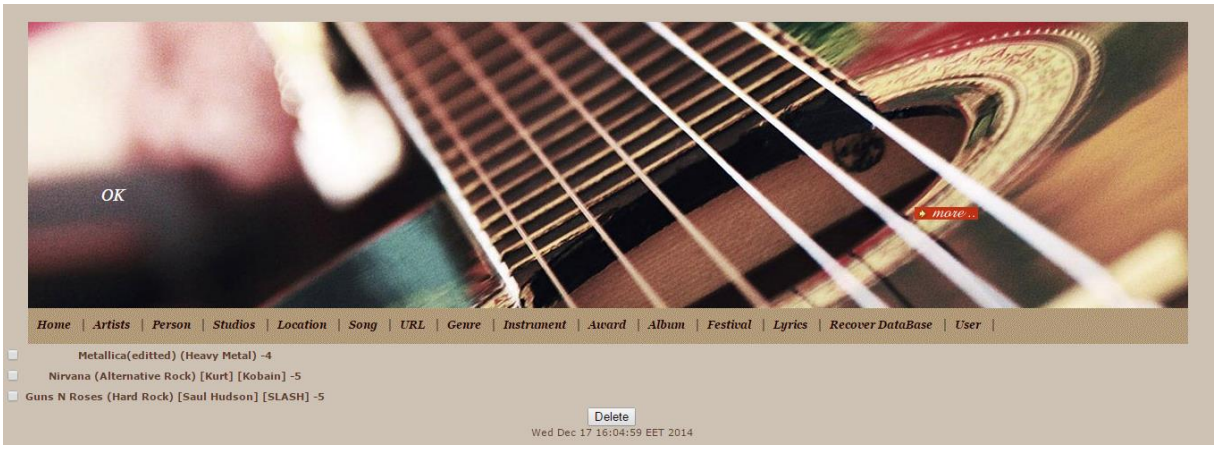
In update operation, user can update one row from our sql table. User clicks on listed compenent, Artists Display page is shown on screen.



If user wants to update this informations , “Edit” button is clicked then edit form page is shown on screen

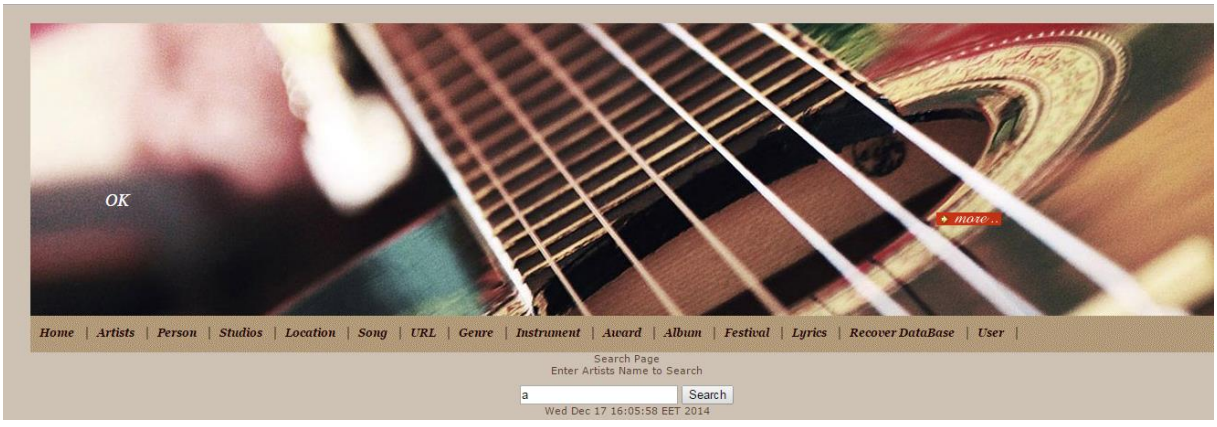


Then user fills parts which will be updated in this form page and click on “Save” button. After clicked on “Save” button, list operation is realized.

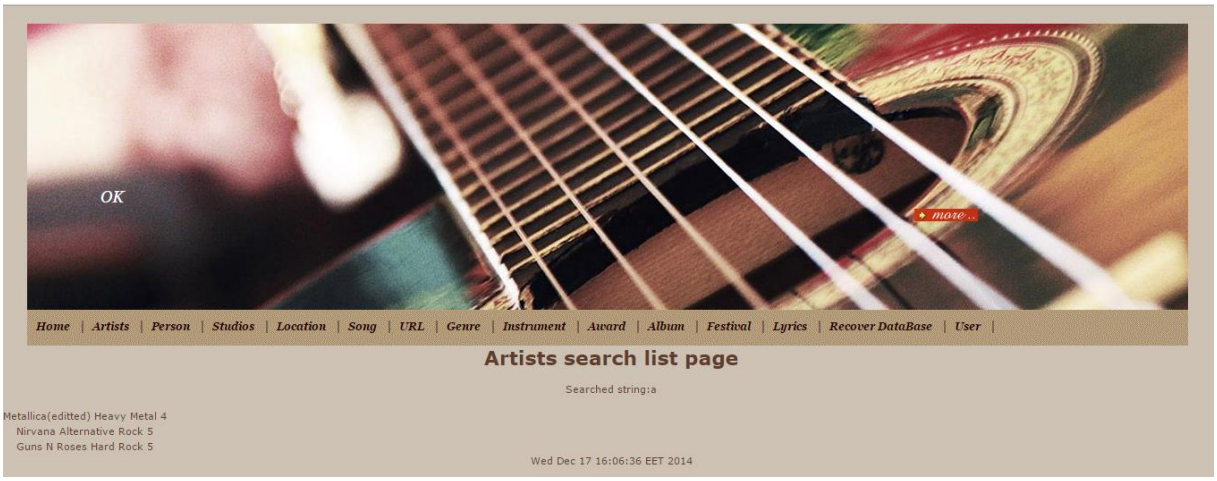


5.3.5. Search Operation

In search operation, any word can be searched in name of person then all found rows from sql table is listed. To realize this operation, firstly, “Search Artist” link is clicked. Then Artist Search Page is shown.



Then searched word is fill in blank than click on “Search” button. Then all found rows from sql table is listed.



6. Location

This class can carry add, list, delete and update operations out. Location class is written for connections of other classes.

6.1. Attributes

Studios class has 3 attributes. Attributes are shown below with their variable types.

String continent;

String country;

String town;

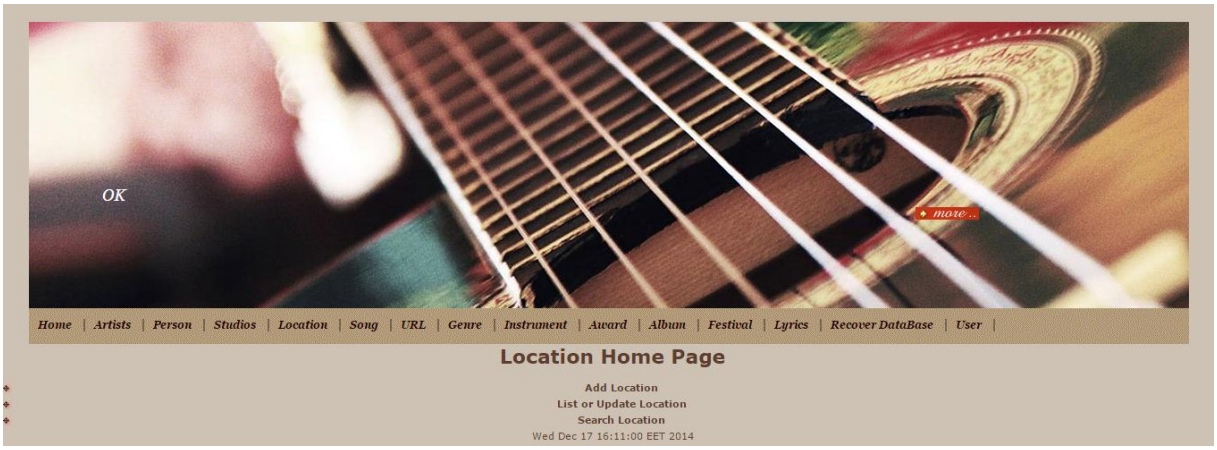
continent attribute is used for continent of a location. country attribute is used for country of a location. town attribute is used for town of a location.

SQL Table

ID	CONTINENT	COUNTRY	TOWN
1	Scandinavia	Sweden	Tumba
2	United States	California	Glendale

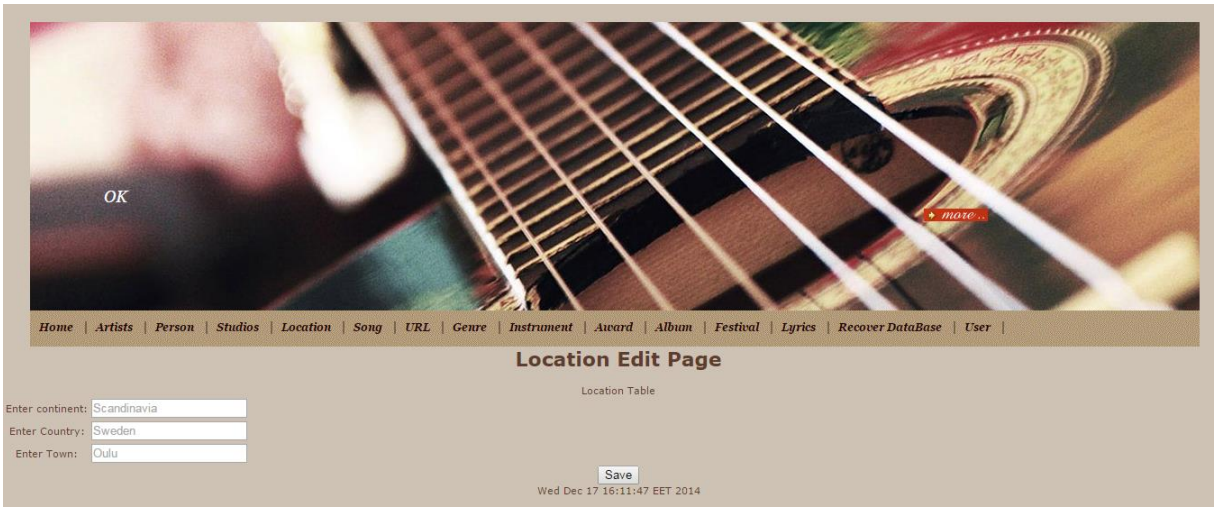
6.2. Operations

For realizing operations, firstly we click on Location tab on our website. It send user to Location Home Page.



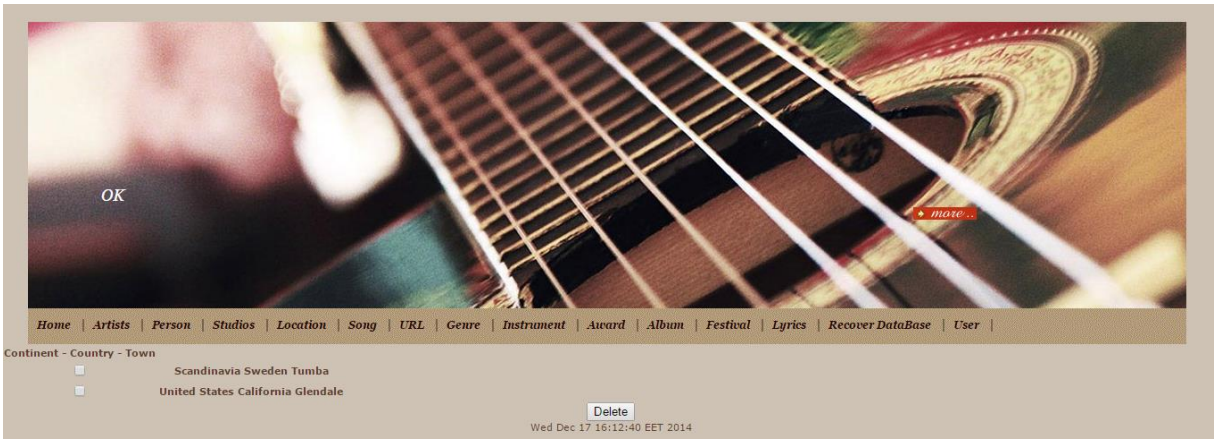
6.2.1. Add Operation

In add operation, we add one row to our sql table. User clicks on Add Location link. Then user fills the blanks in this form page and click on “Save” button. After clicked on “Save” button, list operation is realized.



6.2.2. List Operation

In list operation user can see studios in our sql table. User clicks on “List or Update Location” link.

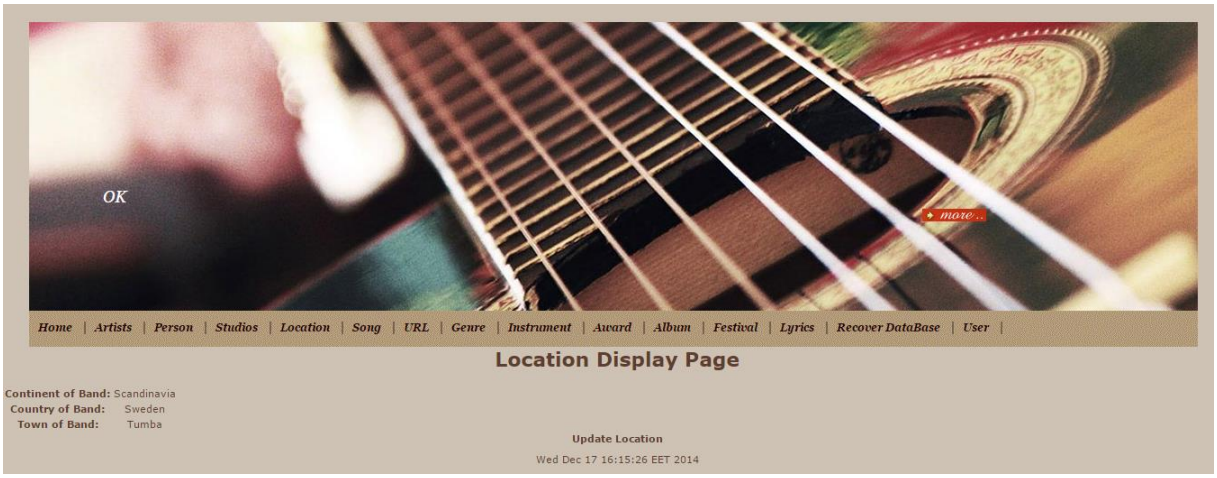


6.2.3. Delete Operation

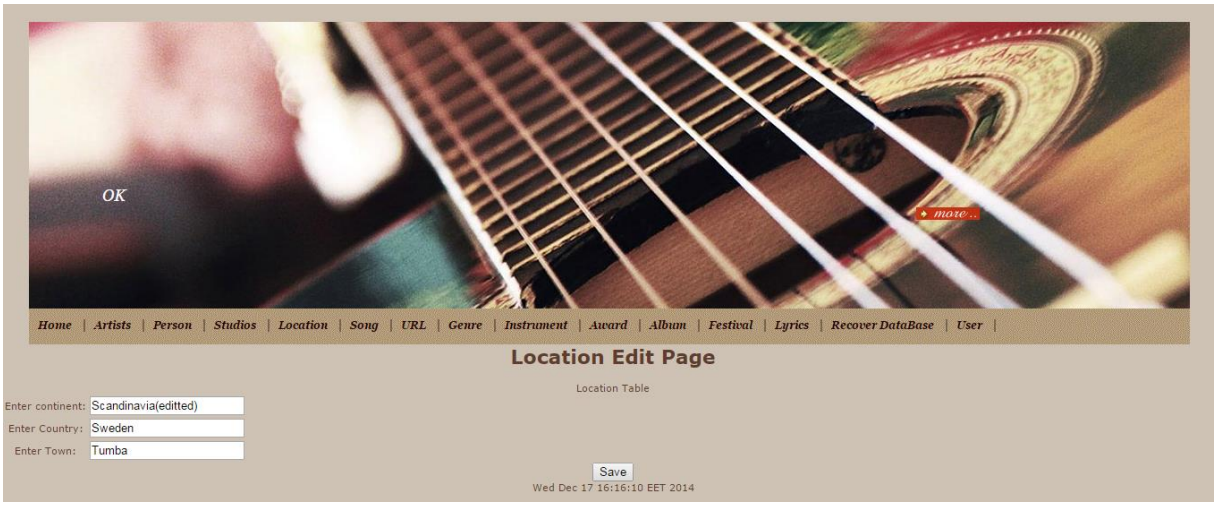
In delete operation, user can delete one row from our sql table. User clicks on check box next to listed compenent which will be deleted. Then there is a “delete” button on bottom. When clicked this button, it is deleted.

6.2.4. Update Operation

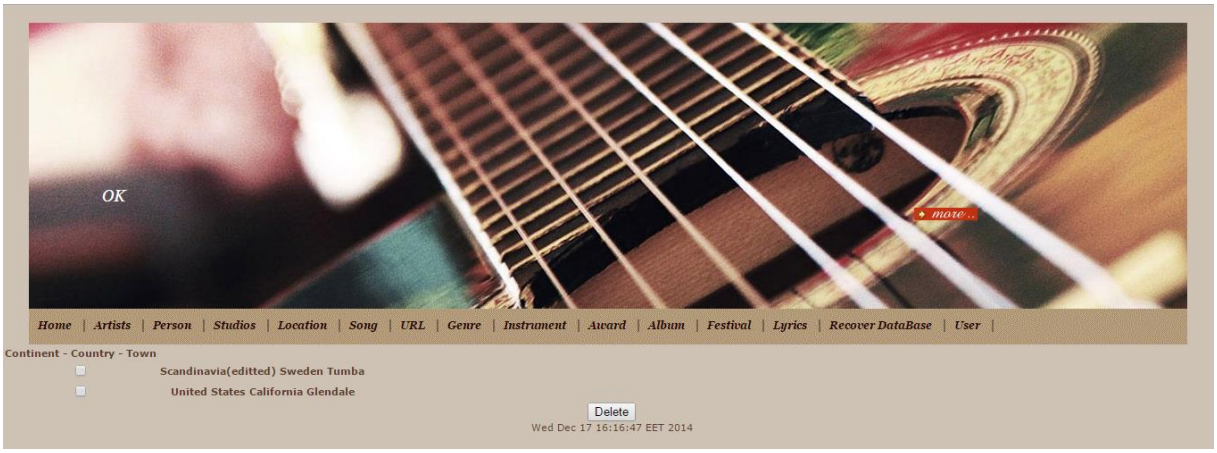
In update operation, user can update one row from our sql table. User clicks on listed compenent, Location Display page is shown on screen.



If user wants to update this informations , “Update Location” button is clicked then edit form page is shown on screen.

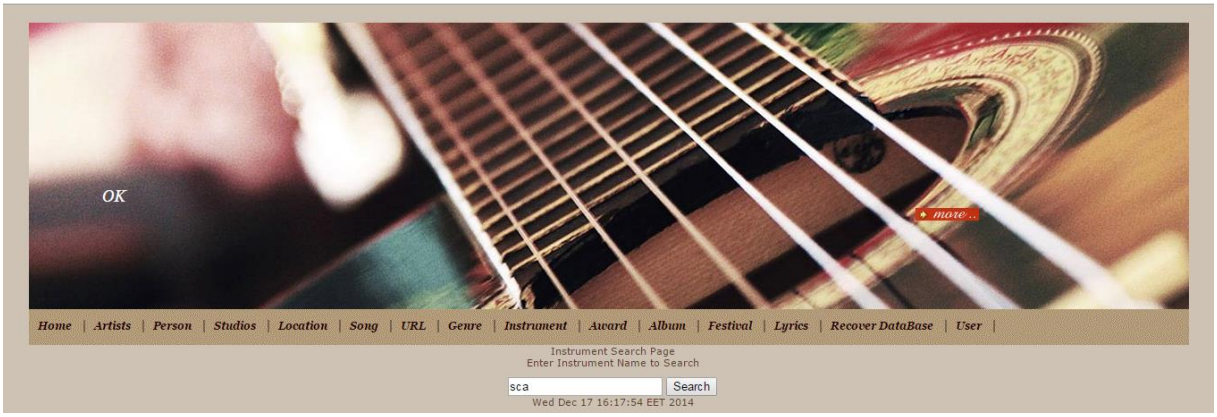


Then user fills parts which will be updated in this form page and click on “Save” button. After clicked on “Save” button, list operation is realized.



6.2.5. Search Operation

In search operation, any word can be searched in name of person then all found rows from sql table is listed. To realize this operation, firstly, “Search Location” link is clicked. Then Location Search Page is shown.



Then searched word is fill in blank than click on “Search” button. Then all found rows from sql table is listed.

7. Award

7.1. Attributes

Award class has 6 attributes. Attributes are shown below with their variable types

```
private Integer _id = null;

private String name;
private String location;
private String type;
```

```
    private Integer year;
    private String artist;
    private String searchWord;
```

name attribute is used for name of a award. type attribute is used for type of award. location attribute is used for giving location of award. Using location attribute, we will connect music Award class and location class. year attribute is used to show year of award. searchWord attribute is used for searching methods of this class. artist attribute is used to show artist or band which is winner of award. Using artist attribute, we will connect artist class and award class.

7.2. Connections

We use alter table property for connections. Firstly alter table additions is written in init.sql file. After that we supply neccessary changes on AwardCollectionJDBC file.

init.sql

```
CREATE TABLE AWARDTABLE (ID INTEGER PRIMARY KEY AUTOINCREMENT,NAME VARCHAR(40) NOT NULL,LOCATION VARCHAR(40),TYPE VARCHAR(40) NOT NULL,YEAR INTEGER,ARTIST VARCHAR(40));


ALTER TABLE GENRETABLE ADD COLUMN FK_AWARD INTEGER REFERENCES AWARDTABLE(ID);

INSERT INTO AWARDTABLE (NAME, LOCATION, TYPE, YEAR, ARTIST) VALUES('KralTv Music Awards', 'Turkey', 'everything', '2014', 'Kurt Cobain');
```

AwardCollectionJDBC

A query is created. This query is used to connect searched id of alter table. For instance, location table and Award table is connected with this query:

```
String query0 = "SELECT ID FROM ARTISTTABLE WHERE NAME ='"
                        + award.getArtist() + "'";
```

Main query of insert method is created.

```
String query = "INSERT INTO AWARDTABLE (NAME, LOCATION, TYPE, YEAR, ARTIST, FK_ARTIST) VALUES (?, ?, ?, ?, ?, ?)";
```

Main query of delete method is created.

```
String query = "DELETE FROM AWARDTABLE WHERE (ID = ?)";
```

Main query of update method is created.

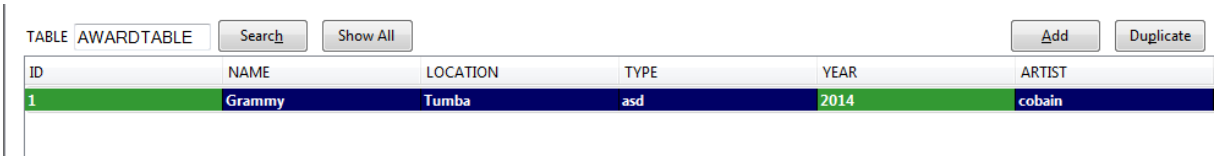
```
String query0 = "SELECT ID FROM ARTISTTABLE WHERE NAME ='"
                        + award.getArtist() + "'";

String query = "UPDATE AWARDTABLE SET NAME=?, LOCATION=?, TYPE=?, YEAR=?, ARTIST=?, FK_ARTIST=? WHERE (ID=?)";
```

Main query of list method is created.

```
String query = "SELECT ID, NAME, LOCATION, TYPE, YEAR, ARTIST FROM AWARDTABLE";
```

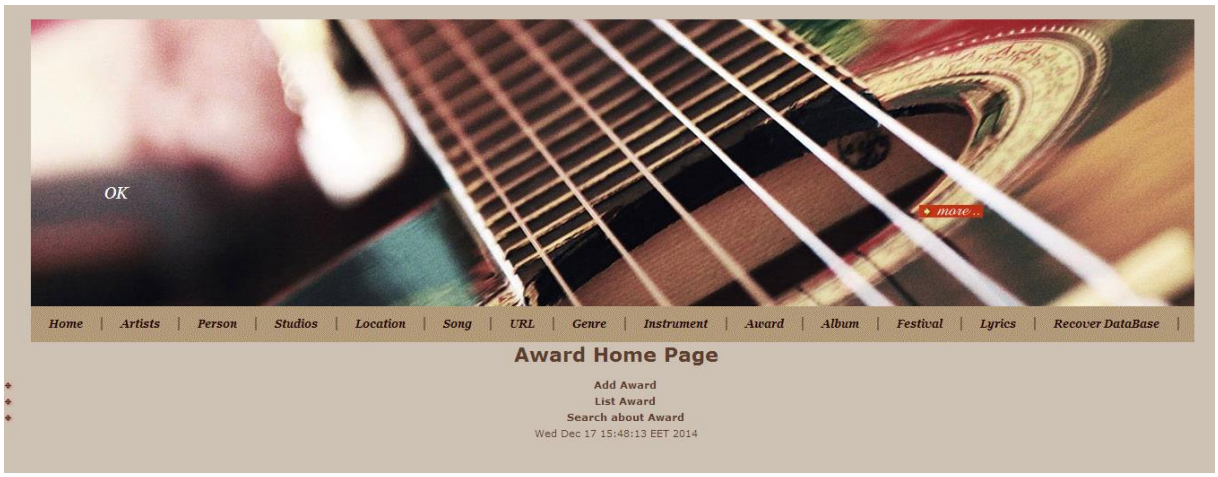
Main query of list method is created.

```
String query = "SELECT ID, NAME, LOCATION, TYPE, YEAR, ARTIST FROM AWARDTABLE
WHERE NAME LIKE '%"
             + aranacak
             + "%'OR LOCATION LIKE '%"
             + aranacak
             + "%' OR TYPE LIKE '%"
             + aranacak
             + "%'OR YEAR LIKE '%"
             + aranacak + "%'OR ARTIST LIKE '%" + aranacak + "%'";
```



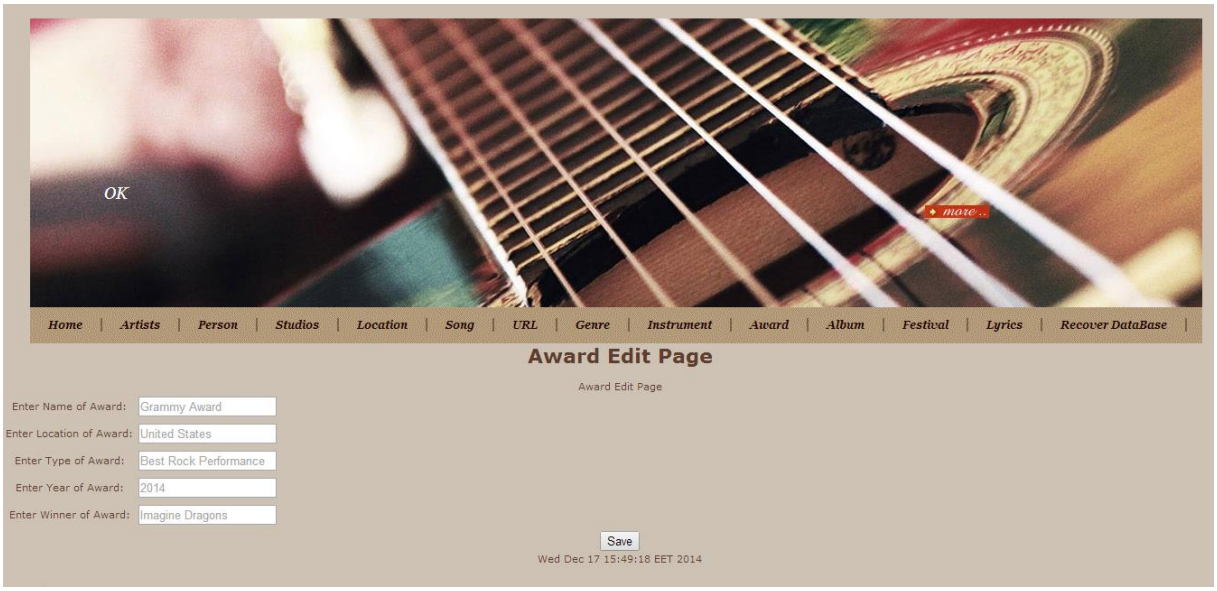
7.3. Operations

For realizing operations, firstly we click on Award tab on our website. It send user to Award Home Page.



7.3.1. Add Operation

In add operation, we add one row to our sql table. User clicks on Add Award link. Then user fills the blanks in this form page and click on “Save” button. After clicked on “Save” button, list operation is realized.

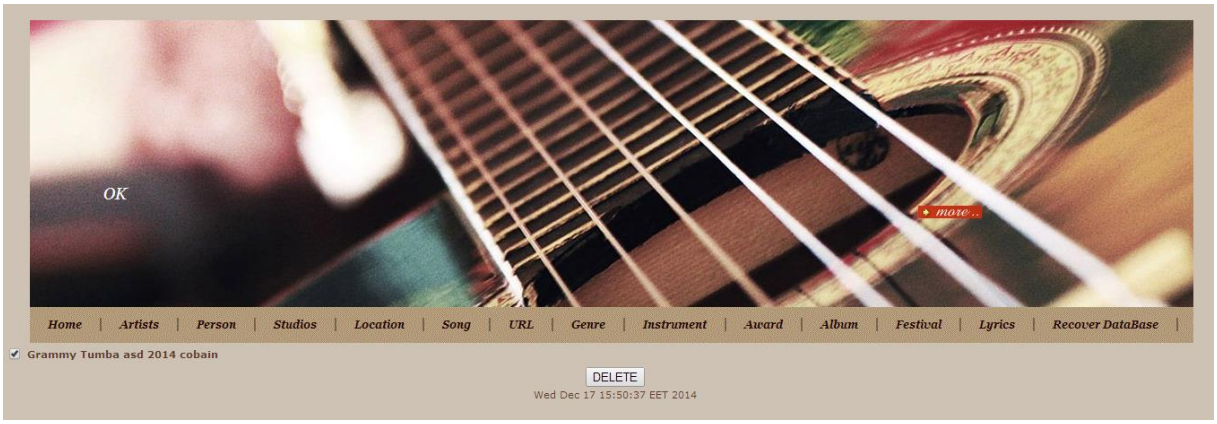


7.3.2. List Operation

In list operation user can see all Awards in our sql table. User clicks on “List Award” link.

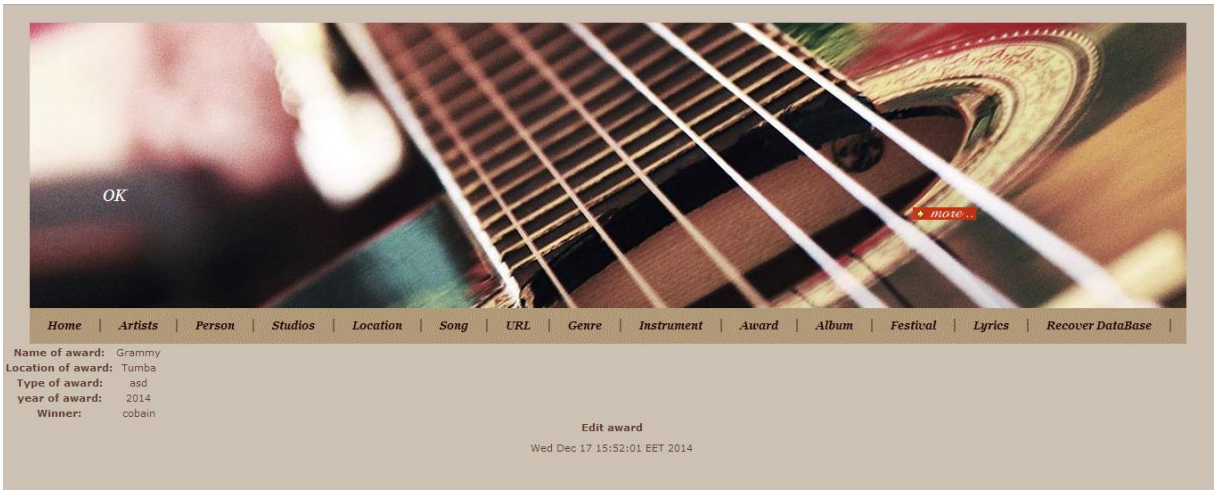
7.3.3. Delete Operation

In delete operation, user can delete one row from our sql table. User clicks on check box next to listed compenent which will be deleted. Then there is a “delete” button on bottom. When clicked this button, it is deleted.

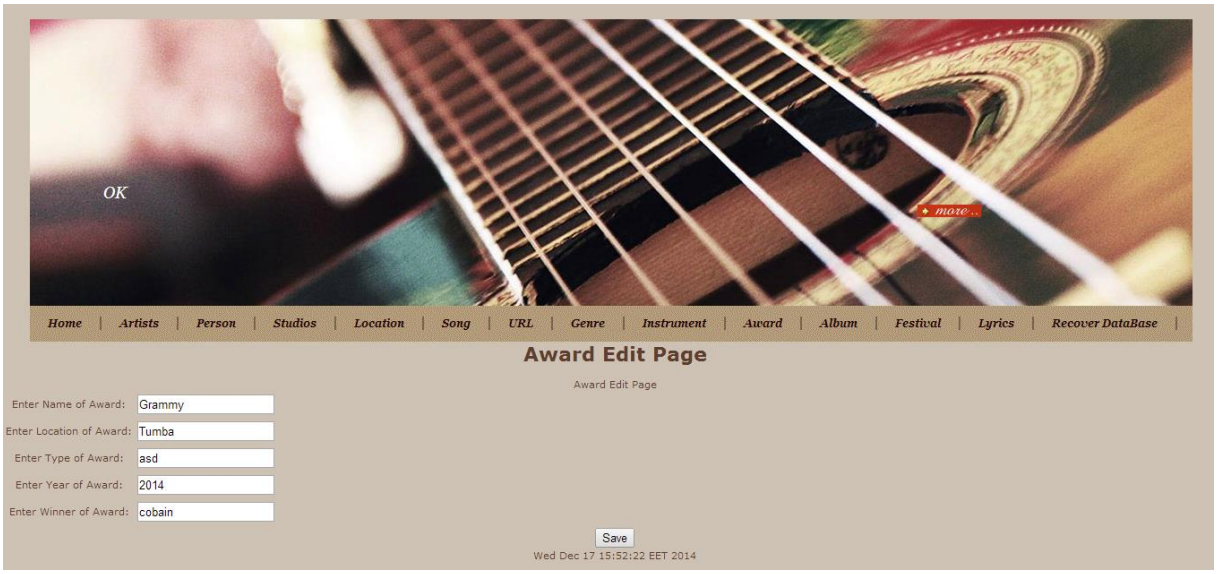


7.3.4. Update Operation

In update operation, user can update one row from our sql table. User clicks on listed compenent, Award Display page is shown on screen.

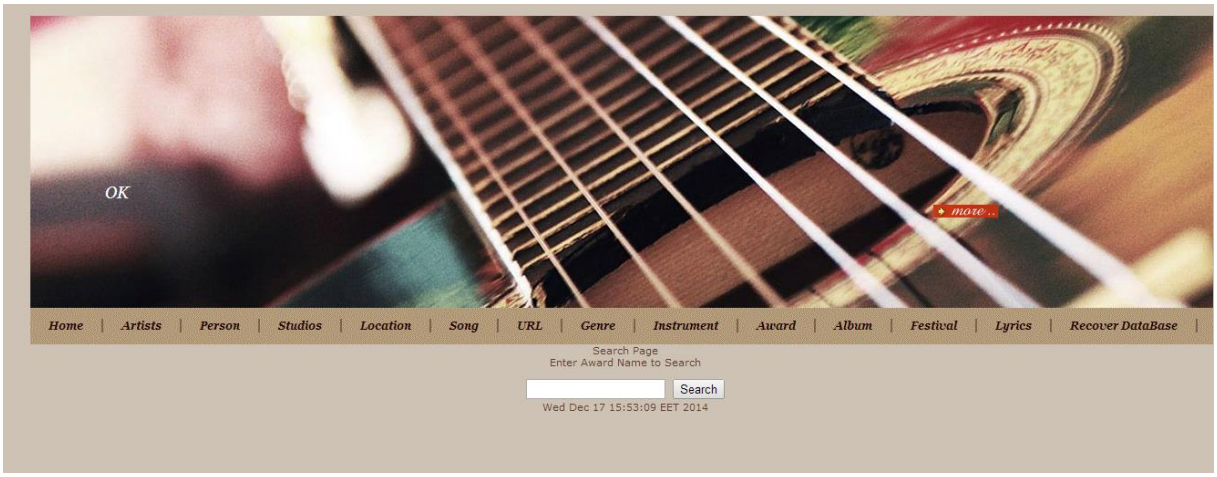


If user wants to update this informations , “Edit Award” button is clicked then edit form page is shown on screen

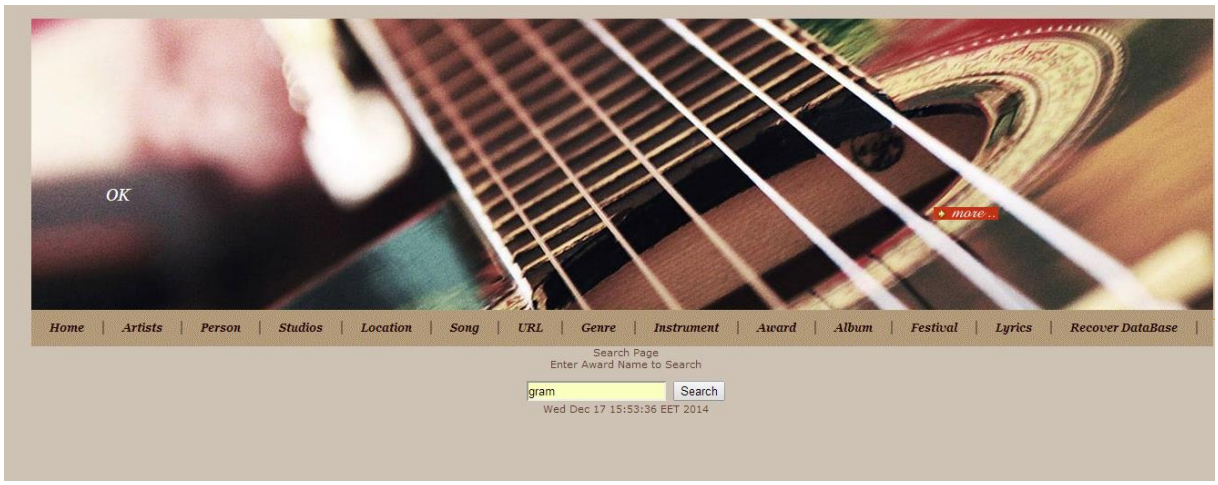


7.3.5. Search Operation

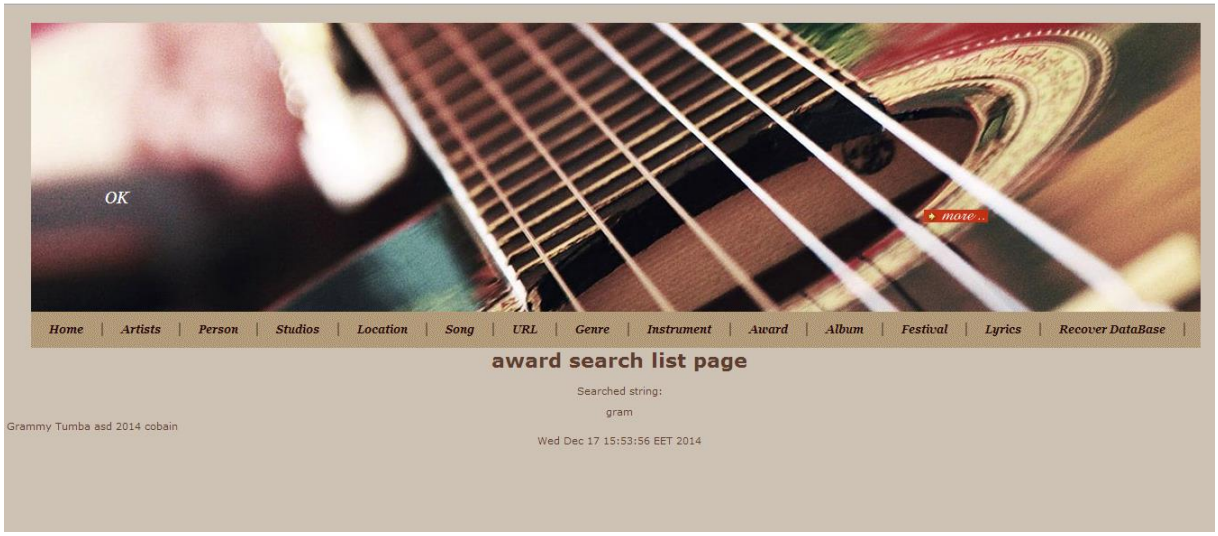
In search operation, any word can be searched in name of studio then all found rows from sql table is listed. To realize this operation, firstly, “Search Award” link is clicked. Then Award Search Page is shown.



Then searched word is fill in blank than click on “Search” button.



Then all found rows from sql table is listed.



8. Instrument

This class is used for describing of musical instrument which is an instrument created or adapted to make musical sounds. In principle, any object that produces sound can be a musical instrument—it is through purpose that the object becomes a musical instrument.

8.1. Attributes

Instrument class has 7 attributes. Attributes are shown below with their variable types.

```
    private String name;
    private String type;
    private String range;
    private String location;
    private String location1;
    private String location2;
    private String searchWord;
```

name attribute is used for name of a musical instrument. type attribute is used for type of musical instrument. location attribute is used for hometown of musical location. Using location attribute, we will connect music Instrument class and location class. range attribute is used to indicate range of musical instrument. searchWord attribute is used for searching methods of this class.

8.2. Connections

We use alter table property for connections. Firstly alter table additions is written in init.sql file. After that we supply neccessary changes on InstrumentCollectionJDBC file.

init.sql

```
CREATE TABLE INSTRUMENTTABLE (ID INTEGER PRIMARY KEY AUTOINCREMENT,NAME
VARCHAR(40) NOT NULL,TYPE VARCHAR(40) NOT NULL,RANGE VARCHAR(55) NOT NULL,LOCATION
VARCHAR(55) NOT NULL , LOCATION1 VARCHAR(55), LOCATION2 VARCHAR(55) );


ALTER TABLE INSTRUMENTTABLE ADD COLUMN FK_LOCATION INTEGER REFERENCES
LOCATIONTABLE(ID);

INSERT INTO INSTRUMENTTABLE (NAME, TYPE, RANGE, LOCATION) VALUES('kemence',
'string', 'soprano','Trabzon');
```

InstrumentCollectionJDBC

A query is created. This query is used to connect searched id of alter table. For instance, location table and instrument table is connected with this query:

```
String query0 = "SELECT ID FROM LOCATIONTABLE WHERE TOWN LIKE '%"
                          + instrument.getLocation() + "%'";
```

Main query of insert method is created.

```
String query = "INSERT INTO INSTRUMENTTABLE (NAME, TYPE, RANGE, LOCATION,
FK_LOCATION) VALUES (?, ?, ?, ?, ?)";
```

Main query of delete method is created.

```
String query = "DELETE FROM INSTRUMENTTABLE WHERE (ID = ?)";
```

Main query of update method is created.

```
String query0 = "SELECT ID FROM LOCATIONTABLE WHERE TOWN LIKE '%"
```

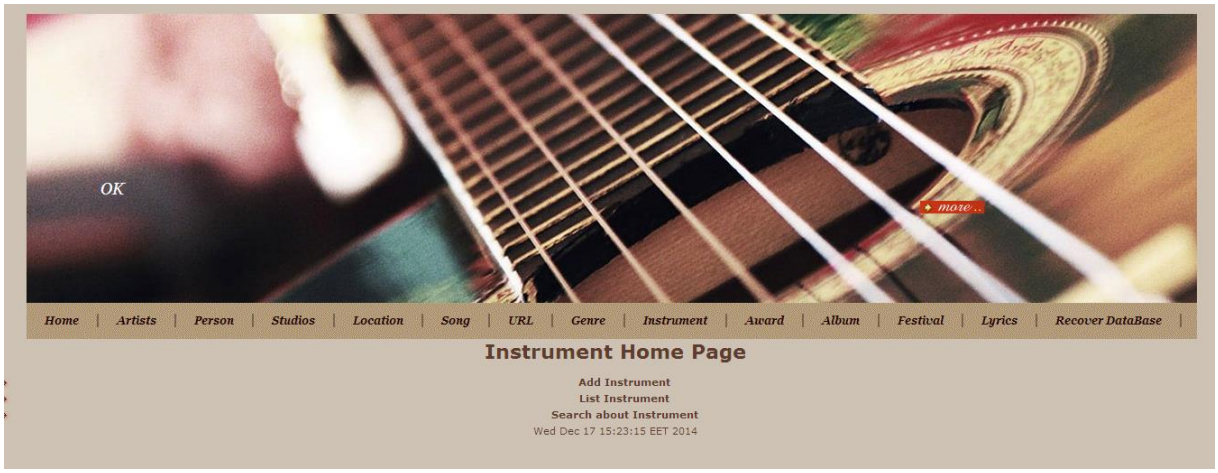
```
                        + instrument.getLocation() + "%'";

String query = "UPDATE INSTRUMENTTABLE SET NAME=?, TYPE=?, RANGE=?, LOCATION=?,
FK_LOCATION=? WHERE (ID=?)";
```

Main query of delete method is created.

```
String query2 = "SELECT  CONTINENT ,COUNTRY FROM LOCATIONTABLE WHERE COUNTRY = '"
                        + instrument.getLocation() + "'";

String query = "SELECT ID, NAME, TYPE, RANGE, LOCATION FROM INSTRUMENTTABLE";
```

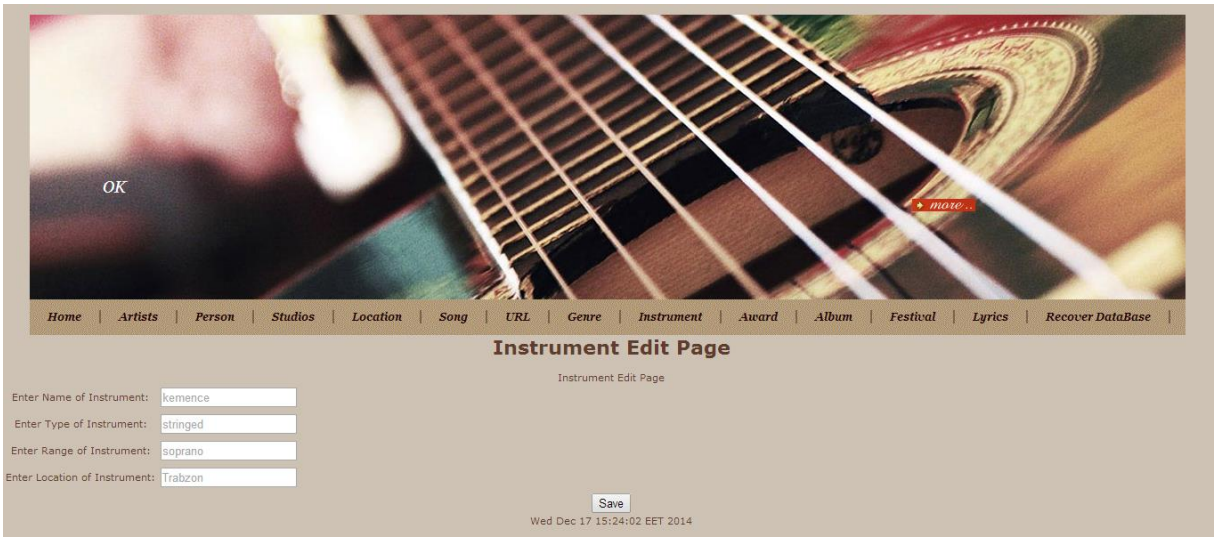
8.3. Operations

For realizing operations, firstly we click on Instrument tab on our website. It send user to Instrument Home Page.



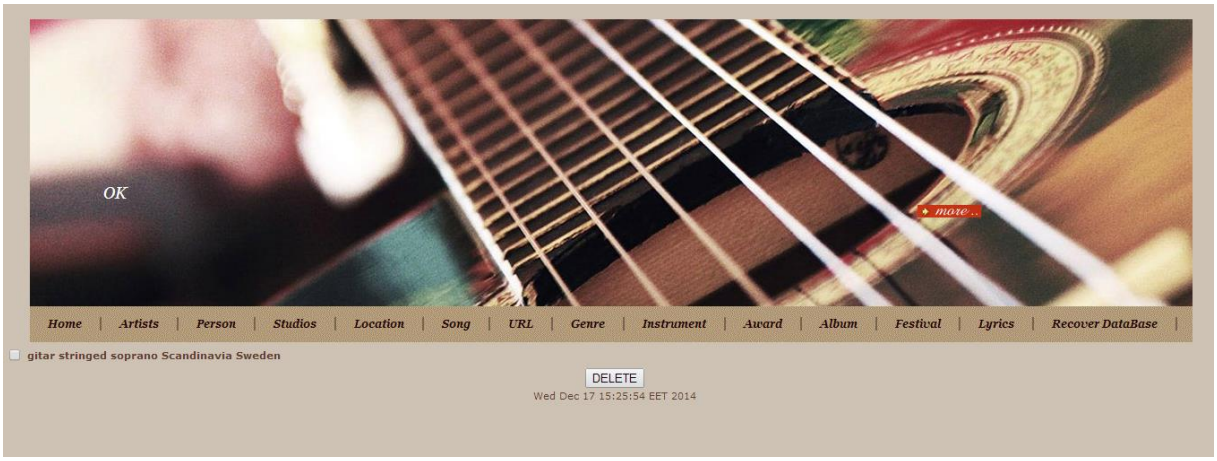
8.3.1. Add Operation

In add operation, we add one row to our sql table. User clicks on Add Instrument link. Then user fills the blanks in this form page and click on “Save” button. After clicked on “Save” button, list operation is realized.



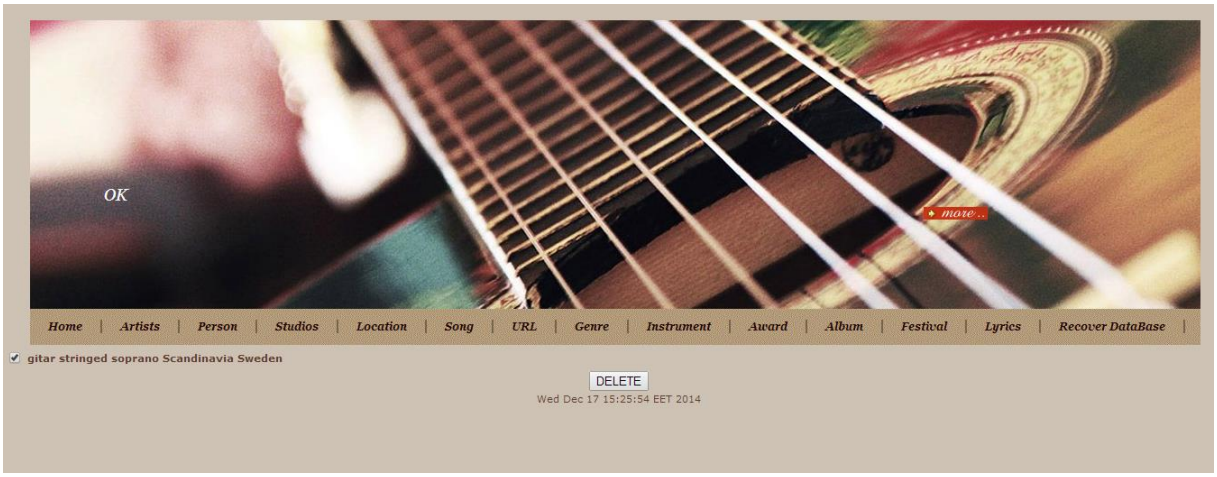
8.3.2. List Operation

In list operation user can see all Instruments in our sql table. User clicks on “List Instrument” link.



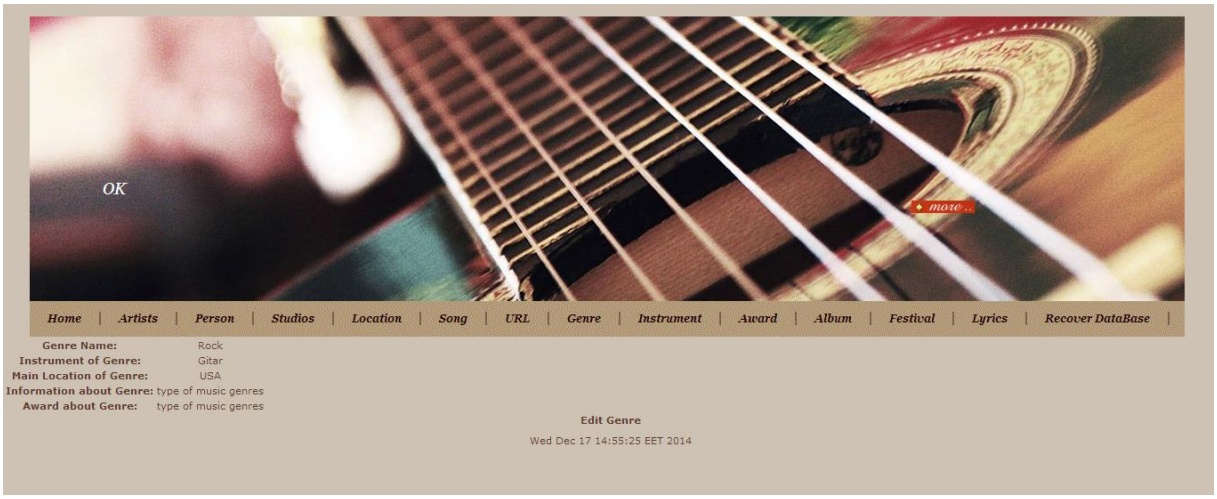
8.3.3. Delete Operation

In delete operation, user can delete one row from our sql table. User clicks on check box next to listed compenent which will be deleted. Then there is a “delete” button on bottom. When clicked this button, it is deleted.

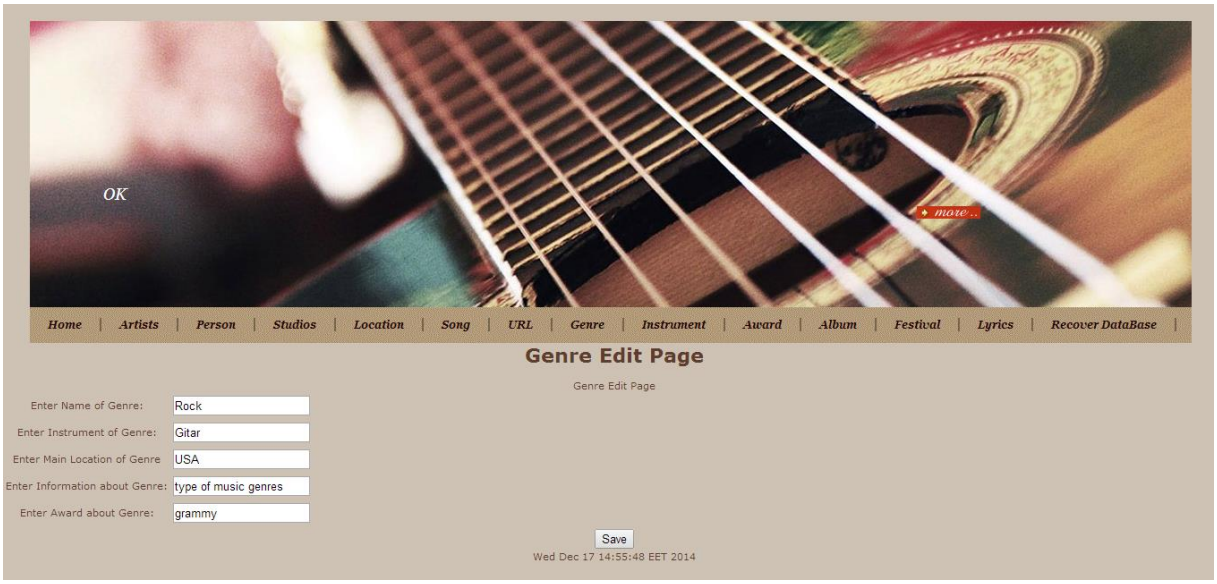


8.3.4. Update Operation

In update operation, user can update one row from our sql table. User clicks on listed compenent, Instrument Display page is shown on screen.



If user wants to update this informations , “Edit Instrument” button is clicked then edit form page is shown on screen



8.3.5. Search Operation

In search operation, any word can be searched in name of studio then all found rows from sql table is listed. To realize this operation, firstly, “Search Instrument” link is clicked. Then Instrument Search Page is shown.

Then searched word is fill in blank than click on “Search” button.

9. Music Genre

This class is used to indicate which music or singer/band in which music genre which is a conventional category that identifies pieces of music as belonging to a shared tradition or set of conventions. It is to be distinguished from musical form andmusical style, although in practice these terms are sometimes used interchangeably. This class can carry add, list, delete, update and search operations out. The main music genre class is connections with other classes.

9.1. Attributes

Genre class has 7 attributes. Attributes are shown below with their variable types.

```
private Integer _id = null;
private String name;
private String instrument;
private String location;
private String info;
private String award;
private String searchWord;
```

name attribute is used for name of a music genre. location attribute is used for hometown of music genre. Using location attribute, we will connect music genre class and location class. instrument attribute is used for instrument of music genre. info shows information about a music genre. Using instrument attribute, we will connect instrument class and genre class. award attribute is used to show given awards of music genre. Using award attribute, we will connect award class and genre class. searchWord attribute is used for searching methods of this class.

9.2. Connections

We use alter table property for connections. Firstly alter table additions is written in init.sql file. After that we supply neccessary changes on GenreCollectionJDBC file.

init.sql

```
CREATE TABLE GENRETABLE (ID INTEGER PRIMARY KEY AUTOINCREMENT,NAME VARCHAR(40) NOT NULL,INSTRUMENT VARCHAR(40) NOT NULL, LOCATION VARCHAR(55) NOT NULL, INFO VARCHAR(55) NOT NULL, AWARD VARCHAR(45));

ALTER TABLE GENRETABLE ADD COLUMN FK_AWARD INTEGER REFERENCES AWARDTABLE(ID);

ALTER TABLE GENRETABLE ADD COLUMN FK_INSTRUMENT INTEGER REFERENCES INSTRUMENTTABLE(ID);

ALTER TABLE GENRETABLE ADD COLUMN FK_LOCATION INTEGER REFERENCES LOCATIONTABLE(ID);


INSERT INTO GENRETABLE (NAME, INSTRUMENT, LOCATION, INFO, AWARD) VALUES('Rock', 'Gitar', 'USA', 'type of music genres', 'grammy');
```

GenreCollectionJDBC

A query is created. This query is used to connect searched id of alter table. For instance, location table and genre table is connected with this query:

```
String query0 = "SELECT ID FROM LOCATIONTABLE WHERE TOWN ='%"
                          + genre.getLocation() + "%'";
```

Another query is created. This query is used to connect searched id of alter table. For instance, award table and genre table is connected with this query:

```
String query1 = "SELECT ID FROM AWARDTABLE WHERE NAME LIKE '%"
                          + genre.getAward() + "%'";
```

Another query is created. This query is used to connect searched id of alter table. For instance, instrument table and genre table is connected with this query:

```
String query2 = "SELECT ID FROM INSTRUMENTTABLE WHERE NAME LIKE '%"
                          + genre.getInstrument() + "%'";
```

Main query of insert method is created.

```
String query = "INSERT INTO GENRETABLE (NAME, INSTRUMENT, LOCATION, INFO, AWARD,
FK_AWARD, FK_LOCATION, FK_INSTRUMENT) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
```

Main query of delete method is created.

```
String query = "DELETE FROM GENRETABLE WHERE (ID = ?)";
```

Main query of update method is created.

```
String query0 = "SELECT ID FROM AWARDTABLE WHERE NAME ='"
                          + genre.getAward() + "'";
String query1 = "SELECT ID FROM AWARDTABLE WHERE NAME LIKE '%"
                          + genre.getAward() + "%'";
String query2 = "SELECT ID FROM INSTRUMENTTABLE WHERE NAME LIKE '%"
                          + genre.getInstrument() + "%'";
```

```
String query = "UPDATE GENRETABLE SET NAME=?, INSTRUMENT=?, LOCATION=?, INFO=?,
AWARD=?, FK_AWARD=? WHERE (ID=?)";
```

Main query of delete method is created.

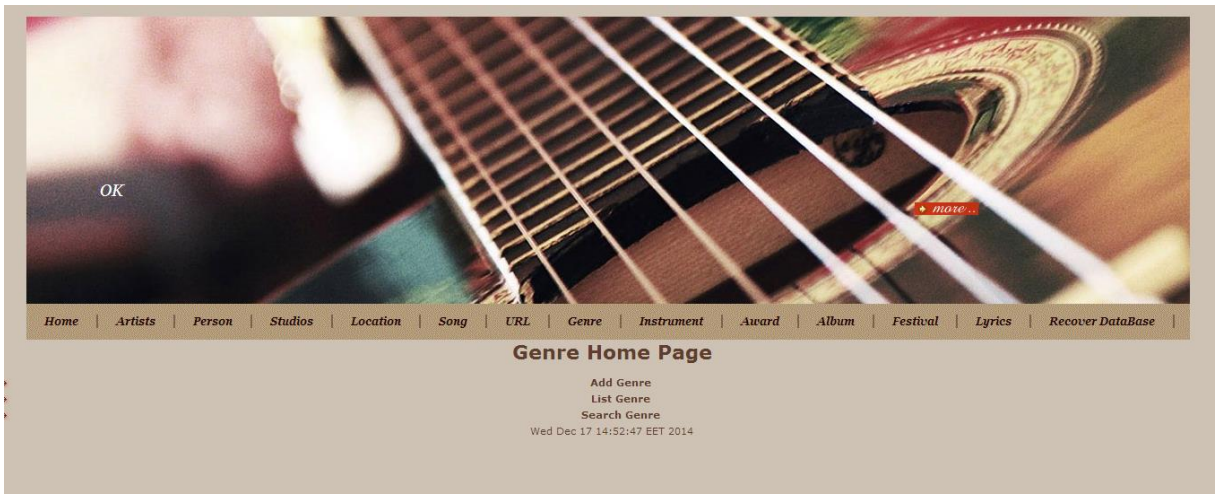
```
String query = "SELECT ID, NAME, INSTRUMENT, LOCATION, INFO, AWARD FROM
GENRETABLE";
```

As seen in sql code, location attribute of Genre class is compared with town attribute in location class, if it is pair, connection is done.

ID	NAME	INSTRUMENT	LOCATION	INFO	AWARD
1	Rock	Gitar	USA	type of music genres	grammy
2	Pop	Gitar	United States of Am...	most known music ...	grammy

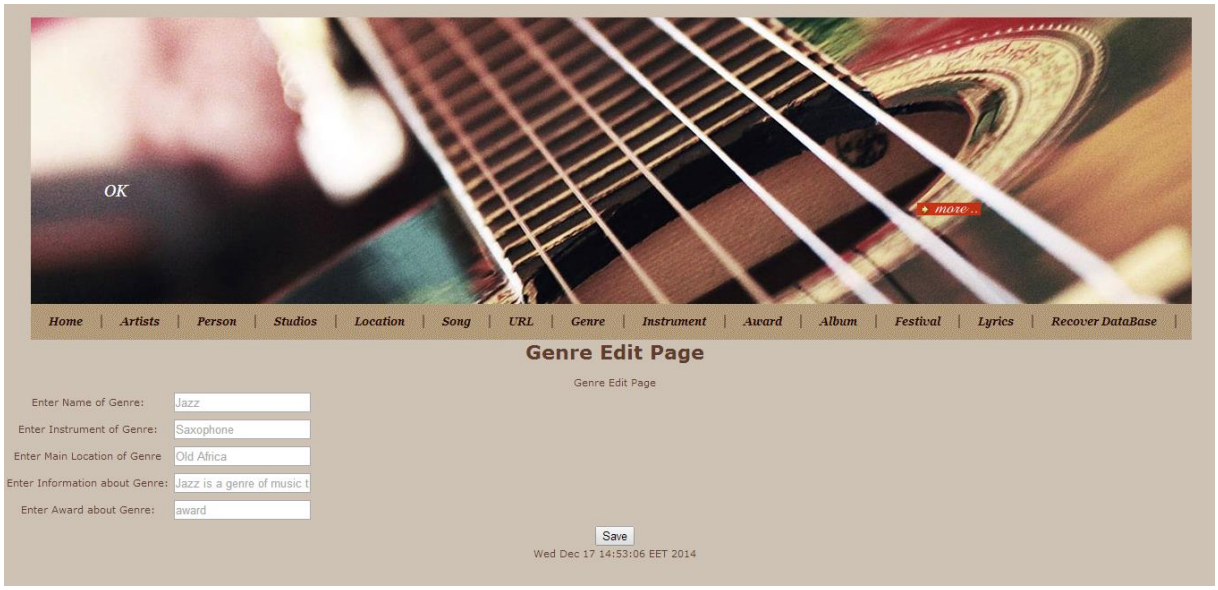
9.3. Operations

For realizing operations, firstly we click on Genre tab on our website. It send user to Genre Home Page.



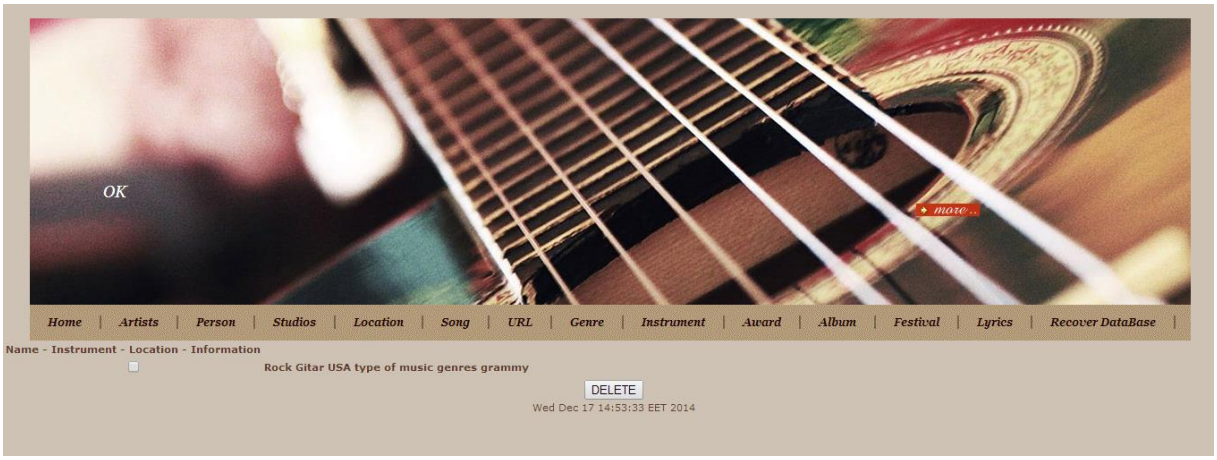
9.3.1. Add Operation

In add operation, we add one row to our sql table. User clicks on Add Genre link. Then user fills the blanks in this form page and click on “Save” button. After clicked on “Save” button, list operation is realized.



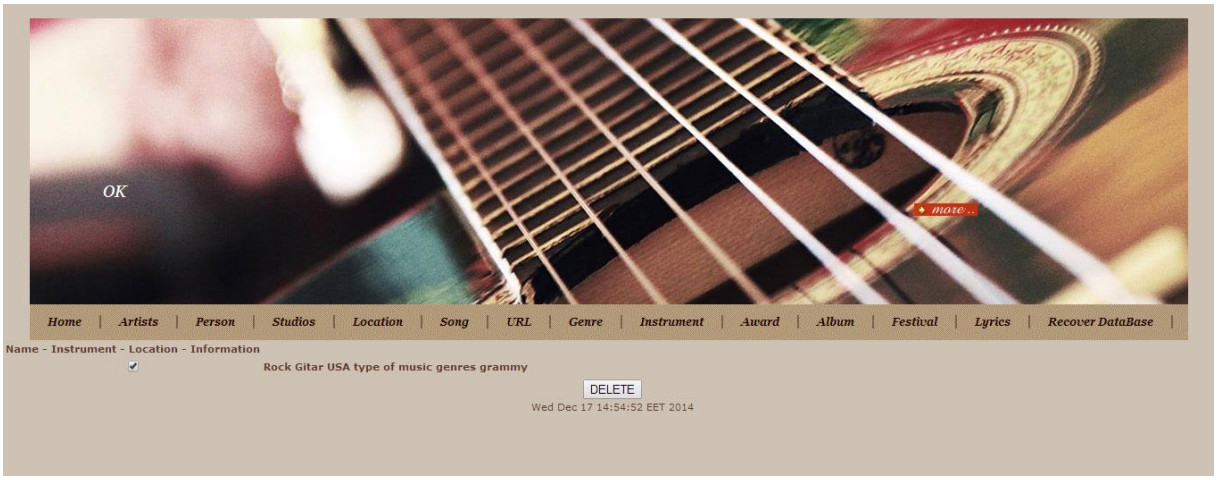
9.3.2. List Operation

In list operation user can see all genres in our sql table. User clicks on “List Genre” link.



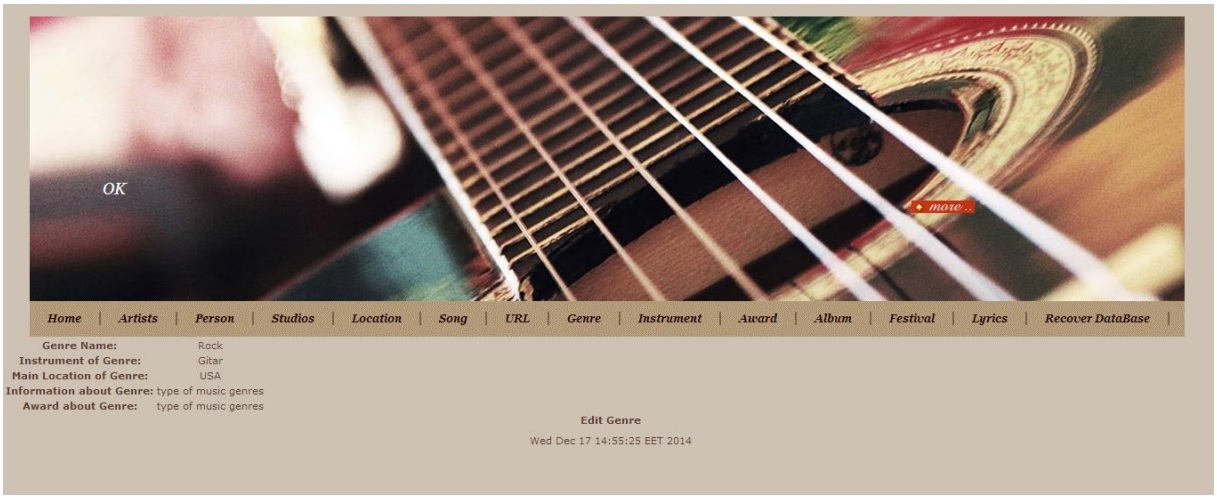
9.3.3. Delete Operation

In delete operation, user can delete one row from our sql table. User clicks on check box next to listed compenent which will be deleted. Then there is a “delete” button on bottom. When clicked this button, it is deleted.

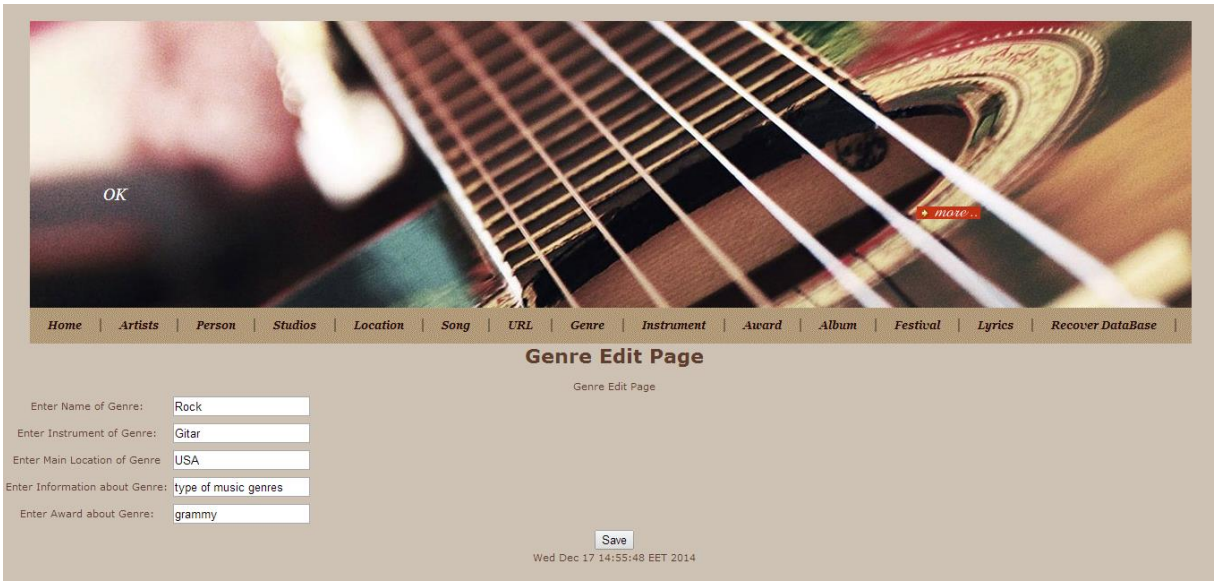


9.3.4. Update Operation

In update operation, user can update one row from our sql table. User clicks on listed compenent, Genre Display page is shown on screen.

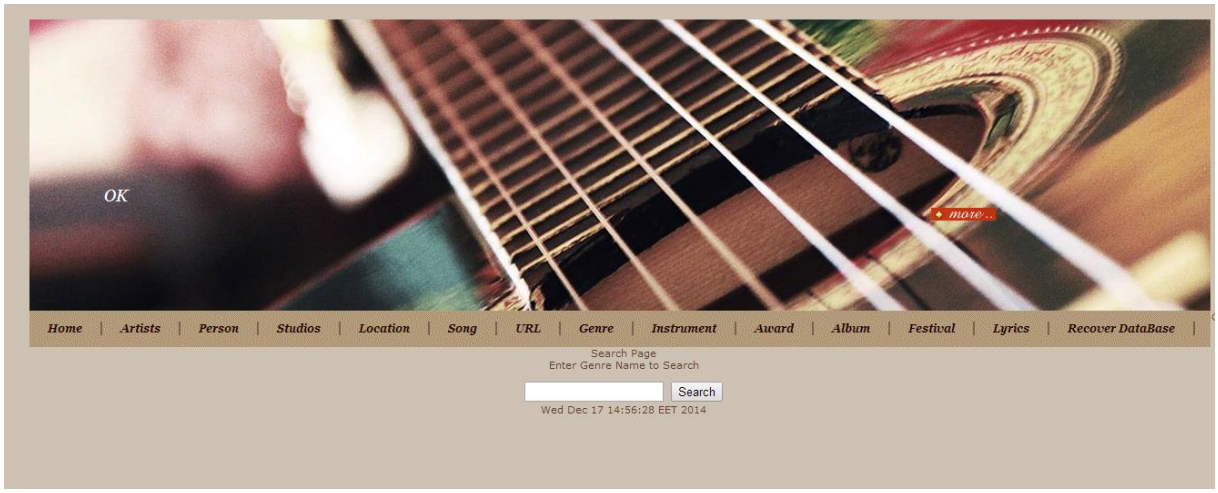


If user wants to update this informations , “Edit Genre” button is clicked then edit form page is shown on screen

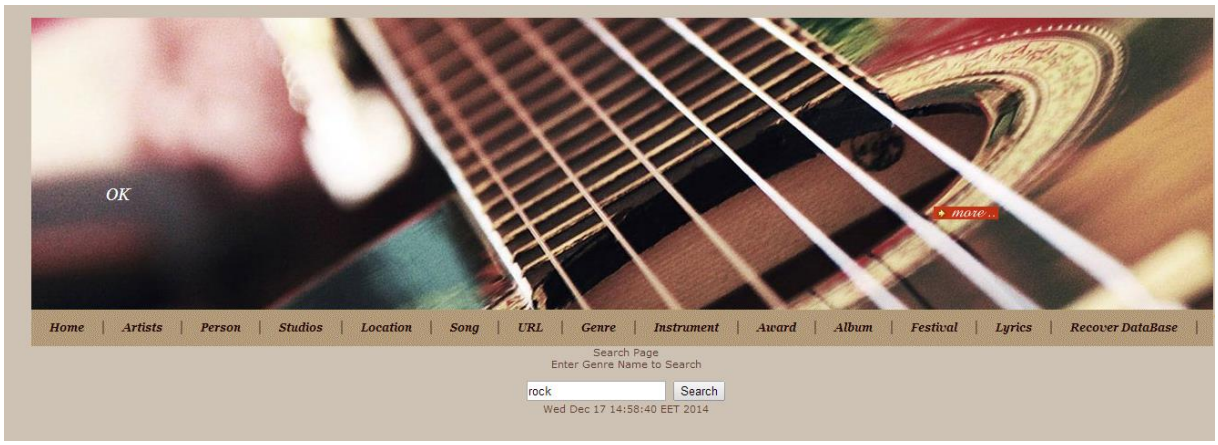


9.3.5. Search Operation

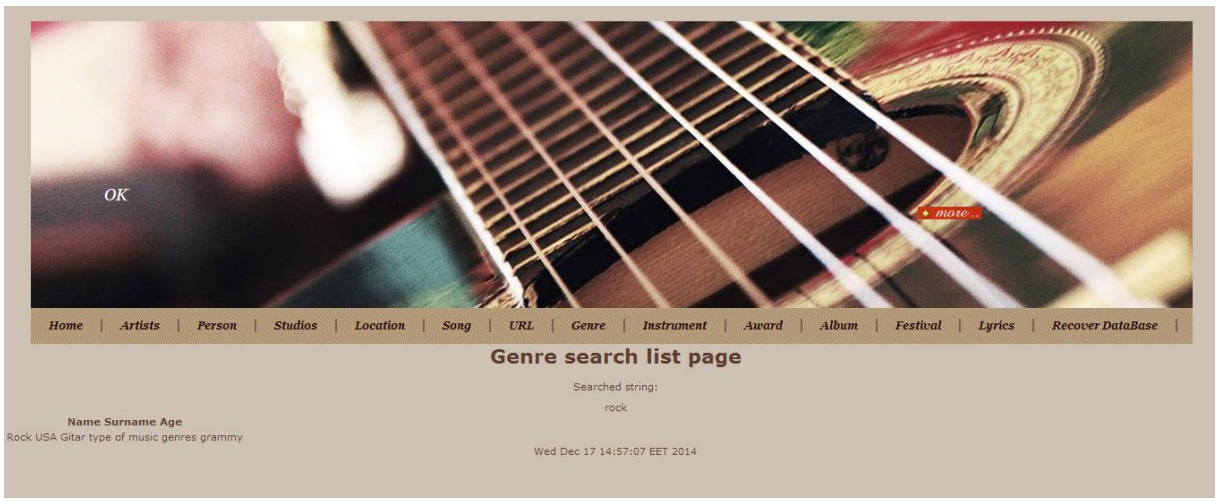
In search operation, any word can be searched in name of studio then all found rows from sql table is listed. To realize this operation, firstly, “Search Genre” link is clicked. Then Genre Search Page is shown.

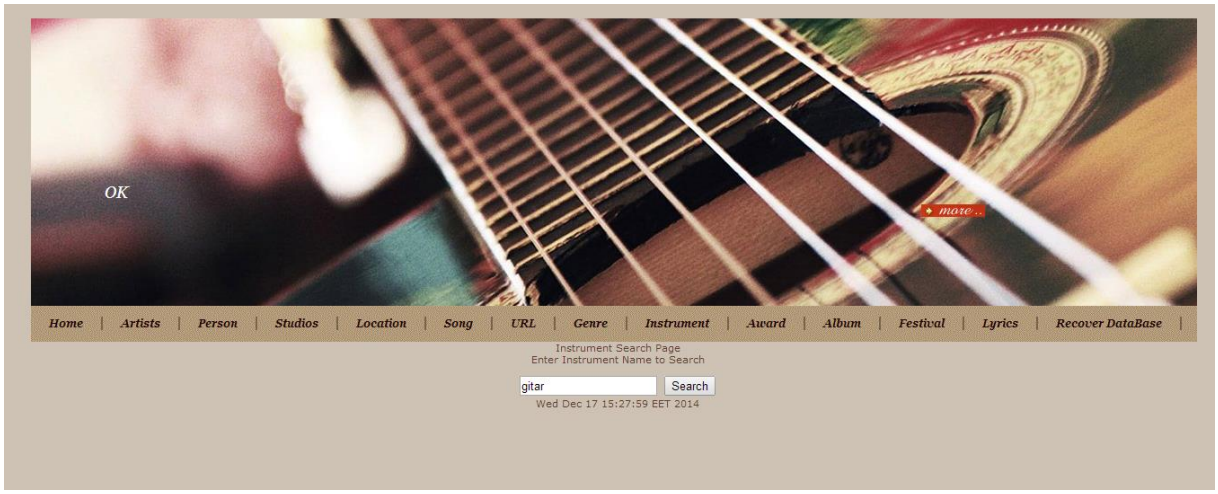


Then searched word is fill in blank than click on “Search” button.



Then all found rows from sql table is listed.





Then all found rows from sql table is listed.