Olive makes it easy to edit your videos.

```
-- MySQL dump 10.11
--
-- Host: localhost    Database: OliveData
-- ------------------------------------------------------
-- Server version   5.0.51a-24+lenny4

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Current Database: `OliveData`
--

CREATE DATABASE /*!32312 IF NOT EXISTS*/ `OliveData` /*!40100 DEFAULT CHARACTER SET latin1 *
/;

USE `OliveData`;

--
-- Table structure for table `Accounts`
--

DROP TABLE IF EXISTS `Accounts`;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
CREATE TABLE `Accounts` (
  `AccountID` int(11) NOT NULL auto_increment,
  `Username` varchar(255) default NULL,
  `Password` varchar(255) default NULL,
  `Name` varchar(255) default NULL,
  `Email` varchar(255) default NULL,
  `SecurityQuestion` varchar(255) default NULL,
  `SecurityAnswer` varchar(255) default NULL,
  PRIMARY KEY  (`AccountID`),
  UNIQUE KEY `Username` (`Username`)
) ENGINE=MyISAM AUTO_INCREMENT=49 DEFAULT CHARSET=latin1;
SET character_set_client = @saved_cs_client;

--
-- Table structure for table `Projects`
--

DROP TABLE IF EXISTS `Projects`;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
CREATE TABLE `Projects` (
  `ProjectID` int(11) NOT NULL auto_increment,
  `Name` varchar(255) default NULL,
  `AccountID` int(11) default NULL,
  `Icon` varchar(255) default NULL,
  `PoolPosition` int(11) default NULL,
  PRIMARY KEY  (`ProjectID`),
  KEY `AccountID` (`AccountID`)
) ENGINE=MyISAM AUTO_INCREMENT=74 DEFAULT CHARSET=latin1;
SET character_set_client = @saved_cs_client;

--
-- Table structure for table `S3Credentials`
--

DROP TABLE IF EXISTS `S3Credentials`;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
CREATE TABLE `S3Credentials` (
  `S3ID` int(11) NOT NULL auto_increment,
  `AWSAccessKey` varchar(50) default NULL,
  `AWSSecretKey` varchar(50) default NULL,
  PRIMARY KEY  (`S3ID`)
) ENGINE=MyISAM AUTO_INCREMENT=3 DEFAULT CHARSET=latin1;
SET character_set_client = @saved_cs_client;

--
-- Table structure for table `Videos`
--

DROP TABLE IF EXISTS `Videos`;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
CREATE TABLE `Videos` (
  `VideoID` int(11) NOT NULL auto_increment,
  `Name` varchar(255) default NULL,
  `URL` varchar(255) default NULL,
  `ProjectID` int(11) default NULL,
  `TimelinePosition` int(11) default NULL,
  `Icon` varchar(255) default NULL,
  `IsSelected` int(11) default NULL,
  `PoolPosition` int(11) default NULL,
  PRIMARY KEY  (`VideoID`),
  KEY `ProjectID` (`ProjectID`)
) ENGINE=MyISAM AUTO_INCREMENT=141 DEFAULT CHARSET=latin1;
SET character_set_client = @saved_cs_client;

--
-- Table structure for table `ZencoderCredentials`
--

DROP TABLE IF EXISTS `ZencoderCredentials`;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
CREATE TABLE `ZencoderCredentials` (
  `ZenID` int(11) NOT NULL auto_increment,
  `ZencoderAPIKey` varchar(255) default NULL,
  PRIMARY KEY  (`ZenID`)
) ENGINE=MyISAM AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;
SET character_set_client = @saved_cs_client;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2011-04-01 22:04:01
```

```
package com.readytalk.olive.json;

public class AddToSelectedRequest {

    // No-args constructor
    public AddToSelectedRequest() {

    }

    public String command;
    public Arguments arguments;

    // Must be static for Gson to work.
    // See: http://sites.google.com/site/gson/gson-user-guide#TOC-Nested-Classes-including-I
nner-Clas
    public static class Arguments {

        // No-args constructor
        public Arguments() {

        }

        public String video;
    }
}
```

```
package com.readytalk.olive.json;


public class CombineVideosRequest {

    public CombineVideosRequest(){

    }
    public String command;
    public Arguments arguments;

    // Must be static for Gson to work.
    // See: http://sites.google.com/site/gson/gson-user-guide#TOC-Nested-Classes-including-I
nner-Clas
    public static class Arguments {

        // No-args constructor
        public Arguments() {

        }
    }
}
```

```
package com.readytalk.olive.json;

public class DeleteAccountRequest {

    // No-args constructor
    public DeleteAccountRequest() {

    }

    public String command;
    public Arguments arguments;

    // Must be static for Gson to work.
    // See: http://sites.google.com/site/gson/gson-user-guide#TOC-Nested-Classes-including-I
nner-Clas
    public static class Arguments {

        // No-args constructor
        public Arguments() {

        }

        public String account;
    }
}
```

```
package com.readytalk.olive.json;

public class DeleteProjectRequest {

    // No-args constructor
    public DeleteProjectRequest() {

    }

    public String command;
    public Arguments arguments;

    // Must be static for Gson to work.
    // See: http://sites.google.com/site/gson/gson-user-guide#TOC-Nested-Classes-including-I
nner-Clas
    public static class Arguments {

        // No-args constructor
        public Arguments() {

        }

        public String project;
    }
}
```

```
package com.readytalk.olive.json;

public class DeleteVideoRequest {

    // No-args constructor
    public DeleteVideoRequest() {

    }

    public String command;
    public Arguments arguments;

    // Must be static for Gson to work.
    // See: http://sites.google.com/site/gson/gson-user-guide#TOC-Nested-Classes-including-I
nner-Clas
    public static class Arguments {

        // No-args constructor
        public Arguments() {

        }

        public String video;
    }
}
```

```
package com.readytalk.olive.json;

public class GeneralRequest {

    // No-args constructor
    public GeneralRequest() {

    }

    public String command;
    public Arguments arguments;

    // Must be static for Gson to work.
    // See: http://sites.google.com/site/gson/gson-user-guide#TOC-Nested-Classes-including-I
nner-Clas
    public static class Arguments {

        // No-args constructor
        public Arguments() {

        }
    }
}
```

```
package com.readytalk.olive.json;

public class GetAccountInformationResponse {

    public GetAccountInformationResponse(String name, String email,
            String password, String securityQuestion, String securityAnswer) {
        this.name = name;
        this.email = email;
        this.password = password;
        this.securityQuestion = securityQuestion;
        this.securityAnswer = securityAnswer;
    }

    public String name;
    public String email;
    public String password;
    public String securityQuestion;
    public String securityAnswer;
}
```

```
package com.readytalk.olive.json;

public class RemoveFromSelectedRequest {

    // No-args constructor
    public RemoveFromSelectedRequest() {

    }

    public String command;
    public Arguments arguments;

    // Must be static for Gson to work.
    // See: http://sites.google.com/site/gson/gson-user-guide#TOC-Nested-Classes-including-I
nner-Clas
    public static class Arguments {

        // No-args constructor
        public Arguments() {

        }

        public String video;
    }
}
```

```
package com.readytalk.olive.json;

public class RenameProjectRequest {

    // No-args constructor
    public RenameProjectRequest() {

    }

    public String command;
    public Arguments arguments;

    // Must be static for Gson to work.
    // See: http://sites.google.com/site/gson/gson-user-guide#TOC-Nested-Classes-including-I
nner-Clas
    public static class Arguments {

        // No-args constructor
        public Arguments() {

        }

        public String oldProjectName;
        public String newProjectName;
    }
}
```

```
package com.readytalk.olive.json;

public class RenameVideoRequest {

    // No-args constructor
    public RenameVideoRequest() {

    }

    public String command;
    public Arguments arguments;

    // Must be static for Gson to work.
    // See: http://sites.google.com/site/gson/gson-user-guide#TOC-Nested-Classes-including-I
nner-Clas
    public static class Arguments {

        // No-args constructor
        public Arguments() {

        }

        public String oldVideoName;
        public String newVideoName;
    }
}
```

```
package com.readytalk.olive.json;

public class SplitVideoRequest {

    // No-args constructor
    public SplitVideoRequest() {

    }

    public String command;
    public Arguments arguments;

    // Must be static for Gson to work.
    // See: http://sites.google.com/site/gson/gson-user-guide#TOC-Nested-Classes-including-I
nner-Clas
    public static class Arguments {

        // No-args constructor
        public Arguments() {

        }

        public String video;
        public double splitTimeInSeconds;
    }
}
```

```
package com.readytalk.olive.json;

public class UpdateProjectsPositionRequest {

    // No-args constructor
    public UpdateProjectsPositionRequest() {

    }

    public String command;
    public Arguments arguments;

    // Must be static for Gson to work.
    // See: http://sites.google.com/site/gson/gson-user-guide#TOC-Nested-Classes-including-I
nner-Clas
    public static class Arguments {

        // No-args constructor
        public Arguments() {

        }

        public Projects[] projects;

        // Must be static for Gson to work.
        // See: http://sites.google.com/site/gson/gson-user-guide#TOC-Nested-Classes-includi
ng-Inner-Clas
        public static class Projects {

            // No-args constructor
            public Projects() {

            }

            public String project;
            public int position;
        }
    }
}
```

```
package com.readytalk.olive.json;

public class UpdateTimelinePositionRequest {

    // No-args constructor
    public UpdateTimelinePositionRequest() {

    }

    public String command;
    public Arguments arguments;

    // Must be static for Gson to work.
    // See: http://sites.google.com/site/gson/gson-user-guide#TOC-Nested-Classes-including-I
nner-Clas
    public static class Arguments {

        // No-args constructor
        public Arguments() {

        }

        public Videos[] videos;

        // Must be static for Gson to work.
        // See: http://sites.google.com/site/gson/gson-user-guide#TOC-Nested-Classes-includi
ng-Inner-Clas
        public static class Videos {

            // No-args constructor
            public Videos() {

            }

            public String video;
            public int position;
        }
    }
}
```

```
package com.readytalk.olive.json;

public class UpdateVideosPositionRequest {

    // No-args constructor
    public UpdateVideosPositionRequest() {

    }

    public String command;
    public Arguments arguments;

    // Must be static for Gson to work.
    // See: http://sites.google.com/site/gson/gson-user-guide#TOC-Nested-Classes-including-I
nner-Clas
    public static class Arguments {

        // No-args constructor
        public Arguments() {

        }

        public Videos[] videos;

        // Must be static for Gson to work.
        // See: http://sites.google.com/site/gson/gson-user-guide#TOC-Nested-Classes-includi
ng-Inner-Clas
        public static class Videos {

            // No-args constructor
            public Videos() {

            }

            public String video;
            public int position;
        }
    }
}
```

```
package com.readytalk.olive.json;

public class ZencoderInitialResponse {

    // No-args constructor
    public ZencoderInitialResponse() {

    }

    public Outputs[] outputs;
    public int id;

    // Must be static for Gson to work.
    // See: http://sites.google.com/site/gson/gson-user-guide#TOC-Nested-Classes-including-I
nner-Clas
    public static class Outputs {

        // No-args constructor
        public Outputs() {

        }

        public String url;
        public String label;
        public int id;
    }
}
```

```
package com.readytalk.olive.logic;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.LinkedList;
import java.util.List;
import java.util.logging.Logger;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.sql.DataSource;

import org.jets3t.service.security.AWSCredentials;

import com.readytalk.olive.model.Project;
import com.readytalk.olive.model.User;
import com.readytalk.olive.model.Video;

/**
 * class DatabaseApi provides tool to connect to Database
 *
 * @author Team Olive
 *
 */
public class DatabaseApi {

    private static Logger log = Logger.getLogger(DatabaseApi.class.getName());

    // CAUTION: closeConnection() must be called sometime after this method.
    /**
     * enables to connect to Olive database
     */
    public static Connection getDBConnection() {
        try {
            Context initCtx = new InitialContext();
            DataSource ds = (DataSource) initCtx
                    .lookup("java:comp/env/jdbc/OliveData");
            return ds.getConnection();
        } catch (Exception e) {
            e.printStackTrace();
        }
        // TODO Close the connection here on error.
        return null;
    }

    /**
     * enables to disconnect from Olive database
     *
     * @param c
     *            an object that is using Olive database
     */
    public static void closeConnection(Connection c) {
        try {
            c.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * Check whether the user and password are registered in the Olive database
     *
     * @param username
     *            username that has been registered in the Olive database
     * @param password
     *            password that has been registered in the Olive database
     * @return return boolean expression saying whether the information ( username/pw ) is r
egistered or not.
     */
    public static Boolean isAuthorized(String username, String password) {
        Connection conn = getDBConnection();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);
            s = "SELECT AccountID FROM Accounts WHERE Username = '" + username
                    + "' AND Password = Password('" + password + "');";
            ResultSet r = st.executeQuery(s);
            if (r.first()) {
                return true;
            }
            return false;
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
        return false; // Error!
    }

    /**
     * Enables to register an account to the Olive database
     *
     * @param user
     *            user information typed in the registration form by an user
     * @return passes true if the passed in values are valid to be registered, and passes fa
lse if the passed in values are invalid to register
     */
    public static boolean AddAccount(User user) {
        Connection conn = getDBConnection();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);
            s = "INSERT INTO Accounts (Username, Password, Name, Email) "
                    + "VALUES ('" + user.getUsername() + "', Password('"
                    + user.getPassword() + "') " + ", '" + user.getName()
                    + "', '" + user.getEmail() + "');";
            st.executeUpdate(s);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
        return false;
    }

    // Used for the general query: "SELECT W FROM X WHERE Y = Z;"
    /**
     * Get an information which requested by an user from encrypted table
     */
```

```
public static String getUnknownValueFromTable(String unknownLabel,
        String table, String knownLabel, String knownValue) {
    Connection conn = getDBConnection();
    try {
        Statement st = conn.createStatement();
        String s = "USE OliveData;";
        st.executeUpdate(s);

        s = "SELECT " + unknownLabel + " FROM " + table + " WHERE "
                + knownLabel + " = '" + knownValue + "';";
        ResultSet r = st.executeQuery(s);
        String unknownValue = "";
        if (r.first()) {
            unknownValue = r.getString(unknownLabel);
        }
        return unknownValue;
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        closeConnection(conn);
    }
    return null;
}

/**
 * Brings the specific account information according to user input id
 *
 * @param username
 *            specific username entered by an user
 * @return return corresponding information of the user name entered by an user
 */
public static int getAccountId(String username) {
    return Integer.parseInt(getUnknownValueFromTable("AccountID",
            "Accounts", "Username", username));
}

/**
 * Brings the Username according to its accountID
 *
 * @param accountId
 *            unique account number given to an user
 * @return corresponding username and account to the accountID from the Olive database
 */
public static String getAccountUsername(int accountId) {
    return getUnknownValueFromTable("Username", "Accounts", "AccountID",
            Integer.toString(accountId));
}

/**
 * Brings the Password information according to its accountID
 *
 * @param accountId
 *            unique account number given to an user
 * @return corresponding password and account to the accountID from the Olive database
 */
public static String getAccountPassword(int accountId) {
    return getUnknownValueFromTable("Password", "Accounts", "AccountID",
            Integer.toString(accountId));
}

/**
 * Brings the Account name according to its accountID
 *
 * @param accountId
 *            unique account number given to an user
 * @return corresponding name and account to the accountID from the Olive database
 */
public static String getAccountName(int accountId) {
    return getUnknownValueFromTable("Name", "Accounts", "AccountID",
            Integer.toString(accountId));
}

/**
 * Brings the email address according to its accountID
 *
 * @param accountId
 *            unique account number given to an user
 * @return corresponding email and account to the accountID from the Olive database
 */
public static String getAccountEmail(int accountId) {
    return getUnknownValueFromTable("Email", "Accounts", "AccountID",
            Integer.toString(accountId));
}

/**
 * Brings the security question according to its accountID
 *
 * @param accountId
 *            unique account number given to an user
 * @return corresponding security question and account to the accountID from the Olive database
 */
public static String getAccountSecurityQuestion(int accountId) {
    return getUnknownValueFromTable("SecurityQuestion", "Accounts",
            "AccountID", Integer.toString(accountId));
}

/**
 * Brings the answer of the security question according to its accountID
 *
 * @param accountId
 *            unique account number given to an user
 * @return corresponding security question answer and account to the accountID from the Olive database
 */
public static String getAccountSecurityAnswer(int accountId) {
    return getUnknownValueFromTable("SecurityAnswer", "Accounts",
            "AccountID", Integer.toString(accountId));
}

/**
 * Gets the number of projects in an account
 *
 * @param accountId
 *            unique account number given to an user
 * @return number of projects
 */
public static int getNumberOfProjects(int accountId) {

    Connection conn = getDBConnection();
    try {
        Statement st = conn.createStatement();
        String s = "USE OliveData;";
        st.executeUpdate(s);
        ResultSet r;
        s = "SELECT ProjectID FROM Projects WHERE AccountID = " + accountId
```

```
                + ";";
        r = st.executeQuery(s);
        int numberOfProjects = 0;

        if (r.first()) {
            do {
                numberOfProjects++;
            } while (r.next());
        }
        return numberOfProjects;
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        closeConnection(conn);
    }
    return -1; // error!
}

public static Boolean usernameExists(String username){
    Connection conn = getDBConnection();
    try {
        Statement st = conn.createStatement();
        String s = "USE OliveData;";
        st.executeUpdate(s);

        s = "SELECT AccountID FROM Accounts WHERE Username = '"+username+"';";
        ResultSet r = st.executeQuery(s);
        if (r.first()) {
            return true;
        }
        return false;
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        closeConnection(conn);
    }

    return false;
}

/**
 * sets the data for edit account section of the web page
 *
 * @param user
 *            brings the data of an user
 * @return true if data is updated, false if not
 */
public static Boolean editAccount(User user) {
    Connection conn = getDBConnection();
    try {
        Statement st = conn.createStatement();
        String s = "USE OliveData;";
        st.executeUpdate(s);
        s = "UPDATE Accounts SET Name = '" + user.getName()
                + "' WHERE Username = '" + user.getUsername() + "';";
        st.executeUpdate(s);

        s = "UPDATE Accounts SET Password = Password('"
                + user.getPassword() + "') WHERE Username = '"
                + user.getUsername() + "';";
        st.executeUpdate(s);

        s = "UPDATE Accounts SET Email = '" + user.getEmail()
                + "' WHERE Username = '" + user.getUsername() + "';";
        st.executeUpdate(s);

        s = "UPDATE Accounts SET SecurityQuestion = '"
                + user.getSecurityQuestion() + "' WHERE Username = '"
                + user.getUsername() + "';";
        st.executeUpdate(s);

        s = "UPDATE Accounts SET SecurityAnswer = '"
                + user.getSecurityAnswer() + "' WHERE Username = '"
                + user.getUsername() + "';";
        st.executeUpdate(s);
        return true;
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        closeConnection(conn);
    }
    return false;
}

/**
 * enables to remove an account registered
 *
 * @param accountId
 *            unique account number given to an user
 */
public static void deleteAccount(int accountId) {
    // TODO implement

    // int projectId = getProjectId(name, accountId);
    // int videoId = getVideoId(name, projectId, accountId);

    // int projectId = -1;
    // int videoId = -1;

    Connection conn = getDBConnection();
    try {
        Statement st = conn.createStatement();
        String s = "USE OliveData;";
        st.executeUpdate(s);
        ResultSet r;
        s = "SELECT ProjectID FROM Projects WHERE AccountID = " + accountId
                + ";";
        r = st.executeQuery(s);
        if (r.first()) {
            do {
                deleteProject(r.getInt("ProjectID"));
            } while (r.next());
        }
        // TODO Broken
        // Delete all videos associated with all projects associated with the account.
        // s = "DELETE FROM Videos WHERE ProjectID = '" + projectId + "';"; // TODO Add
error checking
        // st.executeUpdate(s);

        // Delete all projects associated with the account.
        // s = "DELETE FROM Projects WHERE AccountID = '" + accountId + "';"; // TODO Ad
d error checking
        // st.executeUpdate(s);

        // Delete the account itself.
        s = "DELETE FROM Accounts WHERE AccountID = '" + accountId + "';"; // TODO Add e
```

```
rror checking
            st.executeUpdate(s);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
    }

    /**
     * Brings the project id registered in the Olive database
     *
     * @param projectName
     *            name of the project on the web
     * @param accountId
     *            unique account number given to an user
     * @return projectID if exists on the database, -1 if not there
     */
    public static int getProjectId(String projectName, int accountId) {
        Connection conn = getDBConnection();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);
            s = "SELECT ProjectID FROM Projects WHERE Name = '" + projectName
                    + "' AND AccountID = '" + accountId + "';";
            ResultSet r = st.executeQuery(s);
            int projectId = -1;
            if (r.first()) {
                projectId = r.getInt("ProjectID");
            }
            return projectId;
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
        return -1;
    }

    /**
     * Brings the project name according to its projectID
     *
     * @param projectId
     *            unique project number given to a project of an user
     * @return name of the project in the projects of the accountID table of the Olive datab
ase
     */
    public static String getProjectName(int projectId) {
        return getUnknownValueFromTable("Name", "Projects", "ProjectID",
                Integer.toString(projectId));
    }

    /**
     * Brings the project accountID according to its projectID
     *
     * @param projectId
     *            unique project number given to a project of an user
     * @return accountID of the project in the projects of the accountID table of the Olive
database
     */
    public static int getProjectAccountId(int projectId) {
        return Integer.parseInt(getUnknownValueFromTable("AccountID",
                "Projects", "ProjectID", Integer.toString(projectId)));
    }

    /**
     * Brings the project icon according to its projectID
     *
     * @param projectId
     *            unique project number given to a project of an user
     * @return project icon of the project in the projects of the accountID table of the Oli
ve database
     */
    public static String getProjectIcon(int projectId) {
        return getUnknownValueFromTable("Icon", "Projects", "ProjectID",
                Integer.toString(projectId));
    }

    /**
     * Gets number of videos in a project
     *
     * @param projectId
     *            unique project number given to a project
     * @return number of videos
     */
    public static int getNumberOfVideos(int projectId) {

        Connection conn = getDBConnection();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);
            ResultSet r;
            s = "SELECT VideoID FROM Videos WHERE ProjectID = " + projectId
                    + ";";
            r = st.executeQuery(s);
            int numberOfVideos = 0;

            if (r.first()) {
                do {
                    numberOfVideos++;
                } while (r.next());
            }
            return numberOfVideos;
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
        return -1; // error!
    }

    /**
     * Gets projectIds projects) in a given account
     *
     * @param accountId
     *            unique account number given an user
     * @return lists of project in a form of int array
     */
    public static int[] getProjectIds(int accountId) {
        Connection conn = getDBConnection();
        List<Integer> projectIds = new LinkedList<Integer>();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
```

```
            st.executeUpdate(s);
            ResultSet r;
            s = "SELECT ProjectID FROM Projects WHERE AccountID = " + accountId
                    + ";";
            r = st.executeQuery(s);
            if (r.first()) {
                do {
                    projectIds.add(r.getInt("ProjectID"));
                } while (r.next());
            }

            // Convert the List to an int array.
            int[] projectIdsAsIntArray = new int[projectIds.size()];
            for (int i = 0; i < projectIds.size(); ++i) {
                projectIdsAsIntArray[i] = projectIds.get(i);
            }

            return projectIdsAsIntArray;
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
        return null;
    }

    /**
     * Checks whether given project name exists already in the account
     *
     * @param projectName
     *            name of the project name
     * @param accountId
     *            unique accountId given to an user
     * @return true if it exists, false if not
     */
    public static boolean projectExists(String projectName, int accountId) {
        Connection conn = getDBConnection();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);

            s = "SELECT Name FROM Projects WHERE Name = '" + projectName
                    + "' AND AccountID = '" + accountId + "';";
            ResultSet r = st.executeQuery(s);
            if (r.first()) {
                return true;
            }
            return false;
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
        return false;
    }

    /**
     * Adds new project
     *
     * @param project
     *            project property such as name, accountID
     * @return true if possible to make, false if not possible
     */
    public static boolean addProject(Project project) {
        Connection conn = getDBConnection();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);
            s = "INSERT INTO Projects (Name, AccountID, Icon) " + "VALUES ('"
                    + project.getName() + "', '" + project.getAccountId()
                    + "' , '" + project.getIcon() + "');";
            st.executeUpdate(s);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
        return false;
    }

    /**
     * Enables to rename a name of a project
     *
     * @param projectId
     *            unique project number given to a project
     * @param newProjectName
     *            new name of a project that wanted to be changed
     */
    public static boolean renameProject(int projectId, String newProjectName) {
        Connection conn = getDBConnection();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);
            s = "UPDATE Projects SET Name = '" + newProjectName
                    + "' WHERE ProjectID = '" + projectId + "';";
            st.executeUpdate(s);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
        return false;
    }

    /**
     * Enables to delete a project requested by an user
     *
     * @param projectId
     *            unique projectId after created by an user
     */
    public static void deleteProject(int projectId) {
        Connection conn = getDBConnection();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);

            // Delete all videos associated with the project.
            s = "DELETE FROM Videos WHERE ProjectID = '" + projectId + "';"; // TODO Add err
or checking
            st.executeUpdate(s);
```

```
            // Delete the project itself.
            s = "DELETE FROM Projects WHERE ProjectID = '" + projectId + "';"; // TODO Add e
rror checking
            st.executeUpdate(s);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
    }

    /**
     * Sets the position of projects in the project page
     *
     * @param projectId
     *            unique project number given to a project
     * @param position
     *            position of project in the page
     * @return true if changed, false if not
     */
    public static boolean setProjectPoolPosition(int projectId, int position) {
        String positionType = "PoolPosition";
        Connection conn = getDBConnection();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);
            s = "UPDATE Projects SET " + positionType + " = '" + position
                    + "' WHERE ProjectID = '" + projectId + "';";
            st.executeUpdate(s);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
        return false;
    }

    /**
     * Sets all position to where it is now which is set to be not organized
     *
     * @param accountId
     *            unique account number given to an user
     * @return true everything is set to -1 else false
     */
    public static boolean setAllProjectPoolPositionsToNull(int accountId) {
        Connection conn = getDBConnection();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);
            s = "SELECT ProjectID FROM Projects WHERE AccountID = '"
                    + accountId + "';";
            ResultSet r = st.executeQuery(s);
            if (r.first()) {
                do {
                    setProjectPoolPosition(r.getInt("ProjectID"), -1); // TODO Insert "NULL"
, not -1
                } while (r.next());
            }
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
        return false;
    }

    /**
     * check whether projects are not organized or not
     *
     * @param projectId
     *            unique project id given to a project
     * @return true if project is not organized, false if not.
     */
    public static boolean isProjectPoolPositionNotNull(int projectId) {
        int position = getProjectPoolPosition(projectId);
        if (position != -1) {
            return true;
        }
        return false;
    }

    /**
     * Enables to put project on the project page
     *
     * @param projectId
     *            unique project number given to a project
     * @return ordered projects
     */
    public static int getProjectPoolPosition(int projectId) {
        String projectPoolPosition = getUnknownValueFromTable("PoolPosition",
                "Projects", "ProjectID", Integer.toString(projectId));
        if (projectPoolPosition == null) {
            return -1; // TODO Is this a good idea?
        }
        return Integer.parseInt(projectPoolPosition);
    }

    /**
     * Brings the unique videoId given video name and its unique projectId
     *
     * @param videoName
     *            Name of the video that has been typed by an user
     * @param projectId
     *            unique ProjectId given after project's creation
     * @return corresponding videoId in the Olive database given videoName and projectId
     */
    // You don't need the accountId if you have the projectId. The projectId was
    // calculated using the accountId.
    public static int getVideoId(String videoName, int projectId) {
        Connection conn = getDBConnection();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);
            s = "SELECT VideoID FROM Videos WHERE Name = '" + videoName
                    + "' AND ProjectID = '" + projectId + "';";
            ResultSet r = st.executeQuery(s);
            int videoId = -1;
            if (r.first()) {
                videoId = r.getInt("VideoID");
            }
```

```
            return videoId;
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
        return -1;
    }

    /**
     * Brings the name of the video name given videoId
     *
     * @param videoId
     *            unique videoId for each individual video
     * @return Name of the video in the Olive database
     */
    public static String getVideoName(int videoId) {
        return getUnknownValueFromTable("Name", "Videos", "VideoID",
                Integer.toString(videoId));
    }

    /**
     * Brings the URL where video is located given videoId
     *
     * @param videoId
     *            unique videoId for each individual video
     * @return URL of the video listed in the Olive database
     */
    public static String getVideoUrl(int videoId) {
        return getUnknownValueFromTable("URL", "Videos", "VideoID",
                Integer.toString(videoId));
    }

    /**
     * Brings the video icon given videoId
     *
     * @param videoId
     *            unique videoId for each individual video
     * @return an icon corresponding to video and videoID
     */
    public static String getVideoIcon(int videoId) {
        return getUnknownValueFromTable("Icon", "Videos", "VideoID",
                Integer.toString(videoId));
    }

    /**
     * Brings the project given video Id
     *
     * @param videoId
     *            unique videoId for each individual video
     * @return projectId where video belongs to listed in the Olive database
     */
    public static int getVideoProjectId(int videoId) {
        return Integer.parseInt(getUnknownValueFromTable("ProjectID", "Videos",
                "VideoID", Integer.toString(videoId)));
    }

    /**
     * Check whether there is any video listed in the editing box
     *
     * @param videoId
     *            unique videoId for each individual video
     * @return true if there is nothing in the editing box, false if something exists
     */
    public static boolean isVideoPoolPositionNotNull(int videoId) {
        int position = getVideoPoolPosition(videoId);
        if (position != -1) {
            return true;
        }
        return false;
    }

    /**
     * Check whether there is any video listed in the time line
     *
     * @param videoId
     *            unique videoId for each individual video
     * @return true if there is nothing in the time line, false if something exists
     */
    public static boolean isVideoTimelinePositionNotNull(int videoId) {
        int position = getVideoTimelinePosition(videoId);
        if (position != -1) {
            return true;
        }
        return false;
    }

    /**
     * check the arrangement of the video in the editing box
     *
     * @param videoId
     *            unique videoId for each individual video
     * @return video’s order in the editing box
     */
    public static int getVideoPoolPosition(int videoId) {
        String videoPoolPosition = getUnknownValueFromTable("PoolPosition",
                "Videos", "VideoID", Integer.toString(videoId));
        if (videoPoolPosition == null) {
            return -1; // TODO Is this a good idea?
        }
        return Integer.parseInt(videoPoolPosition);
    }

    /**
     * check the arrangement of the video in the time line
     *
     * @param videoId
     *            unique videoId for each individual video
     * @return video’s order in the time line
     */
    public static int getVideoTimelinePosition(int videoId) {
        String videoTimelinePosition = getUnknownValueFromTable(
                "TimelinePosition", "Videos", "VideoID",
                Integer.toString(videoId));
        if (videoTimelinePosition == null) {
            return -1; // TODO Is this a good idea?
        }
        return Integer.parseInt(videoTimelinePosition);
    }

    /**
     * check whether video is selected or not
     *
     * @param videoId
     *            unique videoId for each individual video
     * @return true if selected, false if not
```

```
 */
public static boolean getVideoIsSelected(int videoId) {
    int isSelectedAsInt = Integer.parseInt(getUnknownValueFromTable(
            "IsSelected", "Videos", "VideoID", Integer.toString(videoId)));
    if (isSelectedAsInt == 0) {
        return false;
    }

    return true;
}

/**
 * Enables to select or unselect
 *
 * @param videoId
 *            unique videoId for each individual video
 * @param isSelected
 *            checker whether it is selected or not
 * @return true if selected, false if unselected
 */
private static boolean setVideoAsSelectedOrUnselected(int videoId,
        boolean isSelected) {
    int isSelectedAsInt;
    if (isSelected) {
        isSelectedAsInt = 1;
    } else {
        isSelectedAsInt = 0;
    }
    Connection conn = getDBConnection();
    try {
        Statement st = conn.createStatement();
        String s = "USE OliveData;";
        st.executeUpdate(s);
        s = "UPDATE Videos SET IsSelected = " + isSelectedAsInt
                + " WHERE VideoID = '" + videoId + "';";
        st.executeUpdate(s);
        return true;
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        closeConnection(conn);
    }
    return false;
}

/**
 * Sets status as selected
 *
 * @param videoId
 *            unique videoId for each individual video
 * @return status of selected
 */
public static boolean setVideoAsSelected(int videoId) {
    return setVideoAsSelectedOrUnselected(videoId, true);
}

/**
 * Sets status as unselected
 *
 * @param videoId
 *            unique videoId for each individual video
 * @return status of unselected
 */
public static boolean setVideoAsUnselected(int videoId) {
    return setVideoAsSelectedOrUnselected(videoId, false);
}

/**
 * Sets the position of an video in the time line
 *
 * @param videoId
 *            unique videoId for each individual video
 * @param position
 *            the order in the editing box/time line
 * @param positionType
 *            Pool or Timeline
 * @return true if it sets, false if it did not
 */
private static boolean setVideoPoolOrTimelinePosition(int videoId,
        int position, String positionType) {
    Connection conn = getDBConnection();
    try {
        Statement st = conn.createStatement();
        String s = "USE OliveData;";
        st.executeUpdate(s);
        s = "UPDATE Videos SET " + positionType + " = '" + position
                + "' WHERE VideoID = '" + videoId + "';";
        st.executeUpdate(s);
        return true;
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        closeConnection(conn);
    }
    return false;
}

/**
 * Sets the position of an video in the editing box
 *
 * @param videoId
 *            unique videoId for each individual video
 * @param position
 *            the order in the editing box/time line
 * @return sets the order
 */
public static boolean setVideoPoolPosition(int videoId, int position) {
    return setVideoPoolOrTimelinePosition(videoId, position, "PoolPosition");
}

/**
 * Sets the position of an video in the timeline
 *
 * @param videoId
 *            unique videoId for each individual video
 * @param position
 *            the order in the editing box/time line
 * @return sets the order
 */
public static boolean setTimelinePosition(int videoId, int position) {
    return setVideoPoolOrTimelinePosition(videoId, position,
            "TimelinePosition");
}

/**
 * Clears the position of video in the timeline
```

```
     *
     * @param projectId
     *            unique videoId for each individual video
     * @param positionType
     *            editing box or timeline
     * @return true if clears, false if not
     */
    public static boolean setAllVideoPoolOrTimelinePositionsToNull(
            int projectId, String positionType) {
        Connection conn = getDBConnection();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);
            s = "SELECT VideoID FROM Videos WHERE ProjectID = '" + projectId
                    + "';";
            ResultSet r = st.executeQuery(s);
            if (r.first()) {
                do {
                    setVideoPoolOrTimelinePosition(r.getInt("VideoID"), -1, // TODO Insert "
NULL", not -1
                            positionType);
                } while (r.next());
            }
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
        return false;
    }

    /**
     * Clears the position in the editing box
     *
     * @param projectId
     *            unique videoId for each individual video
     * @return clears editing box
     */
    public static boolean setAllVideoPoolPositionsToNull(int projectId) {
        return setAllVideoPoolOrTimelinePositionsToNull(projectId,
                "PoolPosition");
    }

    /**
     * Clears the position in the timeline
     *
     * @param projectId
     *            unique videoId for each individual video
     * @return clears timeline
     */
    public static boolean setAllVideoTimelinePositionsToNull(int projectId) {
        return setAllVideoPoolOrTimelinePositionsToNull(projectId,
                "TimelinePosition");
    }

    /**
     * brings the lists of videoIds in a given project
     *
     * @param projectId
     *            unique project number given to a project
     * @return lists of video ids in an integer array form
     */
    public static int[] getVideoIds(int projectId) {
        Connection conn = getDBConnection();
        List<Integer> videoIds = new LinkedList<Integer>();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);
            ResultSet r;
            s = "SELECT VideoID FROM Videos WHERE ProjectID = " + projectId
                    + ";";
            r = st.executeQuery(s);
            if (r.first()) {
                do {
                    videoIds.add(r.getInt("VideoID"));
                } while (r.next());
            }

            // Convert the List to an int array.
            int[] videoIdsAsIntArray = new int[videoIds.size()];
            for (int i = 0; i < videoIds.size(); ++i) {
                videoIdsAsIntArray[i] = videoIds.get(i);
            }

            return videoIdsAsIntArray;
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
        return null;
    }

    /**
     * Checks whether given video name exists already in the account
     *
     * @param videoName
     *            name of the video
     * @param projectId
     *            unique projectId given to a project
     * @return true if it exists, false if not
     */
    public static boolean videoExists(String videoName, int projectId) {
        Connection conn = getDBConnection();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);

            s = "SELECT Name FROM Videos WHERE Name = '" + videoName
                    + "' AND ProjectID = '" + projectId + "';";
            ResultSet r = st.executeQuery(s);
            if (r.first()) {
                return true;
            }
            return false;
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
        return false;
    }
```

```
/**
 * Adds video to the Olive database
 *
 * @param name
 *            name of the video
 * @param url
 *            where video is located
 * @param projectId
 *            unique project number
 * @param icon
 *            unique icon given to the video clip
 */
public static boolean addVideo(Video video) {
    Connection conn = getDBConnection();
    try {
        Statement st = conn.createStatement();
        String s = "USE OliveData;";
        st.executeUpdate(s);
        s = "INSERT INTO Videos (Name, URL, ProjectID, TimelinePosition,"
                + " Icon, IsSelected, PoolPosition) VALUES ('"
                + video.getName() + "', '" + video.getUrl() + "', '"
                + video.getProjectId() + "' , '"
                + video.getTimelinePosition() + "' , '" + video.getIcon()
                + "' , '" + (video.getIsSelected() ? 1 : 0) + "' , '"
                + video.getPoolPosition() + "');";
        st.executeUpdate(s);
        return true;
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        closeConnection(conn);
    }
    return false;
}

/**
 * Enables to rename a video
 *
 * @param videoId
 *            unique video number given to a video
 * @param newVideoName
 *            name of video user wants to rename
 */
public static boolean renameVideo(int videoId, String newVideoName) {
    Connection conn = getDBConnection();
    try {
        Statement st = conn.createStatement();
        String s = "USE OliveData;";
        st.executeUpdate(s);
        s = "UPDATE Videos SET Name = '" + newVideoName
                + "' WHERE VideoID = '" + videoId + "';";
        st.executeUpdate(s);
        return true;
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        closeConnection(conn);
    }
    return false;
}

/**
 * deletes an video
 *
 * @param videoId
 *            unique video id given a specific video clip
 */
public static void deleteVideo(int videoId) {
    Connection conn = getDBConnection();
    try {
        Statement st = conn.createStatement();
        String s = "USE OliveData;";
        st.executeUpdate(s);
        s = "DELETE FROM Videos WHERE VideoID = '" + videoId + "';"; // TODO Add error c
hecking
        st.executeUpdate(s);
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        closeConnection(conn);
    }
}

/**
 * Brings the credential information to connect to database
 *
 * @return credential properties
 */
public static AWSCredentials getAwsCredentials() {
    String awsAccessKeyPropertyName = "";
    String awsSecretKeyPropertyName = "";
    Connection conn = getDBConnection();
    try {
        Statement st = conn.createStatement();
        String s = "USE OliveData;";
        st.executeUpdate(s);
        s = "SELECT * FROM S3Credentials;";
        ResultSet r = st.executeQuery(s);
        if (r.first()) {
            awsAccessKeyPropertyName = r.getString("AWSAccessKey");
            awsSecretKeyPropertyName = r.getString("AWSSecretKey");
        } else {
            log.severe("Cannot locate AWS credentials");
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        closeConnection(conn);
    }

    return new AWSCredentials(awsAccessKeyPropertyName,
            awsSecretKeyPropertyName);
}

/**
 * gets Zencoder access key
 *
 * @return Zencoder api access key information
 */
public static String getZencoderApiKey() {
    String zencoderApiKey = "";
    Connection conn = DatabaseApi.getDBConnection();
    try {
        Statement st = conn.createStatement();
        String s = "USE OliveData;";
```

```
            st.executeUpdate(s);
            s = "SELECT * FROM ZencoderCredentials;";
            ResultSet r = st.executeQuery(s);
            if (r.first()) {
                zencoderApiKey = r.getString("ZencoderAPIKey");
            } else {
                log.severe("Cannot locate Zencoder credentials");
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }

        return zencoderApiKey;
    }

    /**
     * Checks whether input value of security question and answer of an user is correct
     *
     * @param username
     *            id of an user
     * @param securityQuestion
     *            security question made for recovery purpose
     * @param securityAnswer
     *            answer for the security question made
     * @return true if it matches, false if it does not
     */
    public static Boolean isCorrectSecurityInfo(String username,
            String securityQuestion, String securityAnswer) {
        Connection conn = getDBConnection();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);
            s = "SELECT AccountID FROM Accounts WHERE Username = '" + username
                    + "' AND SecurityQuestion = '" + securityQuestion
                    + "' AND SecurityAnswer = '" + securityAnswer + "';";
            ResultSet r = st.executeQuery(s);
            if (r.first()) {
                return true;
            }
            return false;
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
        return false; // Error!
    }

    /**
     * Modifies a password
     *
     * @param username
     *            id of an user
     * @param newPassword
     *            new password typed in
     * @return true if it modified, false if not
     */
    public static Boolean editPassword(String username, String newPassword) {
        Connection conn = getDBConnection();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);
            s = "UPDATE Accounts SET Password = Password('" + newPassword
                    + "') WHERE Username = '" + username + "';";
            st.executeUpdate(s);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
        return false;
    }

    /**
     * Enables to put videos on timeline
     *
     * @param projectId
     *            unique project number given to a project
     * @return lists of string ordered
     */
    public static String[] getVideosOnTimeline(int projectId) {
        Connection conn = getDBConnection();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);
            s = "SELECT * FROM Videos WHERE ProjectID = '" + projectId + "';";
            ResultSet r = st.executeQuery(s);
            ArrayList timelinePositionTemp = new ArrayList();
            if (r.first()) {
                do {
                    int temp = r.getInt("TimelinePosition");
                    if (temp != -1) {
                        timelinePositionTemp.add(temp);
                    }
                } while (r.next());
                Object[] timelinePositionO = timelinePositionTemp.toArray();
                Integer temp = null;
                int[] timelinePosition = new int[timelinePositionO.length];
                int i = 0;
                for (i = 0; i < timelinePositionO.length; i++) {
                    temp = (Integer) timelinePositionO[i];
                    timelinePosition[i] = temp.intValue();
                }
                Arrays.sort(timelinePosition);
                String[] result = new String[timelinePosition.length];
                for (i = 0; i < timelinePosition.length; i++) {
                    s = "SELECT Name FROM Videos WHERE ProjectID = '"
                            + projectId + "' AND TimelinePosition = "
                            + timelinePosition[i] + ";";
                    r = st.executeQuery(s);
                    if (r.first()) {
                        result[i] = r.getString("Name");
                    }
                }
                return result;

            } else {
                return null;
            }
        } catch (Exception e) {
```

```
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
        return null;
    }
}
```

```
package com.readytalk.olive.logic;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.security.NoSuchAlgorithmException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.logging.Logger;

import org.jets3t.service.S3ServiceException;
import org.jets3t.service.ServiceException;
import org.jets3t.service.acl.AccessControlList;
import org.jets3t.service.acl.EmailAddressGrantee;
import org.jets3t.service.acl.GroupGrantee;
import org.jets3t.service.acl.Permission;
import org.jets3t.service.impl.rest.httpclient.RestS3Service;
import org.jets3t.service.model.S3Object;
import org.jets3t.service.security.AWSCredentials;

import com.google.gson.Gson;
import com.readytalk.olive.model.Project;
import com.readytalk.olive.model.Video;
import com.readytalk.olive.servlet.OliveServlet;
import com.readytalk.olive.util.InvalidFileSizeException;

/**
 * class S3Api provides tool to connect to S3 from Olive
 *
 * @author Team Olive
 *
 */
// JetS3t CodeSamples.java: https://bitbucket.org/jmurty/jets3t/src/Release-0_8_0/src/org/je
ts3t/samples/CodeSamples.java
// JetS3t code-samples.html: http://jets3t.s3.amazonaws.com/toolkit/code-samples.html
// JetS3t JavaDocs: http://jets3t.s3.amazonaws.com/api/org/jets3t/service/model/StorageObjec
t.html
public class S3Api {
    private static final String BUCKET_NAME = "test-bucket-Olive";
    public static final String AWS_URL_PREFIX = "https://s3.amazonaws.com/"
            + BUCKET_NAME + "/";
    private static final String ZENCODER_AWS_EMAIL = "aws@zencoder.com";
    private static final long MAX_SIZE_IN_BYTES = 31457280L; // 30 MB
    private static final long MIN_SIZE_IN_BYTES = 1L; // ~0 MB
    private static final String DATE_FORMAT = "yyyy-MM-dd-HH-mm-ss-SSS";
    private static Logger log = Logger.getLogger(S3Api.class.getName());
    private static final int BUFFER_SIZE = 100000; // 100 KB
    private static final byte[] buffer = new byte[BUFFER_SIZE];

    /**
     * Constructs the service and initializes the properties.
     *
     * @return initialized properties
     * @throws S3ServiceException
     *             Exception for use by S3Services and related utilities. This exception can
 hold useful additional information about errors that occur when communicating with S3.
     */
    private static RestS3Service getS3Service() throws S3ServiceException {
        AWSCredentials awsCredentials = DatabaseApi.getAwsCredentials();
        // RestS3Service is similar to HttpClient
        RestS3Service s3Service = new RestS3Service(awsCredentials);
        return s3Service;
    }

    /**
     * Gets a calendar using the default time zone and locale
     *
     * @return simple data format of the calendar
     */
    // TODO Make this a hash function that uses the time
    public static String getTime() {
        Calendar calendar = Calendar.getInstance();
        SimpleDateFormat simpleDateFormat = new SimpleDateFormat(DATE_FORMAT);
        return simpleDateFormat.format(calendar.getTime());
    }

    /**
     * Enables to upload a file
     *
     * @param file
     *            video file
     * @return a converted file's location
     * @throws InvalidFileSizeException
     *             Exception for limiting the size of the video file
     * @throws IOException
     *             Signals that an I/O exception of some sort has occurred.
     * @throws ServiceException
     *             Exception for use by S3Services and related utilities. This exception can
 hold useful additional information about errors that occur when communicating with S3.
     * @throws NoSuchAlgorithmException
     *             This exception is thrown when a particular cryptographic algorithm is req
uested but is not available in the environment.
     */
    public static String[] uploadFile(File file)
            throws InvalidFileSizeException, IOException, ServiceException,
            NoSuchAlgorithmException {
        if (file.length() > MAX_SIZE_IN_BYTES) {
            throw new InvalidFileSizeException("File larger than "
                    + MAX_SIZE_IN_BYTES + " bytes");
        }
        if (file.length() < MIN_SIZE_IN_BYTES) {
            throw new InvalidFileSizeException("File smaller than "
                    + MIN_SIZE_IN_BYTES + " bytes");
        }

        S3Object fileAsS3Object = null;
        try {
            RestS3Service s3Service = getS3Service();

            // Set Content-Length automatically based on the file's extension.
            fileAsS3Object = new S3Object(file);

            String fileNameOnS3 = S3Api.getTime() + "-" + file.getName(); // TODO Make sure
this isn't too long.

            fileAsS3Object.setName(fileNameOnS3);

            // Extend the permissions already set by the bucket.
            AccessControlList acl = s3Service.getBucketAcl(BUCKET_NAME);
            acl.grantPermission(new EmailAddressGrantee(ZENCODER_AWS_EMAIL),
                    Permission.PERMISSION_FULL_CONTROL);
```

```
        acl.grantPermission(GroupGrantee.ALL_USERS,
                Permission.PERMISSION_READ);
        fileAsS3Object.setAcl(acl);

        // Upload the data object.
        s3Service.putObject(BUCKET_NAME, fileAsS3Object);

        String unconvertedVideoUrl = AWS_URL_PREFIX + fileNameOnS3;

        String[] videoUrlAndIcon = ZencoderApi
                .convertToOgg(unconvertedVideoUrl);
        deleteFileInS3(fileAsS3Object.getKey()); // Delete the unconverted version

        return videoUrlAndIcon;
    } catch (IOException e) {
        log.severe("Error connecting with S3");
        e.printStackTrace();
        throw new IOException(e.getMessage());
    } catch (S3ServiceException e) {
        log.severe("Error creating RestS3Service object");
        e.printStackTrace();
        throw new S3ServiceException(e.getMessage());
    } catch (NoSuchAlgorithmException e) {
        log.severe("Error creating S3Object object");
        e.printStackTrace();
        throw new NoSuchAlgorithmException(e.getMessage());
    } finally {
        fileAsS3Object.closeDataInputStream();
    }
}

/**
 * Enables to delete the file in S3
 *
 * @param objectKey
 *            usually corresponds to the name of the file
 */
public static void deleteFileInS3(String objectKey) {
    try {
        RestS3Service s3Service = getS3Service();
        s3Service.deleteObject(BUCKET_NAME, objectKey);
    } catch (S3ServiceException e) {
        log.severe("Error creating RestS3Service object");
        e.printStackTrace();
    } catch (ServiceException e) {
        log.severe("Error deleting object " + objectKey + " in bucket "
                + BUCKET_NAME);
        e.printStackTrace();
    }

}

/**
 * gets the URL address of S3 where video is located
 *
 * @param videoUrl
 *            where video is located in S3
 * @return
 */
public static String getNameFromUrl(String videoUrl) {
    return videoUrl.substring(AWS_URL_PREFIX.length());
}

/**
 * Gets a new time stamp on the URL where video is located
 *
 * @param videoUrl
 *            where video is located in S3
 * @return the time stamped URL
 */
public static String getNameFromUrlWithNewTimeStamp(String videoUrl) {
    return S3Api.getTime()
            + getTimeStampFromUrl(videoUrl);
}

/**
 * Gets the time stamped on the URL where video is located
 *
 * @param videoUrl
 *            where video is located in S3
 * @return the time stamped URL
 */
private static String getTimeStampFromUrl(String videoUrl) {
    return getNameFromUrl(videoUrl).substring(S3Api.getTime().length());
}

/**
 * Gets the project properties
 *
 * @param accountId
 *            unique number given to an user
 * @return project properties in forms of Json
 */
public static String getProjectInformation(int accountId) {
    int[] projectIds = DatabaseApi.getProjectIds(accountId);
    Project[] projects = new Project[projectIds.length];

    for (int projectIndex = 0; projectIndex < projectIds.length; ++projectIndex) {
        String projectName = DatabaseApi
                .getProjectName(projectIds[projectIndex]);
        String projectIcon = DatabaseApi
                .getProjectIcon(projectIds[projectIndex]);
        int poolPosition = DatabaseApi
                .getProjectPoolPosition(projectIds[projectIndex]);

        projects[projectIndex] = new Project(projectName, accountId,
                projectIcon, poolPosition);
    }

    return new Gson().toJson(projects);
}

/**
 * Gets the video properties
 *
 * @param projectId
 *            unique number given to a project
 * @return video properties in a Json form
 */
public static String getVideoInformation(int projectId) {
    int[] videoIds = DatabaseApi.getVideoIds(projectId);
    Video[] videos = new Video[videoIds.length];

    for (int videoIndex = 0; videoIndex < videoIds.length; ++videoIndex) {
        String videoName = DatabaseApi.getVideoName(videoIds[videoIndex]);
        String videoUrl = DatabaseApi.getVideoUrl(videoIds[videoIndex]);
```

```
            String videoIcon = DatabaseApi.getVideoIcon(videoIds[videoIndex]);
            int poolPosition = DatabaseApi
                    .getVideoPoolPosition(videoIds[videoIndex]);
            int timelinePosition = DatabaseApi
                    .getVideoTimelinePosition(videoIds[videoIndex]);
            boolean isSelected = DatabaseApi
                    .getVideoIsSelected(videoIds[videoIndex]);

            videos[videoIndex] = new Video(videoName, videoUrl, videoIcon,
                    projectId, poolPosition, timelinePosition, isSelected);
        }

        return new Gson().toJson(videos);
    }

    /**
     * Gets a file from S3 using URL address given, and puts on the dataInputStream
     *
     * @param videoUrl
     *            where video is located in S3
     * @return a video file
     * @throws IOException
     *             Signals that an I/O exception of some sort has occurred
     */
    public static File getFileFromS3(String videoUrl) throws IOException {
        String fileName = getNameFromUrl(videoUrl);
        S3Object s3Object = null;
        try {
            RestS3Service s3Service = getS3Service();

            s3Object = s3Service.getObject(BUCKET_NAME, fileName);

            return getFileFromInputStream(s3Object.getDataInputStream(),
                    fileName);
        } catch (S3ServiceException e) {
            log.severe("Error accessing object \"" + fileName
                    + "\" in S3 bucket \"" + BUCKET_NAME + "\"");
            e.printStackTrace();
        } catch (ServiceException e) {
            log.severe("Error accessing S3Object input stream for file \""
                    + fileName + "\" in S3 bucket \"" + BUCKET_NAME + "\"");
            e.printStackTrace();
        } finally {
            s3Object.closeDataInputStream();
        }
        log.severe("The function downloadFile returned null instead of a file");
        return null; // Error
    }

    /**
     * Gets file from input stream
     *
     * @param inputStream
     *            holds the video file that was in S3
     * @param name
     *            name of the video file
     * @return video file
     * @throws FileNotFoundException
     *             Signals that an attempt to open the file denoted by a specified pathname
has failed.
     * @throws IOException
     *             Signals that an I/O exception of some sort has occurred.
     */
    private static File getFileFromInputStream(InputStream inputStream,
            String name) throws FileNotFoundException, IOException {
        File file = new File(name);
        OutputStream out = new FileOutputStream(file);
        byte buffer[] = new byte[1024];
        int bufferLength;
        while ((bufferLength = inputStream.read(buffer)) > 0) {
            out.write(buffer, 0, bufferLength);
        }
        out.close();
        inputStream.close();
        return file;
    }

    /**
     *
     * @param videoUrl
     * @return
     * @throws IOException
     */
    // Modified from: http://msdn.microsoft.com/en-us/library/aa478985.aspx
    public static String downloadVideosToTemp(String videoUrl)
            throws IOException {
        File tempDir = OliveServlet.tempDir;
        File inFile = S3Api.getFileFromS3(videoUrl);
        File outFile = new File(tempDir, getNameFromUrl(videoUrl));
        outFile.deleteOnExit(); // Delete the file when the JVM exits (cannot be undone).
        S3Api.saveFileToDisk(inFile, outFile);
        return outFile.getName();
    }

    // Modified from: http://java.sun.com/docs/books/performance/1st_edition/html/JPIOPerfor
mance.fm.html#11078
    public static void saveFileToDisk(File from, File to) throws IOException {
        InputStream in = null;
        OutputStream out = null;
        try {
            in = new FileInputStream(from);
            out = new FileOutputStream(to);
            while (true) {
                synchronized (buffer) {
                    int amountRead = in.read(buffer);
                    if (amountRead == -1) {
                        break;
                    }
                    out.write(buffer, 0, amountRead);
                }
            }
        } finally {
            if (in != null) {
                in.close();
            }
            if (out != null) {
                out.close();
            }
        }
    }
}
```

```
package com.readytalk.olive.logic;

import java.io.File;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.apache.commons.fileupload.FileItem;

/**
 * class Security checks whether the inputs are valid or invalid
 *
 * @author Team Olive
 *
 */
// The regular expressions were taken from the JavaScript, but with A-Z appended
// to all a-z's, to account for case sensitivity that the JavaScript usually
// accounts for using /i at the end of the regular expression.
public class Security {

    private static final int MIN_USERNAME_LENGTH = 3;
    private static final int MAX_USERNAME_LENGTH = 16;
    private static final String SAFE_USERNAME_REGEX = "^[a-zA-z]([0-9a-zA-Z_])+$";

    private static final int MIN_EMAIL_LENGTH = 6;
    private static final int MAX_EMAIL_LENGTH = 64;
    // From jquery.validate.js (by joern), contributed by Scott Gonzalez: http://projects.sc
ottsplayground.com/email_address_validation/
    private static final String SAFE_EMAIL_REGEX = "^((([a-zA-z]|\\d|[!#\\$%&'\\*\\+\\-\\/=\
\?\\^_`{\\|}~]|[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\uFDF0-\\uFFEF])+(\\.([a-zA-z]|\\d|[!#\\$%&'\
\*\\+\\-\\/=\\?\\^_`{\\|}~]|[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\uFDF0-\\uFFEF])+)*)|((\\x22)(((
(\\x20|\\x09)*(\\x0d\\x0a))?(\\x20|\\x09)+)?(([\\x01-\\x08\\x0b\\x0c\\x0e-\\x1f\\x7f]|\\x21|
[\\x23-\\x5b]|[\\x5d-\\x7e]|[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\uFDF0-\\uFFEF])|(\\\\([\\x01-\\
x09\\x0b\\x0c\\x0d-\\x7f]|[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\uFDF0-\\uFFEF]))))*(((\\x20|\\x09
)*(\\x0d\\x0a))?(\\x20|\\x09)+)?(\\x22)))@((([a-zA-z]|\\d|[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\u
FDF0-\\uFFEF])|(([a-zA-z]|\\d|[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\uFDF0-\\uFFEF])([a-zA-z]|\\d|
-|\\.|_|~|[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\uFDF0-\\uFFEF])*([a-zA-z]|\\d|[\\u00A0-\\uD7FF\\u
F900-\\uFDCF\\uFDF0-\\uFFEF]))\\.)+(([a-zA-z]|[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\uFDF0-\\uFFE
F])|(([a-zA-z]|[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\uFDF0-\\uFFEF])([a-zA-z]|\\d|-|\\.|_|~|[\\u0
0A0-\\uD7FF\\uF900-\\uFDCF\\uFDF0-\\uFFEF])*([a-zA-z]|[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\uFDF0
-\\uFFEF]))\\.?$";

    private static final int MIN_PASSWORD_LENGTH = 5;
    private static final int MAX_PASSWORD_LENGTH = 128;
    private static final String SAFE_PASSWORD_REGEX = "^[a-zA-Z0-9_]$|^[a-zA-Z0-9_]+[a-zA-Z0
-9_ ]*[a-zA-Z0-9_]+$"; // Prevent spaces at the beginning or end, but don't require at least
 2 characters.

    private static final int MIN_NAME_LENGTH = 1;
    private static final int MAX_NAME_LENGTH = 32;
    private static final String SAFE_NAME_REGEX = SAFE_PASSWORD_REGEX;

    private static final int MIN_PROJECT_NAME_LENGTH = 1;
    private static final int MAX_PROJECT_NAME_LENGTH = 32;
    private static final String SAFE_PROJECT_NAME_REGEX = SAFE_PASSWORD_REGEX;
    private static final int MAXIMUM_NUMBER_OF_PROJECTS = 25;

    private static final int MIN_VIDEO_NAME_LENGTH = 1;
    private static final int MAX_VIDEO_NAME_LENGTH = 32;
    private static final String SAFE_VIDEO_NAME_REGEX = SAFE_PASSWORD_REGEX;
    private static final int MAXIMUM_NUMBER_OF_VIDEOS = 25;

    public static final double MIN_SPLIT_TIME_IN_SECONDS = 0; // ASSUME: No video is shorter
 than 0 seconds.
    public static final double MAX_SPLIT_TIME_IN_SECONDS = 14400; // ASSUME: No video is lon
ger than 4 hours.

    private static final int MIN_SECURITY_QUESTION_LENGTH = 1;
    private static final int MAX_SECURITY_QUESTION_LENGTH = 128;
    private static final String SAFE_SECURITY_QUESTION_REGEX = "^[0-9a-zA-Z .,]+[?.]*$";

    private static final int MIN_SECURITY_ANSWER_LENGTH = 1;
    private static final int MAX_SECURITY_ANSWER_LENGTH = 128;
    private static final String SAFE_SECURITY_ANSWER_REGEX = "^([0-9a-zA-Z .,])+$";

    // Why some characters are allowed:
    // http://stackoverflow.com/questions/2049502/what-characters-are-allowed-in-email-addre
ss
    private static final String[] ILLEGAL_STRINGS = { "!", "&", "*", "(", ")",
            "-", "=", "{", "}", "[", "]", "\\", "|", ";", "'", "\"", ":", ",",
            "<", ">", "/", "?", "`" };

    /**
     * Checks to see if the length of an input meets database and JavaScript length requirem
ent
     *
     * @param input
     *            user input
     * @param minLength
     *            minimum length allowed by database and Javascript
     * @param maxLength
     *            maximum length allowed by database and Javascript
     * @return the length has to meet the requirements of database and Javascript
     */
    // Input must match database and JavaScript length requirements!
    private static boolean isSafeLength(String input, int minLength,
            int maxLength) {
        return input != null && minLength <= input.length()
                && input.length() <= maxLength;
    }

    /**
     *
     * @param input
     * @param validRegex
     * @return
     */
    // Input must match database and JavaScript value requirements!
    // For Java regular expression syntax, see:
    // http://download.oracle.com/javase/1.4.2/docs/api/java/util/regex/Pattern.html
    private static boolean isSafeValue(String input, String validRegex) {
        Pattern pattern = Pattern.compile(validRegex);
        Matcher matcher = pattern.matcher(input);
        return matcher.matches();
    }

    public static boolean isSafeUsername(String username) {
        return isSafeLength(username, MIN_USERNAME_LENGTH, MAX_USERNAME_LENGTH)
                && isSafeValue(username, SAFE_USERNAME_REGEX);
    }

    public static boolean isSafeEmail(String email) {
        // TODO Make case-insensitive with /i: http://www.regular-expressions.info/javascrip
t.html
        return isSafeLength(email, MIN_EMAIL_LENGTH, MAX_EMAIL_LENGTH)
                && isSafeValue(email, SAFE_EMAIL_REGEX);
    }
```

```
    public static boolean isSafePassword(String password) {
        return isSafeLength(password, MIN_PASSWORD_LENGTH, MAX_PASSWORD_LENGTH)
                && isSafeValue(password, SAFE_PASSWORD_REGEX);
    }

    public static boolean isSafeName(String name) {
        return isSafeLength(name, MIN_NAME_LENGTH, MAX_NAME_LENGTH)
                && isSafeValue(name, SAFE_NAME_REGEX);
    }

    public static boolean isSafeProjectName(String projectName) {
        return isSafeLength(projectName, MIN_PROJECT_NAME_LENGTH,
                MAX_PROJECT_NAME_LENGTH)
                && isSafeValue(projectName, SAFE_PROJECT_NAME_REGEX);
    }

    public static boolean isUniqueProjectName(String projectName, int accountId) {
        return !DatabaseApi.projectExists(projectName, accountId);
    }

    public static boolean projectFits(int numberOfProjects) {
        return numberOfProjects < MAXIMUM_NUMBER_OF_PROJECTS;
    }

    public static boolean isSafeVideoName(String videoName) {
        return isSafeLength(videoName, MIN_VIDEO_NAME_LENGTH,
                MAX_VIDEO_NAME_LENGTH)
                && isSafeValue(videoName, SAFE_VIDEO_NAME_REGEX);
    }

    public static boolean isUniqueVideoName(String videoName, int projectId) {
        return !DatabaseApi.videoExists(videoName, projectId);
    }

    public static boolean videoFits(int numberOfVideos) {
        return numberOfVideos < MAXIMUM_NUMBER_OF_VIDEOS;
    }

    public static boolean isSafeSplitTimeInSeconds(double splitTimeInSeconds) {
        return splitTimeInSeconds > MIN_SPLIT_TIME_IN_SECONDS
                && splitTimeInSeconds < MAX_SPLIT_TIME_IN_SECONDS;
    }

    public static boolean isSafeSecurityQuestion(String securityQuestion) {
        return isSafeLength(securityQuestion, MIN_SECURITY_QUESTION_LENGTH,
                MAX_SECURITY_QUESTION_LENGTH)
                && isSafeValue(securityQuestion, SAFE_SECURITY_QUESTION_REGEX);
    }

    public static boolean isSafeSecurityAnswer(String securityAnswer) {
        return isSafeLength(securityAnswer, MIN_SECURITY_ANSWER_LENGTH,
                MAX_SECURITY_ANSWER_LENGTH)
                && isSafeValue(securityAnswer, SAFE_SECURITY_ANSWER_REGEX);
    }

    public static boolean isSafeProjectIcon(File projectIcon) {
        return false; // TODO Implement this
    }

    public static boolean isSafeVideoIcon(File videoIcon) {
        return false; // TODO Implement this
    }

    public static boolean isSafeVideo(FileItem video) {
        String contentType = video.getContentType();
        if (contentType == null) {
            return false;
        }
        String[] content = contentType.split("/");
        return content[0].equals("video")
                || (content[0].equals("audio") && content[1].equals("ogg"));
    }

    // The registration modal form does its own regular expression checking,
    // which should prevent any bad characters from getting in. This is just a
    // safety check for if the JavaScript got hacked and a bad character got in.
    public static String stripOutIllegalCharacters(String input) {
        String output = input;

        // Remove the *really* bad stuff (which cause XSS attacks and SQL
        // injections).
        for (int i = 0; i < ILLEGAL_STRINGS.length; ++i) {
            output = output.replace(ILLEGAL_STRINGS[i], "");
        }

        return output;
    }
}
```

```
package com.readytalk.olive.logic;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;

import com.google.gson.Gson;
import com.readytalk.olive.json.ZencoderInitialResponse;
import com.readytalk.olive.model.Video;

/**
 * Provides tool to use Zencoder from Olive
 *
 * @author Team Olive
 *
 */
// Modified from: http://www.exampledepot.com/egs/java.net/post.html
public class ZencoderApi {
    private static final String ZENCODER_API_JOBS_URL = "https://app.zencoder.com/api/jobs/"
;
    private static final String ZENCODER_API_OUTPUTS_URL_PREFIX = "https://app.zencoder.com/
api/outputs/";

    /**
     *
     * @param outputId
     * @throws MalformedURLException
     * @throws IOException
     */
    // Don't use id. See: http://zencoder.com/docs/api/#status
    private static void waitForJobToFinish(int outputId)
            throws MalformedURLException, IOException {
        String zencoderOutputsUrl = ZENCODER_API_OUTPUTS_URL_PREFIX + outputId
                + "/progress?api_key=" + DatabaseApi.getZencoderApiKey();

        // Example responses (in order):
        // {"current_event":"Transcoding","progress":"100.0","state":"processing"}
        // {"current_event":"Uploading","progress":"100.0","state":"processing"}
        // {"current_event":"Uploading","state":"finished"}

        while (!doGet(zencoderOutputsUrl).contains("\"state\":\"finished\"")) {
            System.out.println("Job " + outputId + " not done yet.");
            try {
                Thread.sleep(1000); // 1 second
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }

    // Modified from: http://download.oracle.com/javase/tutorial/networking/urls/readingWrit
ing.html
    private static String getResponse(InputStream inputStream)
            throws IOException {
        // Get the response
        BufferedReader bufferedReader = new BufferedReader(
                new InputStreamReader(inputStream));
        String line;
        String response = "";
        while ((line = bufferedReader.readLine()) != null) {
            response += line;
        }
        bufferedReader.close();
        System.out.println(response);
        return response;
    }

    public static String doGet(String url) throws IOException {
        return getResponse((new URL(url)).openConnection().getInputStream());
    }

    // Even though this method lives in the ZencoderApi class, it is generalized
    // for any type of POST request.
    // Modified from: http://download.oracle.com/javase/tutorial/networking/urls/readingWrit
ing.html
    public static String sendReceive(String data, URL url) throws IOException {
        // Send data
        System.out.println("Sending data to Zencoder...");
        URLConnection conn = url.openConnection();
        conn.setDoOutput(true);
        OutputStreamWriter outputStreamWriter = new OutputStreamWriter(
                conn.getOutputStream());
        outputStreamWriter.write(data);
        outputStreamWriter.flush();
        System.out.println("Data sent to Zencoder.");
        outputStreamWriter.close();

        return getResponse(conn.getInputStream());
    }

    private static String getJsonForSplit(String input, String baseUrl,
            String filename, double startClip, double clipLength,
            String thumbBaseUrl, String thumbPrefix, String thumbFormat) {
        String data = "{\"api_key\":\"" + DatabaseApi.getZencoderApiKey()
                + "\",\"input\":\"" + input + "\","
                + "\"output\":[{\"base_url\":\"" + baseUrl + "\","
                + "\"filename\":\"" + filename + "\"," + "\"thumbnails\":{"
                + "\"number\":1," + "\"base_url\":\"" + thumbBaseUrl + "\","
                + "\"prefix\":\"" + thumbPrefix + "\"," + "\"format\":\""
                + thumbFormat + "\"," + "\"public\":1" + "}," + "\"public\":1,"
                + "\"start_clip\":" + startClip + "," + "\"clip_length\":"
                + clipLength + "}]}";
        return data;
    }

    private static String getJsonForConvertToOgg(String input,
            String videoBaseUrl, String filename, String videoCodec,
            String audioCodec, String thumbBaseUrl, String thumbPrefix,
            String thumbFormat) {
        // Codec and file extension must match.
        String data = "{\"api_key\":\"" + DatabaseApi.getZencoderApiKey()
                + "\",\"input\":\"" + input + "\","
                + "\"output\":[{\"base_url\":\"" + videoBaseUrl + "\","
                + "\"filename\":\"" + filename + "\",\"video_codec\":\""
                + videoCodec + "\",\"audio_codec\":\"" + audioCodec + "\","
                + "\"thumbnails\":{" + "\"number\":1," + "\"base_url\":\""
                + thumbBaseUrl + "\"," + "\"prefix\":\"" + thumbPrefix + "\","
                + "\"format\":\"" + thumbFormat + "\"," + "\"public\":1" + "},"
                + "\"public\":1" + "}]}";
        return data;
```

```
    }

    public static Video[] split(int videoId, double splitTimeInSeconds)
            throws IOException {
        String originalVideoUrl = DatabaseApi.getVideoUrl(videoId);
        String awsBaseUrl = S3Api.AWS_URL_PREFIX;
        double maximumStartTimeInSeconds = Security.MIN_SPLIT_TIME_IN_SECONDS;
        double minimumEndTimeInSeconds = Security.MAX_SPLIT_TIME_IN_SECONDS;
        double[] splitStartInSeconds = { 0, splitTimeInSeconds };
        double[] clipLengthInSeconds = {
                splitTimeInSeconds - maximumStartTimeInSeconds,
                minimumEndTimeInSeconds - splitTimeInSeconds }; // Draw a picture to understand this.
        String thumbFormat = "jpg";
        Video[] videoFragments = new Video[2];
        String[] responses = new String[2];

        // Request the first and second halves of the original video.
        for (int i = 0; i < 2; ++i) {
            String videoFragmentFileName = S3Api
                    .getNameFromUrlWithNewTimeStamp(originalVideoUrl);
            String thumbPrefix = S3Api.getTime(); // The video name can't be included because it has a ".".
            responses[i] = ZencoderApi.sendReceive(ZencoderApi.getJsonForSplit(
                    originalVideoUrl, awsBaseUrl, videoFragmentFileName,
                    splitStartInSeconds[i], clipLengthInSeconds[i], awsBaseUrl,
                    thumbPrefix, thumbFormat), new URL(ZENCODER_API_JOBS_URL));
            String videoIcon = awsBaseUrl + thumbPrefix + "_0000."
                    + thumbFormat;
            videoFragments[i] = new Video(
                    DatabaseApi.getVideoName(videoId) + i, S3Api.AWS_URL_PREFIX
                            + videoFragmentFileName, videoIcon, -1, -1, -1,
                    false); // TODO Get icon from Zencoder
        }

        // Wait for the first and second halves of the original video.
        for (int i = 0; i < 2; ++i) {
            ZencoderInitialResponse zencoderInitialResponse = new Gson()
                    .fromJson(responses[i], ZencoderInitialResponse.class);
            waitForJobToFinish(zencoderInitialResponse.outputs[0].id);
        }

        return videoFragments;
    }

    public static String[] convertToOgg(String videoUrl) throws IOException {
        String awsBaseUrl = S3Api.AWS_URL_PREFIX;
        String newExtension = ".ogv";
        String convertedVideoFileName = S3Api
                .getNameFromUrlWithNewTimeStamp(videoUrl) + newExtension;
        String newVideoCodec = "theora";
        String newAudioCodec = "vorbis";
        String thumbPrefix = S3Api.getTime(); // The video name can't be included because it has a ".".
        String thumbFormat = "jpg";

        String response = ZencoderApi.sendReceive(
                getJsonForConvertToOgg(videoUrl, awsBaseUrl,
                        convertedVideoFileName, newVideoCodec, newAudioCodec,
                        awsBaseUrl, thumbPrefix, thumbFormat), new URL(
                        ZENCODER_API_JOBS_URL));
        ZencoderInitialResponse zencoderInitialResponse = new Gson().fromJson(
                response, ZencoderInitialResponse.class);
        waitForJobToFinish(zencoderInitialResponse.outputs[0].id);

        String[] videoUrlAndIcon = new String[] {
                awsBaseUrl + convertedVideoFileName,
                awsBaseUrl + thumbPrefix + "_0000." + thumbFormat };
        return videoUrlAndIcon;
    }
}
```

```
package com.readytalk.olive.model;
/**
 * class Project request the relevant information to user
 * according to its name, accountId, and icon.
 *
 * @author Team Olive
 *
 */

public class Project {

    private String name;
    private int accountId;
    private String icon;
    private int poolPosition;

/**
 * Project Constructor
 * @param name          a name of the Project folder
 * @param accountId     a unique number distinguishes the different user which enables to sh
ow specific user's projects list
 * @param icon          an icon that is specifically designated to a project
 * @param poolPosition  position of a video in the editing box
 */

    public Project(String name, int accountId, String icon, int poolPosition) {
        this.name = name;
        this.accountId = accountId;
        this.icon = icon;
        this.poolPosition = poolPosition;
    }
/**
 * In use of displaying the name of a project for developers for debugging purpose
 *
 * @return name of a project
 */
    @Override
    public String toString() {
        return name;
    }
/**
 * calls the specific name of a project
 *
 * @return name of a project
 */
    public String getName() {
        return name;
    }
/**
 * Sets the name of a project with the name that is typed by an user.
 * @param name name of a project
 */
    public void setName(String name) {
        this.name = name;
    }
/**
 * Calls specific an user account by their unique id registered in the Olive Database
 *
 * @return unique user account id which enables the web to display correct project lists pag
e.
 */
    public int getAccountId() {
        return accountId;
    }
/**
 *
 *
 * @param accountId
 */
    public void setAccountId(int accountId) {
        this.accountId = accountId;
    }
/**
 * Calls the project icon that represents the project.
 *
 * @return icon an image representing a project
 */
    public String getIcon() {
        return icon;
    }
/**
 * Sets the icon to the corresponding project
 *
 * @param icon an image representing a project
 */
    public void setIcon(String icon) {
        this.icon = icon;
    }
/**
 * sets the video position in the editing box (pool)
 * @param poolPosition
 */
    public void setPoolPosition(int poolPosition) {
        this.poolPosition = poolPosition;
    }
/**
 *
 * @return
 */
    public int getPoolPosition() {
        return poolPosition;
    }
}
```

```
package com.readytalk.olive.model;
/**
 * class User holds the information of an user to log in and some additional information for
 recovery.
 *
 * @author Team Olive
 *
 */

public class User {

    private String username;
    private String password;
    private String name;
    private String email;
    private String securityQuestion;
    private String securityAnswer;

    /**
     * User Constructor for Log In and Sign Up.
     *
     * @param username Identification key used for log in
     * @param password Password key used for log in
     * @param name     Name of the user
     * @param email    E-mail address used by an user
     */
    public User(String username, String password, String name, String email) {
        this.username = username;
        this.password = password;
        this.name = name;
        this.email = email;
    }

    /**
     * User Constructor for Edit Account Information
     *
     * @param username Identification key used for log in
     * @param password Password key used for log in
     * @param name     Name of the user
     * @param email    E-mail address used by an user
     * @param securityQuestion Question used for recovering password
     * @param securityAnswer   The answer for the security question
     */
    public User(String username, String password, String name,
            String email, String securityQuestion, String securityAnswer) {
        this.username = username;
        this.password = password;
        this.name = name;
        this.email = email;
        this.setSecurityQuestion(securityQuestion);
        this.setSecurityAnswer(securityAnswer);
    }
    /**
     * In use of displaying the username of a project for developers for debugging purpose
     *
     * @return name of an user
     */
    @Override
    public String toString() {
        return username;
    }
    /**
     * calls the specific user name that is registered through sign up.
     *
     * @return user name registered by an user
     */
    public String getUsername() {
        return username;
    }

    /**
     * Registers the name of an user
     *
     * @param username user name registered by an user
     */
    public void setUsername(String username) {
        this.username = username;
    }

    /**
     * Brings the information for the password that has been set by an user
     *
     * @return password key stored in Olive database
     */
    public String getPassword() {
        return password;
    }

    /**
     * Registers user password that has been passed by user input
     *
     * @param password Secret Key for an user that is stored in
     */
    public void setPassword(String password) {
        this.password = password;
    }

    /**
     * Registers name of an user in the edit account information
     *
     * @param name name of an user registered in the edit account information
     */
    public void setName(String name) {
        this.name = name;
    }

    /**
     * Brings the default name of an user listed in the edit account information
     *
     * @return name of an user in the edit account information
     */
    public String getName() {
        return name;
    }

    /**
     * Registers or changes email of an user in the edit account information
     *
     * @param email email address of an user
     */
    public void setEmail(String email) {
        this.email = email;
    }
    /**
     * Brings the information regarding email address of an user
     * @return email address registered by an user
```

```
     */
    public String getEmail() {
        return email;
    }
    /**
     * Registers a security question for recovering purpose
     * @param securityQuestion question asked later for recovering purpose
     */
    public void setSecurityQuestion(String securityQuestion) {
        this.securityQuestion = securityQuestion;
    }
    /**
     * Brings the information regarding security question stored by an user
     * @return question asked later for recovering purpose
     */
    public String getSecurityQuestion() {
        return securityQuestion;
    }
    /**
     * Registers the answer for security question stored by an user
     * @param securityAnswer an answer for security question
     */
    public void setSecurityAnswer(String securityAnswer) {
        this.securityAnswer = securityAnswer;
    }
    /**
     * Brings an information regarding an answer of a security question stored by an user
     * @return security answer for security question
     */
    public String getSecurityAnswer() {
        return securityAnswer;
    }
}
```

```
package com.readytalk.olive.model;
/**
 * class Video holds information to show basic platform of editing page such as showing vide
o icon after uploading a video, and moving the icons in time line and also in the editing bo
x.
 *
 * @author Team Olive
 *
 */
public class Video {

    private String name;
    private String url;
    private String icon;
    private int projectId;
    private int poolPosition;
    private int timelinePosition;
    private boolean isSelected;
/**
 * Video Constructor
 * @param name name of the project
 * @param url a location of a video which an user wants to upload
 * @param icon icon that represents a video clip
 * @param projectId unique identity key belonging to a project
 * @param poolPosition ordering of video clips in the editing box located top left of UI
 * @param timelinePosition ordering of video clips in the time line
 * @param isSelected check whether an icon is clicked or not
 */
    public Video(String name, String url, String icon, int projectId,
            int poolPosition, int timelinePosition, boolean isSelected) {
        this.name = name;
        this.url = url;
        this.icon = icon;
        this.projectId = projectId;
        this.poolPosition = poolPosition;
        this.timelinePosition = timelinePosition;
        this.isSelected = isSelected;
    }
/**
 * Brings the name of a video clip that has been set by an user
 * @return name of a video clip
 */
    public String getName() {
        return name;
    }
/**
 * Registers a name of a video that is uploaded by an user
 * @param name
 */
    public void setName(String name) {
        this.name = name;
    }
/**
 * Brings an url address where a video user wants to up
 * @return url address
 */
    public String getUrl() {
        return url;
    }
    /**
     * Registers an url address where a video located which an user wants to upload
     * @param url passed in through the uploading interface
     */
    public void setUrl(String url) {
        this.url = url;
    }
    /**
     * Brings an icon designated to a video clip
     * @return icon representing a video clip
     */
    public String getIcon() {
        return icon;
    }
    /**
     * Registers an icon to a uploaded video
     * @param icon an icon representing video on the user interface
     */
    public void setIcon(String icon) {
        this.icon = icon;
    }
    /**
     * Brings the key represents specific project page
     * @return the key presents project
     */
    public int getProjectId() {
        return projectId;
    }
    /**
     * Classifies project folder by designating specific project key
     * @param project unique project key according to which project folder an user is in
     */
    public void setProjectId(int project) {
        this.projectId = project;
    }
    /**
     * Registers the arrangement where a video clip icon has to be located in the editing bo
x of the user interface
     * @param poolPosition the order number of a video in the editing box
     */
    public void setPoolPosition(int poolPosition) {
        this.poolPosition = poolPosition;
    }
    /**
     * Brings the arrangement where a video clip icon has to be located in the editing box o
f user interface
     * @return the order number of a video in the editing box
     */
    public int getPoolPosition() {
        return poolPosition;
    }
    /**
     * Registers a location of a video in the time line
     * @param timelinePosition
     */
    public void setTimelinePosition(int timelinePosition) {
        this.timelinePosition = timelinePosition;
    }
    /**
     * Brings the information regarding the arrangement of a video in the timeline
     * @return a video clip ordering in the timeline
     */
    public int getTimelinePosition() {
        return timelinePosition;
    }
    /**
     * Changes video icon’s status to be selected
```

```
     * @param isSelected status saying whether a video clip is clicked or not
     */
    public void setIsSelected(boolean isSelected) {
        this.isSelected = isSelected;
    }
    /**
     * Returns its status whether a video clip is selected or not
     * @return whether a video clip is clicked or not
     */
    public boolean getIsSelected() {
        return isSelected;
    }

}
```

```
package com.readytalk.olive.servlet;

import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.security.NoSuchAlgorithmException;
import java.util.Iterator;
import java.util.List;
import java.util.logging.Logger;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.FileUploadException;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;
import org.jets3t.service.ServiceException;

import com.google.gson.Gson;
import com.readytalk.olive.json.AddToSelectedRequest;
import com.readytalk.olive.json.CombineVideosRequest;
import com.readytalk.olive.json.DeleteAccountRequest;
import com.readytalk.olive.json.DeleteProjectRequest;
import com.readytalk.olive.json.DeleteVideoRequest;
import com.readytalk.olive.json.GeneralRequest;
import com.readytalk.olive.json.GetAccountInformationResponse;
import com.readytalk.olive.json.RemoveFromSelectedRequest;
import com.readytalk.olive.json.RenameProjectRequest;
import com.readytalk.olive.json.RenameVideoRequest;
import com.readytalk.olive.json.SplitVideoRequest;
import com.readytalk.olive.json.UpdateProjectsPositionRequest;
import com.readytalk.olive.json.UpdateTimelinePositionRequest;
import com.readytalk.olive.json.UpdateVideosPositionRequest;
import com.readytalk.olive.logic.ZencoderApi;
import com.readytalk.olive.logic.DatabaseApi;
import com.readytalk.olive.logic.S3Api;
import com.readytalk.olive.logic.Security;
import com.readytalk.olive.model.Project;
import com.readytalk.olive.model.User;
import com.readytalk.olive.model.Video;
import com.readytalk.olive.util.Attribute;
import com.readytalk.olive.util.InvalidFileSizeException;

/**
 * class OliveServlet
 *
 * @author Team Olive
 *
 */
public class OliveServlet extends HttpServlet {
    // Don't store anything as a member variable in the Servlet.
    // private Object dontDoThis;

    // Generated using Eclipse's "Add generated serial version ID" refactoring.
    private static final long serialVersionUID = -6820792513104430238L;
    // Static variables are okay, though, because they don't change across instances.
    private static Logger log = Logger.getLogger(OliveServlet.class.getName());
    public static final String TEMP_DIR_PATH = "/temp/"; // TODO Make a getter for this.
    public static File tempDir; // TODO Make a getter for this.
    public static final String DESTINATION_DIR_PATH = "/temp/"; // TODO Make a getter for th
is.
    public static File destinationDir; // TODO Make a getter for this.

    // Modified from: http://www.jsptube.com/servlet-tutorials/servlet-file-upload-example.h
tml
    // Also see: http://stackoverflow.com/questions/4101960/storing-image-using-htm-input-ty
pe-file
    /**
     *
     */
    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        createTempDirectories();
    }

    private void createTempDirectories() throws ServletException {
        String realPathTemp = getServletContext().getRealPath(TEMP_DIR_PATH);
        tempDir = new File(realPathTemp);
        if (!tempDir.isDirectory()) {
            throw new ServletException(TEMP_DIR_PATH + " is not a directory");
        }
        String realPathDest = getServletContext().getRealPath(
                DESTINATION_DIR_PATH);
        destinationDir = new File(realPathDest);
        if (!destinationDir.isDirectory()) {
            throw new ServletException(DESTINATION_DIR_PATH
                    + " is not a directory");
        }
    }

    private int getAccountIdFromSessionAttributes(HttpSession session) {
        String sessionUsername = (String) session
                .getAttribute(Attribute.USERNAME.toString());
        return DatabaseApi.getAccountId(sessionUsername);
    }

    private int getProjectIdFromSessionAttributes(HttpSession session) {
        String sessionUsername = (String) session
                .getAttribute(Attribute.USERNAME.toString());
        int accountId = DatabaseApi.getAccountId(sessionUsername);
        String sessionProjectName = (String) session
                .getAttribute(Attribute.PROJECT_NAME.toString());
        int projectId = DatabaseApi.getProjectId(sessionProjectName, accountId);
        return projectId;
    }

    private int getProjectIdFromSessionAttributes(HttpSession session,
            String projectName) {
        return DatabaseApi.getProjectId(projectName,
                getAccountIdFromSessionAttributes(session));
    }

    private int getVideoIdFromSessionAttributes(HttpSession session,
```

```
            String videoName) {
        return DatabaseApi.getVideoId(videoName,
                getProjectIdFromSessionAttributes(session));
    }

    @Override
    protected void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOException {
        HttpSession session = request.getSession();
        if (request.getContentType().contains(
                "application/x-www-form-urlencoded")) { // Full value: "application/x-www-fo
rm-urlencoded"
            // This is a regular text form.
            String id = request.getParameter("FormName");
            log.info("The servlet is responding to an "
                    + "HTTP POST request from form: " + id);
            if (id.equals("LoginUser")) {
                handleLogin(request, response, session);
            } else if (id.equals("EditUser")) {
                handleEditUser(request, response, session);
            } else if (id.equals("AddUser")) {
                handleAddUser(request, response, session);
            } else if (id.equals("AddProject")) {
                handleAddProject(request, response, session);
            } else if (id.equals("security-question-form")) {
                handleSecurityQuestionRetrieval(request, response, session);
            } else if (id.equals("security-question-form-2")) {
                handleSecurityAnswer(request, response, session);
            } else if (id.equals("new_password")) {
                handleNewPassword(request, response, session);
            } else if (id.equals("combine-form")) {
                try {
                    handleCombineVideos(request, response, session,"");
                } catch (NoSuchAlgorithmException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (InvalidFileSizeException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (ServiceException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            } else {
                log.severe("HTTP POST request coming from unknown form: " + id);
            }
        } else if (request.getContentType().contains("multipart/form-data")) { // Full value
: "multipart/form-data; boundary=----WebKitFormBoundaryjAGjLWGWeI3ltfBe"
            // This is a file upload form.
            log.info("The servlet is responding to an "
                    + "HTTP POST request from a file upload form");
            handleUploadVideo(request, response, session);
        } else if (request.getContentType().contains("application/json")) {
            // This is not a form, but a custom POST request with JSON in it.
            log.info("The servlet is responding to an "
                    + "HTTP POST request in JSON format");
            try {
                handleJsonPostRequest(request, response, session);
            } catch (NoSuchAlgorithmException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InvalidFileSizeException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (ServiceException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        } else {
            log.severe("Unknown content type");
        }
    }

    private void handleNewPassword(HttpServletRequest request,
            HttpServletResponse response, HttpSession session)
            throws UnsupportedEncodingException, IOException {
        // TODO Auto-generated method stub
        String newPassword = request.getParameter("password");
        String confirmNewPassword = request.getParameter("confirm_password");
        Boolean newPasswordSet;
        if (Security.isSafePassword(newPassword)
                && Security.isSafePassword(confirmNewPassword)) {
            session.setAttribute(Attribute.IS_SAFE.toString(), true);
            if (newPassword.equals(confirmNewPassword)) {
                session.setAttribute(Attribute.PASSWORDS_MATCH.toString(), true);
                String username = (String) session
                        .getAttribute(Attribute.USERNAME.toString());
                newPasswordSet = DatabaseApi
                        .editPassword(username, newPassword);
                session.setAttribute(Attribute.EDIT_SUCCESSFULLY.toString(),
                        newPasswordSet);
            } else {
                session.setAttribute(Attribute.PASSWORDS_MATCH.toString(),
                        false);
                session.setAttribute(Attribute.EDIT_SUCCESSFULLY.toString(),
                        false);
            }
        } else {
            session.setAttribute(Attribute.IS_SAFE.toString(), false);
            session.setAttribute(Attribute.EDIT_SUCCESSFULLY.toString(), false);
        }
        response.sendRedirect("new-password-form.jsp");
        session.removeAttribute(Attribute.USERNAME.toString());
    }

    private void handleSecurityQuestionRetrieval(HttpServletRequest request,
            HttpServletResponse response, HttpSession session)
            throws UnsupportedEncodingException, IOException {
        // TODO Auto-generated method stub
        String username = request.getParameter("username");
        if (Security.isSafeUsername(username)) {
            session.setAttribute(Attribute.IS_SAFE.toString(), true);
            if (DatabaseApi.usernameExists(username)) {
                String securityQuestion = DatabaseApi
                        .getAccountSecurityQuestion(DatabaseApi
                                .getAccountId(username));
                if (securityQuestion != null) {
                    session.setAttribute(
                            Attribute.SECURITY_QUESTION.toString(),
                            securityQuestion);
```

```
                    session.setAttribute(Attribute.USERNAME.toString(),
                            username);
                    session.removeAttribute(Attribute.IS_SAFE.toString()); // Cleared so as
to not interfere with any other form.
                    response.sendRedirect("securityQuestion.jsp");
                } else {
                    session.setAttribute(Attribute.IS_CORRECT.toString(), false);
                    response.sendRedirect("forgot.jsp");
                }
            } else {
                session.setAttribute(Attribute.IS_CORRECT.toString(), false);
                response.sendRedirect("forgot.jsp");
            }
        } else {
            session.setAttribute(Attribute.IS_SAFE.toString(), false);
            session.setAttribute(Attribute.IS_CORRECT.toString(), false);
            response.sendRedirect("forgot.jsp");
        }
    }

    private void handleSecurityAnswer(HttpServletRequest request,
            HttpServletResponse response, HttpSession session)
            throws UnsupportedEncodingException, IOException {
        // TODO Auto-generated method stub
        String answer = request.getParameter("security_answer");
        String username = (String) session.getAttribute(Attribute.USERNAME
                .toString());
        if (Security.isSafeSecurityAnswer(answer)) {
            session.setAttribute(Attribute.IS_SAFE.toString(), true);
            String securityQuestion = DatabaseApi
                    .getAccountSecurityQuestion(DatabaseApi
                            .getAccountId(username));
            Boolean isCorrect = DatabaseApi.isCorrectSecurityInfo(username,
                    securityQuestion, answer);
            if (isCorrect) {
                session.setAttribute(Attribute.IS_CORRECT.toString(), true);
                session.removeAttribute(Attribute.IS_SAFE.toString()); // Cleared so as to n
ot interfere with any other form.
                response.sendRedirect("new-password-form.jsp");
            } else {
                session.setAttribute(Attribute.IS_CORRECT.toString(), false);
                response.sendRedirect("securityQuestion.jsp");
            }
        } else {
            session.setAttribute(Attribute.IS_SAFE.toString(), false);
            session.setAttribute(Attribute.IS_CORRECT.toString(), false);
            response.sendRedirect("securityQuestion.jsp");
        }
    }

    // http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/Servlet-Tutorial-Form-Data.html
    @Override
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOException {
        log.info("The servlet is responding to an HTTP GET request");
        response.setContentType("text/html");
        HttpSession session = request.getSession();
        String projectName = request.getParameter("projectName");
        int accountId = DatabaseApi.getAccountId((String) session
                .getAttribute(Attribute.USERNAME.toString()));
        if (projectName != null && Security.isSafeProjectName(projectName)
                && DatabaseApi.projectExists(projectName, accountId)) { // Short-circuiting
            session.setAttribute(Attribute.PROJECT_NAME.toString(), projectName);
            response.sendRedirect("editor.jsp");
        } else {
            response.sendRedirect("projects.jsp");
        }
        PrintWriter out = response.getWriter();
        out.println("File uploaded. Please close this window and refresh the editor page.");
        out.close();
    }

    private void handleLogin(HttpServletRequest request,
            HttpServletResponse response, HttpSession session)
            throws UnsupportedEncodingException, IOException {
        Boolean isAuthorized;
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        if (Security.isSafeUsername(username)
                && Security.isSafePassword(password)) {
            session.setAttribute(Attribute.IS_SAFE.toString(), true);
            isAuthorized = DatabaseApi.isAuthorized(username, password);
            session.setAttribute(Attribute.IS_AUTHORIZED.toString(),
                    isAuthorized);
            if (isAuthorized) { // Take the user to the projects page.
                int accountId = DatabaseApi.getAccountId(username);
                session.setAttribute(Attribute.USERNAME.toString(),
                        DatabaseApi.getAccountUsername(accountId));
                session.setAttribute(Attribute.PASSWORD.toString(), password);
                session.setAttribute(Attribute.EMAIL.toString(),
                        DatabaseApi.getAccountEmail(accountId));
                session.setAttribute(Attribute.NAME.toString(),
                        DatabaseApi.getAccountName(accountId));
                session.setAttribute(Attribute.IS_FIRST_SIGN_IN.toString(),
                        false);
                session.removeAttribute(Attribute.IS_SAFE.toString()); // Cleared so as to n
ot interfere with any other form.
                response.sendRedirect("projects.jsp");
            } else {
                response.sendRedirect("index.jsp"); // Keep the user on the same page.
            }
        } else {
            session.setAttribute(Attribute.IS_SAFE.toString(), false);
            session.setAttribute(Attribute.IS_AUTHORIZED.toString(), false);
            response.sendRedirect("index.jsp");
        }
    }

    private void handleEditUser(HttpServletRequest request,
            HttpServletResponse response, HttpSession session)
            throws UnsupportedEncodingException, IOException {
        String username = (String) session.getAttribute(Attribute.USERNAME
                .toString());
        String newName = request.getParameter("new-name");
        String newEmail = request.getParameter("new-email");
        String newPassword = request.getParameter("new-password");
        String confirmNewPassword = request
                .getParameter("confirm-new-password");
        String securityQuestion = request.getParameter("new-security-question");
        String securityAnswer = request.getParameter("new-security-answer");
        if (Security.isSafeName(newName) && Security.isSafeEmail(newEmail)
                && Security.isSafePassword(newPassword)
                && Security.isSafePassword(confirmNewPassword)
                && Security.isSafeSecurityQuestion(securityQuestion)
                && Security.isSafeSecurityAnswer(securityAnswer)) {
            if (newPassword.equals(confirmNewPassword)) {
```

```
            User updateUser = new User(username, newPassword, newName,
                    newEmail, securityQuestion, securityAnswer);
            Boolean editSuccessfully = DatabaseApi.editAccount(updateUser);
            session.setAttribute(Attribute.EDIT_SUCCESSFULLY.toString(),
                    editSuccessfully);
            session.setAttribute(Attribute.PASSWORDS_MATCH.toString(), true);
            session.setAttribute(Attribute.PASSWORD.toString(), newPassword);
            session.setAttribute(Attribute.EMAIL.toString(), newEmail);
            session.setAttribute(Attribute.NAME.toString(), newName);
            session.setAttribute(Attribute.SECURITY_QUESTION.toString(),
                    securityQuestion);
            session.setAttribute(Attribute.SECURITY_ANSWER.toString(),
                    securityAnswer);
        } else {
            session.setAttribute(Attribute.EDIT_SUCCESSFULLY.toString(),
                    false);
            session.setAttribute(Attribute.PASSWORDS_MATCH.toString(),
                    false);
        }
    } else {
        session.setAttribute(Attribute.EDIT_SUCCESSFULLY.toString(), false);
    }
    response.sendRedirect("account.jsp");
}

private void handleAddUser(HttpServletRequest request,
        HttpServletResponse response, HttpSession session)
        throws IOException {
    // The jQuery regex should catch malicious input, but sanitize just to
    // be safe.
    String username = Security.stripOutIllegalCharacters(request
            .getParameter("name"));
    String password = Security.stripOutIllegalCharacters(request
            .getParameter("password"));
    String email = Security.stripOutIllegalCharacters(request
            .getParameter("email"));
    User newUser = new User(username, password, "", email);
    Boolean addSuccessfully = DatabaseApi.AddAccount(newUser);
    if (addSuccessfully) {
        session.setAttribute(Attribute.IS_AUTHORIZED.toString(), true);
        session.setAttribute(Attribute.USERNAME.toString(), username);
        session.setAttribute(Attribute.PASSWORD.toString(), password);
        session.setAttribute(Attribute.EMAIL.toString(), email);
        session.setAttribute(Attribute.IS_FIRST_SIGN_IN.toString(), true);
        response.sendRedirect("projects.jsp");
    } else {
        response.sendRedirect("index.jsp");
        // TODO Add error message here
    }
}

private void handleAddProject(HttpServletRequest request,
        HttpServletResponse response, HttpSession session)
        throws UnsupportedEncodingException, IOException {
    int accountId = getAccountIdFromSessionAttributes(session);
    String projectName = request.getParameter("new-project-name");
    if (Security.isSafeProjectName(projectName)
            && Security.isUniqueProjectName(projectName, accountId)
            && Security.projectFits(DatabaseApi
                    .getNumberOfProjects(accountId))) {
        session.setAttribute(Attribute.IS_SAFE.toString(), true);

        String icon = ""; // TODO Get this from user input.
        Project project = new Project(projectName, accountId, icon, -1);
        Boolean added = DatabaseApi.addProject(project);
        if (!added) {
            session.setAttribute(Attribute.ADD_SUCCESSFULLY.toString(),
                    false);
        } else {
            session.setAttribute(Attribute.ADD_SUCCESSFULLY.toString(),
                    true);
            session.setAttribute(Attribute.IS_FIRST_SIGN_IN.toString(),
                    false);
        }
    } else {
        session.setAttribute(Attribute.IS_SAFE.toString(), false);
    }
    response.sendRedirect("projects.jsp");
}

private void handleUploadVideo(HttpServletRequest request,
        HttpServletResponse response, HttpSession session)
        throws IOException {
    PrintWriter out = response.getWriter();
    out.println("Uploading file...");

    response.setContentType("text/plain");

    DiskFileItemFactory fileItemFactory = new DiskFileItemFactory();
    // Set the size threshold, above which content will be stored on disk.
    fileItemFactory.setSizeThreshold(1 * 1024 * 1024); // 1 MB

    // Set the temporary directory to store the uploaded files of size above threshold.
    fileItemFactory.setRepository(tempDir);

    ServletFileUpload uploadHandler = new ServletFileUpload(fileItemFactory);
    File file = null;
    try {
        /*
         * Parse the request
         */
        List items = uploadHandler.parseRequest(request);
        Iterator itr = items.iterator();

        /*
         * The two items in the form
         */
        FileItem videoNameItem = null;
        FileItem fileItem = null;
        while (itr.hasNext()) {
            FileItem item = (FileItem) itr.next();
            /*
             * Handle Form Fields.
             */
            if (item.isFormField()
                    && item.getFieldName().equals("new-video-name")) { // Short-circuitr
y
                // Handle text fields
                log.info("Form Name = \"" + item.getFieldName()
                        + "\", Value = \"" + item.getString() + "\"");
                videoNameItem = item;
            } else {
                // Handle Uploaded files.
                log.info("Field Name = \"" + item.getFieldName()
                        + "\", File Name = \"" + item.getName()
                        + "\", Content type = \""
```

```
                            + item.getContentType() // TODO Save this
                            + "\", File Size (bytes) = \"" + item.getSize()
                            + "\"");
                        fileItem = item;
                    }
                }

                if (videoNameItem == null) {
                    log.severe("Video name field not found in video upload form");
                    return;
                }
                if (fileItem == null) {
                    log.severe("File field not found in video upload form");
                    return;
                }

                /*
                 * Write file to the ultimate location.
                 */
                FileItem i = fileItem;
                file = new File(destinationDir, fileItem.getName()); // Allocate the space
                fileItem.write(file); // Save the file to the allocated space
                int projectId = getProjectIdFromSessionAttributes(session);
                String videoName = videoNameItem.getString();
                if (Security.isSafeVideoName(videoName)
                        && Security.isSafeVideo(i)
                        && Security.isUniqueVideoName(videoName, projectId)
                        && Security.videoFits(DatabaseApi
                                .getNumberOfVideos(projectId))) {
                    String[] videoUrlAndIcon = S3Api.uploadFile(file);
                    String videoUrl = videoUrlAndIcon[0];
                    String videoIcon = videoUrlAndIcon[1];
                    if (videoUrl != null) {
                        DatabaseApi.addVideo(new Video(videoName, videoUrl,
                                videoIcon, projectId, -1, -1, false)); // TODO Get icon from Zen
coder.
                        // File downloadedFile = S3Api.downloadFile(videoUrl); // TODO Add to /t
emp/ folder so it can be played in the player.
                        out.println("File uploaded. Please close this window and refresh the edi
tor page.");
                        out.println();
                        response.sendRedirect("editor.jsp"); // Keep the user on the same page.
                    } else {
                        out.println("Upload Failed. Error uploading video to the cloud.");
                        log.warning("Upload Failed. Error uploading video to the cloud.");
                        // response.sendError(HttpServletResponse.SC_BAD_REQUEST);
                        return;
                    }
                } else if (!Security.isSafeVideoName(videoName)) {
                    out.println("Upload Failed. Video name may consist of a-z, 0-9; and must beg
in with a letter.");
                    log.warning("Upload Failed. Video name may consist of a-z, 0-9; and must beg
in with a letter.");
                    // response.sendError(HttpServletResponse.SC_UNSUPPORTED_MEDIA_TYPE);
                    return;
                } else if (!Security.isUniqueVideoName(videoName, projectId)) {
                    out.println("Upload Failed. Video name already exists.");
                    log.warning("Upload Failed. Video name already exists.");
                    return;
                } else {
                    out.println("Upload Failed. Video type is invalid or maximum number of video
s reached.");
                    log.warning("Upload Failed. Video type is invalid or maximum number of video
s reached.");
                    // response.sendError(HttpServletResponse.SC_BAD_REQUEST, "Bad Name");
                    return;
                }

            } catch (FileUploadException e) {
                log.severe("Error encountered while parsing the request in the upload handler");
                out.println("Upload failed.");
                e.printStackTrace();
            } catch (InvalidFileSizeException e) {
                log.severe("Invalid file size");
                out.println("Upload failed (invalid file size)");
                e.printStackTrace();
            } catch (Exception e) {
                log.severe("Unknown error encountered while uploading file");
                out.println("Upload failed (unknown reason).");
                e.printStackTrace();
            } finally {
                if (out != null) {
                    out.close();
                }
                if (file != null) {
                    file.delete();
                }
            }
        }

        // Gson help: http://code.google.com/p/google-gson/
        // http://stackoverflow.com/questions/338586/a-better-java-json-library
        // http://stackoverflow.com/questions/1688099/converting-json-to-java/1688182#1688182
        private void handleJsonPostRequest(HttpServletRequest request,
                HttpServletResponse response, HttpSession session)
                throws IOException, NoSuchAlgorithmException,
                InvalidFileSizeException, ServiceException, InterruptedException {
            String line;
            String json = "";
            while ((line = request.getReader().readLine()) != null) {
                json += line;
            }
            request.getReader().close();

            GeneralRequest generalRequest = new Gson().fromJson(json,
                    GeneralRequest.class);

            if (generalRequest.command.equals("deleteAccount")) {
                handleDeleteAccount(request, response, session, json);
            } else if (generalRequest.command.equals("getAccountInformation")) {
                handleGetAccountInformation(request, response, session, json);
            } else if (generalRequest.command.equals("getProjects")) {
                handleGetProjects(request, response, session, json);
            } else if (generalRequest.command.equals("createProject")) {
                handleCreateProject(request, response, session, json);
            } else if (generalRequest.command.equals("deleteProject")) {
                handleDeleteProject(request, response, session, json);
            } else if (generalRequest.command.equals("renameProject")) {
                handleRenameProject(request, response, session, json);
            } else if (generalRequest.command.equals("updateProjectsPosition")) {
                handleUpdateProjectsPosition(request, response, session, json);
            } else if (generalRequest.command.equals("getProjectInformation")) {
                handleGetProjectInformation(request, response, session, json);
            } else if (generalRequest.command.equals("getVideos")) {
                handleGetVideos(request, response, session, json);
            } else if (generalRequest.command.equals("createVideo")) {
```

```
            handleCreateVideo(request, response, session, json);
        } else if (generalRequest.command.equals("deleteVideo")) {
            handleDeleteVideo(request, response, session, json);
        } else if (generalRequest.command.equals("renameVideo")) {
            handleRenameVideo(request, response, session, json);
        } else if (generalRequest.command.equals("addToTimeline")) {
            handleAddToTimeline(request, response, session, json);
        } else if (generalRequest.command.equals("removeFromTimeline")) {
            handleRemoveFromTimeline(request, response, session, json);
        } else if (generalRequest.command.equals("addToSelected")) {
            handleAddToSelected(request, response, session, json);
        } else if (generalRequest.command.equals("removeFromSelected")) {
            handleRemoveFromSelected(request, response, session, json);
        } else if (generalRequest.command.equals("splitVideo")) {
            handleSplitVideo(request, response, session, json);
        } else if (generalRequest.command.equals("combineVideos")) {
            handleCombineVideos(request, response, session, json);
        } else if (generalRequest.command.equals("updateVideosPosition")) {
            handleUpdateVideosPosition(request, response, session, json);
        } else if (generalRequest.command.equals("updateTimelinePosition")) {
            handleUpdateTimelinePosition(request, response, session, json);
        } else if (generalRequest.command.equals("getVideoInformation")) {
            handleGetVideoInformation(request, response, session, json);
        } else if (generalRequest.command.equals("isFirstSignIn")) {
            handleIsFirstSignIn(request, response, session, json);
        } else {
            log.warning("JSON request not recognized.");
            log.warning("JSON request not recognized.");
            response.sendError(HttpServletResponse.SC_BAD_REQUEST);
            return;
        }
    }

    private void handleDeleteAccount(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        DeleteAccountRequest deleteAccountRequest = new Gson().fromJson(json,
                DeleteAccountRequest.class);

        response.setContentType("text/plain");
        // response.setStatus(HttpServletResponse.SC_OK); // Unnecessary

        PrintWriter out = response.getWriter();

        String sessionUsername = (String) session
                .getAttribute(Attribute.USERNAME.toString());
        int accountId = DatabaseApi.getAccountId(sessionUsername);
        DatabaseApi.deleteAccount(accountId);

        out.println(deleteAccountRequest.arguments.account
                + " deleted successfully.");
        out.close();
    }

    private void handleGetAccountInformation(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        response.setContentType("application/json; charset=utf-8");
        PrintWriter out = response.getWriter();

        int accountId = getAccountIdFromSessionAttributes(session);
        String name = DatabaseApi.getAccountName(accountId);
        String email = DatabaseApi.getAccountEmail(accountId);
        String password = ""; // The encryption function is one-way (and it's a security iss
ue to redisplay this).
        String securityQuestion = DatabaseApi
                .getAccountSecurityQuestion(accountId);
        String securityAnswer = DatabaseApi.getAccountSecurityAnswer(accountId);

        out.println(new Gson().toJson(new GetAccountInformationResponse(name,
                email, password, securityQuestion, securityAnswer)));
        out.close();
    }

    private void handleGetProjects(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        log.severe("handleGetProjects has not yet been implemented.");
    }

    private void handleCreateProject(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        log.severe("handleCreateProject has not yet been implemented.");
    }

    private void handleDeleteProject(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        DeleteProjectRequest deleteProjectRequest = new Gson().fromJson(json,
                DeleteProjectRequest.class);

        response.setContentType("text/plain");
        // response.setStatus(HttpServletResponse.SC_OK); // Unnecessary

        PrintWriter out = response.getWriter();

        String sessionUsername = (String) session
                .getAttribute(Attribute.USERNAME.toString());
        int accountId = DatabaseApi.getAccountId(sessionUsername);
        String projectToDelete = deleteProjectRequest.arguments.project;
        int projectId = DatabaseApi.getProjectId(projectToDelete, accountId);
        DatabaseApi.deleteProject(projectId);

        out.println(deleteProjectRequest.arguments.project
                + " deleted successfully.");
        out.close();
    }

    private void handleRenameProject(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        RenameProjectRequest renameProjectRequest = new Gson().fromJson(json,
                RenameProjectRequest.class);

        String newProjectName = renameProjectRequest.arguments.newProjectName;
        String oldProjectName = renameProjectRequest.arguments.oldProjectName;
        int projectId = getProjectIdFromSessionAttributes(session,
                oldProjectName);
        int accountId = getAccountIdFromSessionAttributes(session);
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();

        if (Security.isSafeProjectName(newProjectName)
                && Security.isUniqueProjectName(newProjectName, accountId)) {
            DatabaseApi.renameProject(projectId, newProjectName);
```

```
            out.println(newProjectName);
        } else {
            out.println(oldProjectName);
        }
        out.close();
    }

    private void handleUpdateProjectsPosition(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        UpdateProjectsPositionRequest updateProjectsPositionRequest = new Gson()
                .fromJson(json, UpdateProjectsPositionRequest.class);

        int accountId = getAccountIdFromSessionAttributes(session);
        DatabaseApi.setAllProjectPoolPositionsToNull(accountId);

        int numberOfProjects = updateProjectsPositionRequest.arguments.projects.length;
        for (int projectIndex = 0; projectIndex < numberOfProjects; ++projectIndex) {
            String projectName = updateProjectsPositionRequest.arguments.projects[projectInd
ex].project;
            int projectId = getProjectIdFromSessionAttributes(session,
                    projectName);
            int position = updateProjectsPositionRequest.arguments.projects[projectIndex].po
sition;
            DatabaseApi.setProjectPoolPosition(projectId, position);
        }
    }

    private void handleGetProjectInformation(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        int accountId = getAccountIdFromSessionAttributes(session);
        String projectString = S3Api.getProjectInformation(accountId);
        response.setContentType("application/json; charset=utf-8");
        PrintWriter out = response.getWriter();
        out.println(projectString);
        out.close();
    }

    private void handleGetVideos(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        log.severe("handleGetVideos has not yet been implemented.");
    }

    private void handleCreateVideo(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        log.severe("handleCreateVideo has not yet been implemented.");
    }

    private void handleDeleteVideo(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        DeleteVideoRequest deleteVideoRequest = new Gson().fromJson(json,
                DeleteVideoRequest.class);

        response.setContentType("text/plain");

        PrintWriter out = response.getWriter();

        int projectId = getProjectIdFromSessionAttributes(session);
        int videoId = DatabaseApi.getVideoId(
                deleteVideoRequest.arguments.video, projectId);

        S3Api.deleteFileInS3(S3Api.getNameFromUrl(DatabaseApi
                .getVideoUrl(videoId)));    // Delete video
        S3Api.deleteFileInS3(S3Api.getNameFromUrl(DatabaseApi
                .getVideoIcon(videoId)));   // Delete icon

        DatabaseApi.deleteVideo(videoId);

        out.println(deleteVideoRequest.arguments.video
                + " deleted successfully.");
        out.close();
    }

    private void handleRenameVideo(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        RenameVideoRequest renameVideoRequest = new Gson().fromJson(json,
                RenameVideoRequest.class);

        String newVideoName = renameVideoRequest.arguments.newVideoName;
        String oldVideoName = renameVideoRequest.arguments.oldVideoName;
        int videoId = getVideoIdFromSessionAttributes(session, oldVideoName);
        int projectId = getProjectIdFromSessionAttributes(session);
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();

        if (Security.isSafeVideoName(newVideoName)
                && Security.isUniqueVideoName(newVideoName, projectId)) {
            DatabaseApi.renameVideo(videoId, newVideoName);
            out.println(newVideoName);
        } else {
            out.println(oldVideoName);
        }
        out.close();
    }

    private void handleAddToTimeline(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        log.severe("handleAddToTimeline has not yet been implemented.");
    }

    private void handleRemoveFromTimeline(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        log.severe("handleRemoveFromTimeline has not yet been implemented.");
    }

    private void handleAddToSelected(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        AddToSelectedRequest addToSelectedRequest = new Gson().fromJson(json,
                AddToSelectedRequest.class);

        int videoId = getVideoIdFromSessionAttributes(session,
                addToSelectedRequest.arguments.video);

        if (!DatabaseApi.setVideoAsSelected(videoId)) {
            log.severe("Error marking video " + videoId + " as selected");
        }
    }
```

```
private void handleRemoveFromSelected(HttpServletRequest request,
        HttpServletResponse response, HttpSession session, String json)
        throws IOException {
    RemoveFromSelectedRequest removeFromSelectedRequest = new Gson()
            .fromJson(json, RemoveFromSelectedRequest.class);

    int videoId = getVideoIdFromSessionAttributes(session,
            removeFromSelectedRequest.arguments.video);

    if (!DatabaseApi.setVideoAsUnselected(videoId)) {
        log.severe("Error marking video " + videoId + " as unselected");
    }
}

private void handleSplitVideo(HttpServletRequest request,
        HttpServletResponse response, HttpSession session, String json)
        throws IOException {
    SplitVideoRequest splitVideoRequest = new Gson().fromJson(json,
            SplitVideoRequest.class);

    response.setContentType("text/plain");

    PrintWriter out = response.getWriter();

    if (!Security.isSafeVideoName(splitVideoRequest.arguments.video)) {
        out.println("Name of video to split is invalid.");
        log.warning("Name of video to split is invalid.");
        response.sendError(HttpServletResponse.SC_BAD_REQUEST);
        return;
    }

    if (!Security
            .isSafeSplitTimeInSeconds(splitVideoRequest.arguments.splitTimeInSeconds)) {
        out.println("Split time (in seconds) is invalid.");
        log.warning("Split time (in seconds) is invalid.");
        response.sendError(HttpServletResponse.SC_BAD_REQUEST);
        return;
    }

    int projectId = getProjectIdFromSessionAttributes(session);
    int videoId = DatabaseApi.getVideoId(splitVideoRequest.arguments.video,
            projectId);
    Video[] videoFragments = ZencoderApi.split(videoId,
            splitVideoRequest.arguments.splitTimeInSeconds);

    for (Video videoFragment : videoFragments) { // foreach-loop
        DatabaseApi.addVideo(new Video(videoFragment.getName(),
                videoFragment.getUrl(), videoFragment.getIcon(), projectId,
                -1, -1, false)); // projectId not computed by Zencoder
    }

    out.println(splitVideoRequest.arguments.video + " split at "
            + splitVideoRequest.arguments.splitTimeInSeconds
            + " seconds successfully.");
    out.close();
}

private void handleCombineVideos(HttpServletRequest request,
        HttpServletResponse response, HttpSession session, String json)

throws IOException, NoSuchAlgorithmException, InvalidFileSizeException,
        ServiceException, InterruptedException {
    //CombineVideosRequest combineVideosRequest = new Gson().fromJson(json,
        //  CombineVideosRequest.class);
    log.info("COMBINING VIDEOS");
    // response.setContentType("text/plain");

    // PrintWriter out = response.getWriter();
    int projectId = getProjectIdFromSessionAttributes(session);
    String[] videos = DatabaseApi.getVideosOnTimeline(projectId);
    log.info("videos length: "+videos.length);
    if(videos.length > 0){
        String[] videoURLs = new String[videos.length];
        for (int i = 0; i < videos.length; i++) {
            videoURLs[i] = DatabaseApi.getVideoUrl(DatabaseApi.getVideoId(
                    videos[i], projectId));
        }
        String combinedURL = "";
        if(videoURLs.length == 1){
            combinedURL = S3Api.downloadVideosToTemp(videoURLs[0]);
        }
        else if (videoURLs.length > 1){
            combinedURL = combineVideos(videoURLs, videos);
        }

        // My view resource servlet:
        // Use a ServletOutputStream because we may pass binary information
        log.info("Combined. Now Dowloading");
        final ServletOutputStream out = response.getOutputStream();
        response.setContentType("application/octet-stream");
        response.setHeader("Content-Disposition",
        "attachment;filename=combinedVideo.ogv");
        File file = new File(combinedURL);
        BufferedInputStream is = new BufferedInputStream(new FileInputStream(
                file));
        byte[] buf = new byte[4 * 1024]; // 4K buffer
        int bytesRead;
        log.info("downloading");
        while ((bytesRead = is.read(buf)) != -1) {
            log.info("in while");
            out.write(buf, 0, bytesRead);
            log.info("...Dowloading in while...");
        }

        is.close();
        out.close();
        log.info("end of handleCombinedVideos");
    }
    else{
        response.sendRedirect("editor.jsp");
    }
}

private String combineVideos(String[] videoURLs, String[] videos)
        throws IOException, NoSuchAlgorithmException,
        InvalidFileSizeException, ServiceException, InterruptedException {
    String[] result = new String[2];
    result[0] = "combined";
    Runtime r = Runtime.getRuntime();
    boolean isWindows = isWindows();
    boolean isLinux = isLinux();

    File combined = new File(videoURLs[0]);
    S3Api.downloadVideosToTemp(videoURLs[0]);
    Process p;
```

```
        String videoName = "";
        for (int i = 0; i < videos.length - 1; i++) { // Use i+1 everywhere
            videoName = S3Api.downloadVideosToTemp(videoURLs[i + 1]);
            p = r.exec("ffmpeg -i " + combined.getName() + " -sameq temp.mpg",
                    null, tempDir);

            /*
             * BufferedReader in = new BufferedReader(new InputStreamReader(p.getErrorStream
())) ;
             * String currentLine = null;
             * while (( currentLine = in.readLine()) != null )
             * System.out.println ( currentLine ) ;
             */

            // p.waitFor();
            // log.info("Process 1...Done");
            p = r.exec("ffmpeg -i " + videoName + " -sameq temp2.mpg", null,
                    tempDir);
            // p.waitFor();
            // log.info("Process 2...Done");
            if (isWindows) {

                log.info("Windows");
                p = r.exec(
                        "cmd /c copy /b temp.mpg+temp2.mpg intermediateTemp.mpg",
                        null, tempDir);
                InputStream is2 = p.getInputStream();
                log.info("InputStream2");
                InputStreamReader isr2 = new InputStreamReader(is2);
                log.info("InputStreamReader2");
                BufferedReader br2 = new BufferedReader(isr2);
                String line;
                for (int j = 0; j < 10; j++) {
                    line = br2.readLine();
                    if (line != null) {
                        log.info("Line " + j + 1 + ": " + line);
                    }
                }
                // p.waitFor();
                log.info("after Windows Process finishes");
                // log.info("Process 3...Done");
                // r.exec("cmd /c del temp\\"+videos[i+1]+".mpg");
            } else if (isLinux) {

                log.info("Linux");
                String[] arr = { "/bin/sh", "-c",
                        "cat temp.mpg temp2.mpg > intermediateTemp.mpg" };
                p = r.exec(arr, null, tempDir);
                // p.waitFor();
                // log.info("Process 3...Done");
                // r.exec("rm temp\\"+videos[i+1]+".mpg");
            } else {
                return null;
            }
            log.info("after IFS");

            p = r.exec("ffmpeg -i intermediateTemp.mpg -sameq combined.ogv",
                    null, tempDir);
            InputStream is2 = p.getErrorStream();
            log.info("InputStream");
            InputStreamReader isr2 = new InputStreamReader(is2);
            log.info("InputStreamReader");
            BufferedReader br2 = new BufferedReader(isr2);
            String line;
            for (int j = 0; j < 25; j++) {
                line = br2.readLine();
                if (line != null) {
                    log.info("FFMPEG Line " + j + 1 + ": " + line);
                }
            }
            log.info("after last ffmpeg process");

            BufferedReader in = new BufferedReader(new InputStreamReader(
                    p.getErrorStream()));
            String currentLine = null;
            while ((currentLine = in.readLine()) != null)
                System.out.println(currentLine);
            p.waitFor();
            // log.info("Process 4...Done");
            combined = new File(tempDir.getAbsolutePath() + File.separator
                    + "combined.ogv");
            // process.waitFor();

            videos[0] = "combined";
        }

        // Removing all temp files except for the one combined video
        // result[1] = videoURLs[0];
        // if(inFor){
        // return S3Api.uploadFile(combined);
        // }
        // else{
        return combined.getAbsolutePath();
        // }
        // return S3Api.uploadFile(new File(videoName));
        // return null;
        // return videoURLs[0];

    }

    // http://www.mkyong.com/java/how-to-detect-os-in-java-systemgetpropertyosname/
    private Boolean isWindows() {
        String os = System.getProperty("os.name").toLowerCase();
        // windows
        return (os.indexOf("win") >= 0);
    }

    private Boolean isLinux() {
        String os = System.getProperty("os.name").toLowerCase();
        // linux or unix
        return (os.indexOf("nix") >= 0 || os.indexOf("nux") >= 0);
    }

    private void handleUpdateVideosPosition(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        UpdateVideosPositionRequest updateVideosPositionRequest = new Gson()
                .fromJson(json, UpdateVideosPositionRequest.class);

        int projectId = getProjectIdFromSessionAttributes(session);
        DatabaseApi.setAllVideoPoolPositionsToNull(projectId);

        int numberOfVideos = updateVideosPositionRequest.arguments.videos.length;
        for (int videoIndex = 0; videoIndex < numberOfVideos; ++videoIndex) {
            String videoName = updateVideosPositionRequest.arguments.videos[videoIndex].vide
o;
```

```
            int videoId = getVideoIdFromSessionAttributes(session, videoName);
            int position = updateVideosPositionRequest.arguments.videos[videoIndex].position
;
            DatabaseApi.setVideoPoolPosition(videoId, position);
        }
    }

    private void handleUpdateTimelinePosition(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        UpdateTimelinePositionRequest updateTimelinePositionRequest = new Gson()
                .fromJson(json, UpdateTimelinePositionRequest.class);

        int projectId = getProjectIdFromSessionAttributes(session);
        DatabaseApi.setAllVideoTimelinePositionsToNull(projectId);

        int numberOfVideos = updateTimelinePositionRequest.arguments.videos.length;
        for (int videoIndex = 0; videoIndex < numberOfVideos; ++videoIndex) {
            String videoName = updateTimelinePositionRequest.arguments.videos[videoIndex].vi
deo;
            int videoId = getVideoIdFromSessionAttributes(session, videoName);
            int position = updateTimelinePositionRequest.arguments.videos[videoIndex].positi
on;
            DatabaseApi.setTimelinePosition(videoId, position);
        }
    }

    private void handleGetVideoInformation(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        int projectId = getProjectIdFromSessionAttributes(session);
        String videoString = S3Api.getVideoInformation(projectId);
        response.setContentType("application/json; charset=utf-8");
        PrintWriter out = response.getWriter();
        out.println(videoString);
        out.close();
    }

    private void handleIsFirstSignIn(HttpServletRequest request,
            HttpServletResponse response, HttpSession session, String json)
            throws IOException {
        response.setContentType("application/json; charset=utf-8");
        PrintWriter out = response.getWriter();
        out.println(new Gson().toJson((Boolean) session
                .getAttribute(Attribute.IS_FIRST_SIGN_IN.toString())));
        out.close();
    }
}
```

```
package com.readytalk.olive.util;
/**
 * Doc for enum
 * @author Team Olive
 *
 */
public enum Attribute {
    /**
     * Uses for whether
     */
    IS_AUTHORIZED,
    /**
     *
     */
    USERNAME,
    /**
     *
     */
    PASSWORD,
    /**
     *
     */
    EDIT_SUCCESSFULLY,
    /**
     *
     */
    ADD_SUCCESSFULLY,
    /**
     *
     */
    PROJECT_NAME,
    /**
     *
     */
    PASSWORDS_MATCH,
    /**
     *
     */
    IS_SAFE,
    /**
     *
     */
    EMAIL,
    /**
     *
     */
    NAME,
    /**
     *
     */
    SECURITY_QUESTION,
    /**
     *
     */
    SECURITY_ANSWER,
    /**
     *
     */
    IS_CORRECT,
    /**
     *
     */
    IS_FIRST_SIGN_IN
}
```

```
package com.readytalk.olive.util;

public class InvalidFileSizeException extends Exception {

    // Created using Eclipse’s "Add generated serial version ID" refactoring.
    private static final long serialVersionUID = 16991312545332943 30L;

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

    public InvalidFileSizeException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }

    public InvalidFileSizeException(Throwable cause) {
        super(cause);
        // TODO Auto-generated constructor stub
    }

    public InvalidFileSizeException(String message, Throwable cause) {
        super(message, cause);
        // TODO Auto-generated constructor stub
    }

}
```

```
/*
    ----------------------------------------
    Edit account page
    ----------------------------------------
*/
#edit-account-container {
    padding: 5%;
}

#edit-account-container h2 {
    margin: .75em 0;
}
```

```
/*
    ----------------------------------------
    Editor page
    ----------------------------------------
*/
#videos-container {
    position: absolute;
    top: 20px;
    right: 60%;
    bottom: 125px;
    left: 0px;
    /* 100% - (video width) - (2 x (sum of padding, border, margin)) */
}

#videos-header {
    position: absolute;
    top: 0%;
    right: 0%;
    bottom: 85%;
    left: 0%;
}

#videos-title {
    float: left;
    margin: 3%;
    font-size: 240%;    /* 24px */
    color: #92bd6a; /* The Olive color */
}

#videos-controls {
    float: right;
    margin: 1%;
}

#videos {
    position: absolute;
    top: 15%;
    right: 0%;
    bottom: 0%;
    left: 0%;
    text-align: left;
    border: 1px solid gray;
    margin: 10px 0px 10px 10px;
    overflow: auto;
}

.video-container {
    float: left;
    height: 100px;
    width: 120px;
    margin: 5px;
    padding: 5px;
    border: 1px solid green;
    background-color: #FEFEFE;
    text-align: center;
    overflow: hidden;
}

.video-image {
    border: 1px solid gray;
    height: 40px;
}

.video-name {
    line-height: 1.2em; /* Larger than font-size */
    font-size: 1em; /* Smaller than line-height */
    max-height: 3.6em;  /* 3 x line-height */
    text-align: center;
    word-wrap: break-word;
    overflow: hidden; /* Temporary solution to text-overflow not working */
}

.video-name:hover {
    background-color: #ffffd3;
}

.video-name > input {
    width: 90%; /* Same as .video-container */
}

.video-name > button {
}

#player-container {
    position: absolute;
    top: 0px;
    right: 0px;
    bottom: 125px;
    left: 40%;
    /* A bit smaller than #videos-container's right edge (compensate for border, etc. */
    border: 1px solid gray;
    background-color: #101010;
    text-align: center;
    margin: 10px;
}

#player {
    /* Width and height should be 100% to ensure the video can never extend
     * beyond its container, but doing so allows the controls (which don't have
     * an aspect ratio to maintain) to extend beyond the video playing area.
     */
    width: 100%;
    height: 100%;
}

#timeline {
    position: absolute;
    right: 150px;
    bottom: 0px;
    left: 0px;
    height: 125px; /* Larger than height of .video-container */
    border-top: 1px solid gray;
    border-right: 1px solid gray;
    border-left: 1px solid gray;
    margin-left: 10px;
    overflow-x: auto;
    /*text-align: center;*/
}

/*
#timeline-background-text {
    font-size: 100px;
    color: #eeeeee;
    position: relative;
    top: 40%;
}*/
```

```
#tick-anchor {
    position: absolute;
    top: 50%;
    height: 0px;
    width: 100%;
    /*border-top: 1px solid green;*/
}

#export {
    position: absolute;
    right: 0px;
    bottom: 0px;
}
```

```
/*
    ----------------------------------------
    Login page
    ----------------------------------------
*/
/*#splash-container {
    position: absolute;
    top: 0%;
    right: 50%;
    bottom: 0%;
    left: 0%;
    padding: 10% 10% 10% 10%;
}

#splash-image {
    float: right;
    width: 379px;
    height: 145px;
}*/

#login-form-container {
    position: absolute;
    top: 0%;
    right: 0%;
    bottom: 0%;
    left: 55%;
    padding: 10% 10% 10% 10%;
}

#login-form {

}
#screencast-container{
    position: absolute;
    top: 0%;
    right: 50%;
    bottom: 0%;
    left: 0%;
    padding: 2% 10% 10% 5%;

}

#main{
    min-width: 1200px;
}
```

```
@charset "UTF-8";

/* For HTML4 defaults, see: http://www.w3.org/TR/CSS2/sample.html */ /*
    ---------------------------------------
    HTML elements
    ---------------------------------------
*/
body {
    /* The default font-size is 16px, but this makes the math easier. For
     * example, to set the size of the 'p' element to 16px, set it to 160%.
     * To set the size of the 'h1' element to 32px, set it to 320%. Why not just
     * set the 'p' element to 16px and the 'h1' element to 32px? Because users
     * who want to resize the text on the page won't be able to because the size
     * is fixed. It's better to express font-size in terms of percentages, and
     * this is a neat way to do that.
     */
    font-size: 100%;
    line-height: 20px;
    font-family: Helvetica, Arial, sans-serif;
    text-align: left;
    /*background-color: #edf4e6;*/
    /* A lightened version of the Olive color */
}

h1,h2,h3,h4,h5,h6 {
    color: #92bd6a; /* The Olive color */
    font-family: Calibri, Helvetica, Arial, sans-serif;
}

h1 {
    font-size: 320%; /* 32px */ /*margin: .67em 0;*/
    padding: 20px;
}

h2 {
    font-size: 240%; /* 24px */ /*margin: .75em 0*/
}

h3 {
    font-size: 187.2%; /* 18.72px */ /*margin: .83em 0;*/
}

h4 {
    font-size: 160%; /* 16px */ /*margin: 1.12em 0;*/
}

h5 {
    font-size: 132.8%; /* 13.28px */ /*margin: 1.5em 0;*/
}

h6 {
    font-size: 120%; /* 12px */ /*margin: 1.67em 0;*/
}

p {
    font-size: 120%; /* 12px */
    margin: 1.12em 0;
    line-height: 1.2em;
}

a:link {
    color: green;
}

a:visited {
    color: green;
}

a:hover {
    color: #92bd6a;
}

table {
    border-spacing: 40px 10px;
    margin-left: 100px;
    padding: 10px;
}

tr,td {
    padding: 0px 10px 0px 10px;
}

button { /*background-color: #edf4e6;*/
    /* A lightened version of the Olive color */

}

ul {
    list-style-type: none;
}

strong {
    font-weight: bolder;
}

small {
    font-size: .83em;
}

/*
    ---------------------------------------
    General styling
    ---------------------------------------
*/
.clear {
    clear: both;
}

.center-text {
    text-align: center
}

.center-block {
    margin-left: auto;
    margin-right: auto;
}

.warning {
    color: red;
    text-decoration: underline;
    cursor: pointer;
}
.link {
    color: blue;
    text-decoration:underline;
    cursor: pointer;
```

```
}

.hidden {
    display: none;
}

/*
    ---------------------------------------
    Common elements to every page
    ---------------------------------------
*/
#header {
    position: absolute;
    top: 0px;
    right: 0px;
    left: 0px;
    height: 50px;
    border-bottom: 1px solid #888888;
    min-width: 1000px;
}

#header-left {
    text-align: left;
    margin-left: 1%;
}

#olive-icon {
    float: left;
    height: 45px;
    width: 45px;
    margin-top: 5px;
}

#olive-title {
    float: left;
}

#header-right {
    float: right;
    text-align: right;
    margin-right: 1%;
}

#main {
    position: absolute;
    top: 50px;
    right: 0px;
    bottom: 30px;
    left: 0px;
    min-width: 1000px;
    min-height: 450px;
    overflow: auto;
}

#footer {
    position: absolute;
    right: 0px;
    bottom: 0px;
    left: 0px;
    height: 30px;
    border-top: 1px solid #888888;
    min-width: 1000px;
}

#footer-left {
    float: left;
    text-align: left;
    margin-left: 1%;
}

#footer-right {
    float: right;
    text-align: right;
    margin-right: 1%;
}

/*
    ---------------------------------------
    jQuery modal dialog
    ---------------------------------------
*/
label,input {
    display: block;
}

input.text {
    margin-bottom: 12px;
    width: 95%;
    padding: .4em;
}

fieldset {
    padding: 0;
    border: 0;
    margin-top: 25px;
}

div#users-contain {
    width: 350px;
    margin: 20px 0;
}

div#users-contain table {
    margin: 1em 0;
    border-collapse: collapse;
    width: 100%;
}

div#users-contain table td,div#users-contain table th {
    border: 1px solid #eee;
    padding: .6em 10px;
    text-align: left;
}

.ui-dialog .ui-state-error {
    padding: .3em;
}

.validateTips {
    border: 1px solid transparent;
    padding: 0.3em;
}
```

```
/*
    ----------------------------------------
    Projects page
    ----------------------------------------
*/
#projects-container {
    position: absolute;
    top: 0%;
    right: 25%;
    bottom: 0%;
    left: 25%;
}

#projects-title {
    position: absolute;
    top: 0%;
    right: 0%;
    bottom: 85%;
    left: 0%;
    overflow: hidden;
}

#projects-controls {
    position: absolute;
    top: 15%;
    right: 0%;
    bottom: 75%;
    left: 0%;
    overflow: hidden;
}

#projects {
    position: absolute;
    top: 25%;
    right: 0%;
    bottom: 0%;
    left: 0%;
    text-align: left;
    border-top: 1px solid gray;
    overflow: auto;
}

.project-container {
    float: left;
    height: 125px;
    width: 125px;
    margin: 5px;
    padding: 5px;
    border: 1px solid green;
    background-color: #FEFEFE;
    text-align: center;
}

.project-image {
    border: 1px solid gray;
    height: 40px;
}

.project-name {
    line-height: 1.2em; /* Larger than font-size */
    font-size: 1em; /* Smaller than line-height */
    max-height: 3.6em;  /* 3 x line-height */
    text-align: center;
    word-wrap: break-word;
    overflow: hidden; /* Temporary solution to text-overflow not working */
}

.project-name:hover {
    background-color: #ffffd3;
}

.project-name > input {
    width: 90%; /* Same as .video-container */
}

.project-name > button {
}
```

```
@charset "UTF-8";

/*
  This file resets the default browser behavior so the web page is consistent
  across all browsers.

  Modified from: http://html5boilerplate.com/
  To revert, see: http://www.w3.org/TR/CSS2/sample.html
*/ /*
  HTML5 â\234° Boilerplate

  style.css contains a reset, font normalization and some base styles.

  credit is left where credit is due.
  much inspiration was taken from these projects:
    yui.yahooapis.com/2.8.1/build/base/base.css
    camendesign.com/design/
    praegnanz.de/weblog/htmlcssjs-kickstart
*/ /*
  html5doctor.com Reset Stylesheet (Eric Meyer's Reset Reloaded + HTML5 baseline)
  v1.4 9-07-27 | Authors: Eric Meyer & Richard Clark
  html5doctor.com/html-5-reset-stylesheet/
*/
html,body,div,span,object,iframe,h1,h2,h3,h4,h5,h6,p,blockquote,pre,abbr,address,cite,code,d
el,dfn,em,img,ins,kbd,q,samp,small,strong,sub,sup,var,b,i,dl,dt,dd,ol,ul,li,fieldset,form,la
bel,legend,table,caption,tbody,tfoot,thead,tr,th,td,article,aside,canvas,details,figcaption,
figure,footer,header,hgroup,menu,nav,section,summary,time,mark,audio,video
    {
    margin: 0;
    padding: 0;
    border: 0;
    outline: 0;
    /*font-size: ; */
    vertical-align: baseline;
    background: transparent;
}

article,aside,details,figcaption,figure,footer,header,hgroup,menu,nav,section
    {
    display: block;
}

nav ul {
    list-style: none;
}

blockquote,q {
    quotes: none;
}

blockquote:before,blockquote:after,q:before,q:after {
    content: '';
    content: none;
}

a {
    margin: 0;
    padding: 0;
    font-size: ;
    vertical-align: baseline;
    background: transparent;
}

ins {
    background-color: #ff9;
    color: # text-decoration :           none;
}

mark {
    background-color: #ff9;
    color: # font-style :           italic;
    font-weight: bold;
}

del {
    text-decoration: line-through;
}

abbr[title],dfn[title] {
    border-bottom: 1px dotted;
    cursor: help;
}

/* tables still need cellspacing="0" in the markup */
table {
    border-collapse: collapse;
    border-spacing: 0;
}

hr {
    display: block;
    height: 1px;
    border: 0;
    border-top: 1px solid #ccc;
    margin: 1em 0;
    padding: 0;
}

input,select {
    vertical-align: middle;
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Context antiJARLocking="true" path="/olive">
    <Resource name="jdbc/OliveData" auth="Container" type="javax.sql.DataSource"
        username="olive" password="Xsbpdrhr" driverClassName="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost:3306/" maxActive="15" maxIdle="3" />
</Context>
```

```
This file exists purely to keep its parent directly under source control.
```

```
/*
 * This is Olive's JavaScript file for account.js only.
 */

var deleteAccountDialogContext; // TODO Remove this global variable.

// Called once the DOM is ready but before the images, etc. load.
// Failsafe jQuery code modified from: http://api.jquery.com/jQuery/#jQuery3
jQuery(function($) {
    attachDeleteAccountHandlers();
    injectAccountData();
});

function attachDeleteAccountHandlers() {
    $('#delete-account').click(function () {
        $('#confirm-delete-account-dialog').dialog('open');
        deleteAccountDialogContext = this;  // This is a global variable.
    });

    $('#confirm-delete-account-dialog').dialog({
        autoOpen: false,
        resizable: false,
        height: 275,
        modal: true,
        buttons: {
            'Delete': function () {
                deleteAccount.call(deleteAccountDialogContext); // We don't want the context
 to be the dialog element, but rather the element that triggered it.
                $(this).dialog('close');
            },
            Cancel: function () {
                $(this).dialog('close');
            }
        }
    });
}

// Perform a deleteAccount request
function deleteAccount() {
    var requestData = '{'
            +    '"command" : "deleteAccount",'
            +    '"arguments" : {'
            +       '"account" : "' + $(this).attr('id') + '"'
            +     '}'
            +  '}';
    makeAjaxPostRequest(requestData, logout, null); // Defined in "/olive/scripts/master.js"
.
}

function injectAccountData() {
    var requestData = '{'
        +    '"command" : "getAccountInformation"'
        +  '}';
    makeAjaxPostRequest(requestData, function (responseData) {
        $('#new-name').val(responseData.name).change();
        $('#new-email').val(responseData.email).change();
        $('#new-password').val(responseData.password).change();
        $('#confirm-new-password').val(responseData.password).change();
        $('#new-security-question').val(responseData.securityQuestion).change();
        $('#new-security-answer').val(responseData.securityAnswer).change();

    }, null);   // Defined in "/olive/scripts/master.js".
}
```

```
/*
 * This is Olive's JavaScript file for editor.jsp only.
 * Dependencies: "/olive/scripts/master.js"
 */

// Called once the DOM is ready but before the images, etc. load.
// Failsafe jQuery code modified from: http://api.jquery.com/jQuery/#jQuery3
jQuery(function($) {
    populateVideos();
});

function attachHandlers() {
    attachUploadNewVideoHandlers();
    attachDeleteVideoHandlers();
    attachSplitVideoHandlers();
    attachVideoClickHandlers();
    attachRenameVideoHandlers();
    //attachPublishButtonHandlers();
    enableDragAndDrop();
}

function createNewVideoContainer(videoName, videoNum, videoIcon) {
    var videoContainer = '<div id="video-container-'
        + videoNum
        + '" class="video-container"><img id="video-image-'
        + videoNum
        + '" class="video-image" src="'
        + videoIcon
        + '" alt="video-image-'
        + videoNum
        + '" /><div class="video-name">'
        + videoName
        + '</div><div class="video-controls"><small><a id="split-video-link-'
        + videoNum
        + '" class="split-video-link link hidden">Split</a>'
        + '<span class="video-controls-divider hidden"> | </span>'
        + '<a id="delete-video-link-'
        + videoNum
        + '" class="delete-video-link warning">Delete</a></small></div></div>';

    $('#videos').append(videoContainer);

    // Return the object that was just appended (with the jQuery wrapper stripped off).
    return $('#video-container-' + videoNum).get(0);
}

function createVideoSpinner() {
    var videoContainer = '<div class="video-container">'
        + '<img class="video-image" src="/olive/images/ajax-loader.gif" />'
        + '<div>Preparing video...</div>'
        + '</div>';
    $('#videos').append(videoContainer);
}

function populateVideos() {
    $('.video-container').hide();

    var requestData = '{'
        +    '"command" : "getVideoInformation"'
        +  '}';
    makeAjaxPostRequest(requestData, function (responseData) {
        var poolPositions = [];
        var timelinePositions = [];
        for (var i = 0; i < responseData.length; ++i) {
            var element = createNewVideoContainer(responseData[i].name, i, responseData[i].i
con);
            $(element).data('name', responseData[i].name);
            $(element).data('url', responseData[i].url);
            $(element).data('icon', responseData[i].icon);

            // Modified from: http://stackoverflow.com/questions/600700/jquery-javascript-re
ordering-rows/617349#617349
            if (responseData[i].poolPosition != -1) {
                $(element).data('poolPosition', responseData[i].poolPosition);
                poolPositions[(responseData.length - 1) - responseData[i].poolPosition] = el
ement;  // Sort in reverse order to work with prepending.
            }
            if (responseData[i].timelinePosition != -1) {
                $(element).data('timelinePosition', responseData[i].timelinePosition);
                timelinePositions[(responseData.length - 1) - responseData[i].timelinePositi
on] = element;  // Sort in reverse order to work with prepending.
            }

            $(element).data('isSelected', responseData[i].isSelected);
            makeSelectionVisible(element);
        }
        // Append in the sorted order
        for (var poolIndex = 0; poolIndex < poolPositions.length; ++poolIndex) {
            $('#videos').prepend(poolPositions[poolIndex]); // Prepend to keep unsorted elem
ents (poolPosition == -1) at the end.
        }
        for (var timelineIndex = 0; timelineIndex < timelinePositions.length; ++timelineInde
x) {
            $('#timeline').prepend(timelinePositions[timelineIndex]);   // Prepend to keep u
nsorted elements (timelinePosition == -1) at the end.
        }

        $('.video-container').show();

        enableOrDisablePublishButton();

        attachHandlers();   // This must go inside the ajax callback or it will be called to
o early.
    }, null);   // Defined in "/olive/scripts/master.js".
}

function attachUploadNewVideoHandlers() {
    var newVideoName = $('#new-video-name'),
        allFields = $([]).add(newVideoName);

    $('#new-video-dialog-form').dialog({
        autoOpen : false,
        height : 400,
        width : 400,
        modal : true,
        buttons : {
            'Upload New Video' : function () {
                var bValid = true;
                allFields.removeClass('ui-state-error');

                bValid = bValid
                        && checkLength(newVideoName,
                                'new-video-name',
                                MIN_VIDEO_NAME_LENGTH,
                                MAX_VIDEO_NAME_LENGTH);
                bValid = bValid
```

```
                        && checkRegexp(newVideoName,
                                SAFE_VIDEO_NAME_REGEX,
                                'Video name may consist of letters, numbers, underscores, an
d spaces.');
                if (bValid) {
                    createVideoSpinner();
                    $('#new-video-form').submit();
                    $(this).dialog('close');
                }
            },
            Cancel : function () {
                $(this).dialog('close');
            }
        },
        close : function () {
            allFields.val('').change().removeClass('ui-state-error');
        }
    });

    $('#upload-new-video-button').click(function() {
        $('#new-video-dialog-form').dialog('open');
    });
}

function attachDeleteVideoHandlers() {
    var videoToDelete;

    $('.delete-video-link').click(function () {
        doNotSelectThisTime();
        $('#confirm-delete-video-dialog').dialog('open');
        videoToDelete = $(this).parent().parent().parent();
    });

    $('#confirm-delete-video-dialog').dialog({
        autoOpen: false,
        resizable: false,
        height: 275,
        modal: true,
        buttons: {
            'Delete': function () {
                deleteVideo($(videoToDelete).data('name'));
                $(this).dialog('close');
            },
            Cancel: function () {
                $(this).dialog('close');
            }
        }
    });
}

//Perform a deleteVideo request
function deleteVideo(videoName) {
    var requestData = '{'
            +    '"command" : "deleteVideo",'
            +    '"arguments" : {'
            +        '"video" : "' + videoName + '"'
            +    '}'
            +  '}';
    makeAjaxPostRequest(requestData, refresh, null);    // Defined in "/olive/scripts/master
.js".
}

function attachSplitVideoHandlers() {
    $('.split-video-link').click(function () {
        var video = $('video').get(0);
        if (video.currentTime === 0 || video.ended) {
            $('#invalid-split-dialog').dialog('open');
        } else {
            var videoToSplit = $(this).parent().parent().parent();
            var maximumZencoderDecimalPlaces = 2;
            createVideoSpinner();   // First half
            createVideoSpinner();   // Second half
            splitVideo($(videoToSplit).data('name'),
                    video.currentTime.toFixed(maximumZencoderDecimalPlaces));
        }
        doNotSelectThisTime();
    });

    $('#invalid-split-dialog').dialog( {
        autoOpen : false,
        buttons: {
            OK: function() {
                $(this).dialog('close');
            }
        }
    });
}

//Perform a splitVideo request
function splitVideo(videoName, splitTimeInSeconds) {
    var requestData = '{'
            +    '"command" : "splitVideo",'
            +    '"arguments" : {'
            +        '"video" : "' + videoName + '",'
            +        '"splitTimeInSeconds" : ' + splitTimeInSeconds + ''
            +    '}'
            +  '}';
    makeAjaxPostRequest(requestData, refresh, null);    // Defined in "/olive/scripts/master
.js".
}

function attachVideoClickHandlers() {
    $('.video-container').click(function (eventObject) {
        // Don't let the parent div's .click() event fire.
        // Modified from: http://stackoverflow.com/questions/1369035/how-do-i-prevent-a-pare
nts-onclick-event-from-firing-when-a-child-anchor-is-clic
        eventObject.stopPropagation();

        if ($(this).data('isSelected')) {
            unselect(this);
        } else {
            unselectAll();
            select(this);
        }
    });

    $('#videos').click(function () {
        unselectAll();
    });

    $('#timeline').click(function () {
        unselectAll();
    });
}

function attachRenameVideoHandlers() {
```

```
    // Downloaded from: http://www.arashkarimzadeh.com/jquery/7-editable-jquery-plugin.html
    $('.video-name').editable({
        type: 'text',
        submit: 'Save',
        cancel: 'Cancel',
        onEdit: function () {
            doNotSelectThisTime();

            // Restrict input length
            // Another way: http://www.arashkarimzadeh.com/jquery/9-how-to-extend-jquery-edi
table.html
            var maxVideoNameLength = 32;
            $(this).children('input').attr('maxlength', maxVideoNameLength);
        },
        onSubmit: function (content) {
            renameVideo(content.previous, content.current);
        },
        onCancel: function (content) {
        }
    });

    // These don't work.
    $('.video-container input').live('click', function (){
        doNotSelectThisTime();
    });
    $('.video-container button').live('click', function (){
        doNotSelectThisTime();
    });
}

function doNotSelectThisTime() {
    event.stopPropagation();    // Prevent selecting from happening.
}

function attachPublishButtonHandlers(){
    $('#export-button').click(function(){
        $(this).text("Please wait...");
        combineVideos();
    });
}

// Perform a combineVideos request
function combineVideos(){
    var requestData = '{'
        +    '"command" : "combineVideos",'
        +    '"arguments" : {'
        +    '}'
        +  '}';
    makeAjaxPostRequest(requestData, null, null);
}

//Perform a renameVideo request
function renameVideo(oldVideoName, newVideoName) {
    var requestData = '{'
        +    '"command" : "renameVideo",'
        +    '"arguments" : {'
        +        '"oldVideoName" : "' + oldVideoName + '",'
        +        '"newVideoName" : "' + newVideoName + '"'
        +    '}'
        +  '}';
    makeAjaxPostRequest(requestData, refresh, null);    // Defined in "/olive/scripts/master
.js".
}


function makeSelectionVisible(element) {
    if ($(element).data('isSelected')) {
        $(element).css( {
            'background-color': '#edf4e6'   // A lighter version of the Olive color
        });
        $(element).find('.split-video-link').removeClass('hidden');
        $(element).find('.video-controls-divider').removeClass('hidden');
        updatePlayerWithNewElement(element);
    } else {
        $(element).css( {
            'background-color': ''
        });
        $(element).find('.split-video-link').addClass('hidden');
        $(element).find('.video-controls-divider').addClass('hidden');
        updatePlayerWithNoElements();
    }
}

function select(element) {
    $(element).data('isSelected', true);
    makeSelectionVisible(element);
    addToSelected($(element).data('name'));

    attachAutomaticPlaybackHandlers();
}

function unselect(element) {
    $(element).data('isSelected', false);
    makeSelectionVisible(element);
    removeFromSelected($(element).data('name'));
}

function unselectAll() {
    $('.video-container').each(function () {
        unselect(this); // 'this' is a different 'this' than outside .each()
    });
}

//Perform an addToSelected request
function addToSelected(videoName) {
    var requestData = '{'
        +    '"command" : "addToSelected",'
        +    '"arguments" : {'
        +        '"video" : "' + videoName + '"'
        +    '}'
        +  '}';
    makeAjaxPostRequest(requestData, null, null);   // Defined in "/olive/scripts/master.js"
.
}

// Perform a removeFromSelected request
function removeFromSelected(videoName) {
    var requestData = '{'
        +    '"command" : "removeFromSelected",'
        +    '"arguments" : {'
        +        '"video" : "' + videoName + '"'
        +    '}'
        +  '}';
    makeAjaxPostRequest(requestData, null, null);   // Defined in "/olive/scripts/master.js"
.
}
```

```
// Video tag codecs: http://www.webmonkey.com/2010/02/embed_audio_and_video_in_html_5_pages/
// Also: http://stackoverflow.com/questions/2425218/html5-video-tag-in-chrome-wmv
function updatePlayerWithNewElement(element) {
    var video = '<video id="player" preload controls poster="'
        + $(element).data('icon')
        + '">'
        + '<source src="' + $(element).data('url')
        + '" type="video/ogg; codecs=\'theora,vorbis\'" /></video>';    // TODO Get this fro
m the database.
    $('#player-container').append(video);
}

function updatePlayerWithNoElements() {
    $('#player-container').empty();
}

function enableDragAndDrop() {
    $('#videos').sortable( {
        appendTo: 'body',
        connectWith: '#timeline',
        helper: 'clone',
        items: '.video-container',
        revert: true,
        scroll: false,
        tolerance: 'pointer',
        update: function (event, ui) {
            updateVideosPosition();
        }
    });

    $('#timeline').sortable( {
        appendTo: 'body',
        connectWith: '#videos',
        helper: 'clone',
        items: '.video-container',
        revert: true,
        scroll: false,
        tolerance: 'pointer',
        update: function (event, ui) {
            enableOrDisablePublishButton();
            updateTimelinePosition();
        }
    });
}

// Perform an update<command>Position request
function updatePosition(command, collectionItems) {
    var requestData = '{'
        +    '"command" : "' + command + '",'
        +    '"arguments" : {'
        +      '"videos" : [';

    if ($(collectionItems).length > 0) {
        $(collectionItems).each(function(index) {
            requestData += '{'
            +          '"video" : "' + $(this).data('name') + '",'
            +          '"position" : ' + index
            +        '},';  // This will result in an extra comma.
        });

        // Strip off the extra comma.
        requestData = requestData.substring(0, requestData.length - 1);
    }

    requestData += ']}}';

    makeAjaxPostRequest(requestData, null, null);   // Defined in "/olive/scripts/master.js"
.
}

// Perform an updateVideosPosition request
function updateVideosPosition() {
    updatePosition('updateVideosPosition', '#videos > .video-container');
}

// Perform an updateTimelinePosition request
function updateTimelinePosition() {
    updatePosition('updateTimelinePosition', '#timeline > .video-container');
}

function enableOrDisablePublishButton() {
    if ($('#timeline').sortable('toArray').length > 0){
        $('#export-button').removeAttr('disabled');
    } else {
        $('#export-button').attr('disabled', 'disabled');
    }
}

function getNextVideoToPlay() {
    var numberOfVideosInTimeline = $('#timeline > .video-container').length;
    var retval = null;

    $('#timeline > .video-container').each(function (index) {
        if ($(this).data('isSelected')) {
            if (index < numberOfVideosInTimeline - 1) {
                retval = $('#timeline > .video-container').get(index + 1);
            }
        }
    });

    return retval;
}

function selectAndPlayNextVideo() {
    var nextVideoToPlay = getNextVideoToPlay();
    if (nextVideoToPlay !== null) { // If null, the last video was just played; do nothing.
        unselectAll();
        select(nextVideoToPlay);
        var video = $('video').get(0);
        video.play();
    }
}

function attachAutomaticPlaybackHandlers() {
    // Modified from: http://blog.gingertech.net/2009/08/19/jumping-to-time-offsets-in-video
s/
    var video = $('video').get(0);
    video.addEventListener("timeupdate", function() {
        if (video.ended) {
            selectAndPlayNextVideo();
    }}, false);
}
```

```
/*
 * This is Olive's JavaScript file for index.jsp only.
 */

// Called once the DOM is ready but before the images, etc. load.
// Failsafe jQuery code modified from: http://api.jquery.com/jQuery/#jQuery3
jQuery(function($) {
    attachRegistrationHandlers();
});

function attachRegistrationHandlers() {
    var name = $('#register-name'), email = $('#register-email'), password = $('#register-pa
ssword'), cPassword = $('#confirm-register-password'), allFields = $(
            []).add(name).add(email).add(password).add(cPassword);

    $('#dialog-form').dialog({
        autoOpen : false,
        height : 550,
        width : 450,
        modal : true,
        buttons : {
            'Create an account' : function() {
                var bValid = true;
                allFields.removeClass('ui-state-error');

                bValid = bValid
                        && checkLength(name,
                                'register-username',
                                MIN_USERNAME_LENGTH,
                                MAX_USERNAME_LENGTH);
                bValid = bValid
                        && checkLength(email,
                                'register-email',
                                MIN_EMAIL_LENGTH,
                                MAX_EMAIL_LENGTH);
                bValid = bValid
                        && checkLength(password,
                                'register-password',
                                MIN_PASSWORD_LENGTH,
                                MAX_PASSWORD_LENGTH);
                bValid = bValid
                        && checkRegexp(name,
                                SAFE_USERNAME_REGEX,
                                'Username may consist of a-z, 0-9, underscores; and must beg
in with a letter.');
                bValid = bValid
                        && checkRegexp(email,
                                SAFE_EMAIL_REGEX,
                                'Email should be in the form: example@example.com');
                bValid = bValid
                        && checkRegexp(password,
                                SAFE_PASSWORD_REGEX,
                                'Password may consist of letters, numbers, underscores, and
spaces.');
                bValid = bValid
                        && checkPasswordsEqual(password,
                                cPassword,
                                'Passwords do not match.');
                if (bValid) {
                    $('#register-form').submit();
                    $(this).dialog('close');
                }
            },
            Cancel : function() {
                $(this).dialog('close');
            }
        },
        close : function() {
            allFields.val('').change().removeClass('ui-state-error');
        }
    });

    $('#create-user').click(function() {
        $('#dialog-form').dialog('open');
    });
}

$(document).ready(function(){

    var Playlist = function(instance, playlist, options) {
        var self = this;

        this.instance = instance; // String: To associate specific HTML with this playlist
        this.playlist = playlist; // Array of Objects: The playlist
        this.options = options; // Object: The jPlayer constructor options for this playlist

        this.current = 0;

        this.cssId = {
            jPlayer: "jquery_jplayer_",
            interface: "jp_interface_",
            playlist: "jp_playlist_"
        };
        this.cssSelector = {};

        $.each(this.cssId, function(entity, id) {
            self.cssSelector[entity] = "#" + id + self.instance;
        });

        if(!this.options.cssSelectorAncestor) {
            this.options.cssSelectorAncestor = this.cssSelector.interface;
        }

        $(this.cssSelector.jPlayer).jPlayer(this.options);

        $(this.cssSelector.interface + " .jp-previous").click(function() {
            self.playlistPrev();
            $(this).blur();
            return false;
        });

        $(this.cssSelector.interface + " .jp-next").click(function() {
            self.playlistNext();
            $(this).blur();
            return false;
        });
    };

    Playlist.prototype = {
        displayPlaylist: function() {
            var self = this;
            $(this.cssSelector.playlist + " ul").empty();
            for (i=0; i < this.playlist.length; i++) {
                var listItem = (i === this.playlist.length-1) ? "<li class='jp-playlist-last
'>" : "<li>";
                listItem += "<a href='#' id='" + this.cssId.playlist + this.instance + "_ite
```

```
m_" + i +"' tabindex='1'>"+ this.playlist[i].name +"</a>";

                // Create links to free media
                if(this.playlist[i].free) {
                    var first = true;
                    listItem += "<div class='jp-free-media'>(";
                    $.each(this.playlist[i], function(property,value) {
                        if($.jPlayer.prototype.format[property]) { // Check property is a me
dia format.
                            if(first) {
                                first = false;
                            } else {
                                listItem += " | ";
                            }
                            listItem += "<a id='" + self.cssId.playlist + self.instance + "_
item_" + i + "_" + property + "' href='" + value + "' tabindex='1'>" + property + "</a>";
                        }
                    });
                    listItem += ")</span>";
                }

                listItem += "</li>";

                // Associate playlist items with their media
                $(this.cssSelector.playlist + " ul").append(listItem);
                $(this.cssSelector.playlist + "_item_" + i).data("index", i).click(function(
) {
                    var index = $(this).data("index");
                    if(self.current !== index) {
                        self.playlistChange(index);
                    } else {
                        $(self.cssSelector.jPlayer).jPlayer("play");
                    }
                    $(this).blur();
                    return false;
                });

                // Disable free media links to force access via right click
                if(this.playlist[i].free) {
                    $.each(this.playlist[i], function(property,value) {
                        if($.jPlayer.prototype.format[property]) { // Check property is a me
dia format.
                            $(self.cssSelector.playlist + "_item_" + i + "_" + property).dat
a("index", i).click(function() {
                                var index = $(this).data("index");
                                $(self.cssSelector.playlist + "_item_" + index).click();
                                $(this).blur();
                                return false;
                            });
                        }
                    });
                }
            }
        },
        playlistInit: function(autoplay) {
            if(autoplay) {
                this.playlistChange(this.current);
            } else {
                this.playlistConfig(this.current);
            }
        },
        playlistConfig: function(index) {
            $(this.cssSelector.playlist + "_item_" + this.current).removeClass("jp-playlist-
current").parent().removeClass("jp-playlist-current");
            $(this.cssSelector.playlist + "_item_" + index).addClass("jp-playlist-current").
parent().addClass("jp-playlist-current");
            this.current = index;
            $(this.cssSelector.jPlayer).jPlayer("setMedia", this.playlist[this.current]);
        },
        playlistChange: function(index) {
            this.playlistConfig(index);
            $(this.cssSelector.jPlayer).jPlayer("play");
        },
        playlistNext: function() {
            var index = (this.current + 1 < this.playlist.length) ? this.current + 1 : 0;
            this.playlistChange(index);
        },
        playlistPrev: function() {
            var index = (this.current - 1 >= 0) ? this.current - 1 : this.playlist.length -
1;
            this.playlistChange(index);
        }
    };

    var videoPlaylist = new Playlist("1", [
        {
            name:"Uploading Video",
            free:true,
            ogv: "screencasts/uploadingVideo.ogv"
           /* poster:"images/splash-simple.png"*/
        },
        {
            name:"Splitting",
            ogv:"screencasts/split.ogv"

        }
    ], {
        ready: function() {
            videoPlaylist.displayPlaylist();
            videoPlaylist.playlistInit(false); // Parameter is a boolean for autoplay.
        },
        ended: function() {
            videoPlaylist.playlistNext();
        },
        play: function() {
            $(this).jPlayer("pauseOthers");
        },
        swfPath: "../scripts",
        supplied: "ogv"
    });



/*  $("#jplayer_inspector_1").jPlayerInspector({jPlayer:$("#jquery_jplayer_1")});
    $("#jplayer_inspector_2").jPlayerInspector({jPlayer:$("#jquery_jplayer_2")});*/
});
```

```
/*
 * This is Olive's master JavaScript file for every JSP page.
 */

// Called once the DOM is ready but before the images, etc. load.
// Failsafe jQuery code modified from: http://api.jquery.com/jQuery/#jQuery3

// The following should be exact copies of the constants in Security.java.
var MIN_USERNAME_LENGTH = 3;
var MAX_USERNAME_LENGTH = 16;
var SAFE_USERNAME_REGEX = /^[a-zA-z]([0-9a-zA-Z_])+$/;

var MIN_EMAIL_LENGTH = 6;
var MAX_EMAIL_LENGTH = 64;
var SAFE_EMAIL_REGEX = /^((([a-zA-z]|\d|[!#\$%&'\*\+\-\/=\?\^_`{\|}~]|[\u00A0-\uD7FF\uF900-\
uFDCF\uFDF0-\uFFEF])+(\.([a-zA-z]|\d|[!#\$%&'\*\+\-\/=\?\^_`{\|}~]|[\u00A0-\uD7FF\uF900-\uFD
CF\uFDF0-\uFFEF])+)*)|((\x22)((((\x20|\x09)*(\x0d\x0a))?(\x20|\x09)+)?(([\x01-\x08\x0b\x0c\x
0e-\x1f\x7f]|\x21|[\x23-\x5b]|[\x5d-\x7e]|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])|(\\([\x
01-\x09\x0b\x0c\x0d-\x7f]|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])))*(((\x20|\x09)*(\x0d\
x0a))?(\x20|\x09)+)?(\x22)))@((([a-zA-z]|\d|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])|(([a-
zA-z]|\d|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])([a-zA-z]|\d|-|\.|_|~|[\u00A0-\uD7FF\uF90
0-\uFDCF\uFDF0-\uFFEF])*([a-zA-z]|\d|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])))\.)+(([a-zA
-z]|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])|(([a-zA-z]|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-
\uFFEF])([a-zA-z]|\d|-|\.|_|~|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])*([a-zA-z]|[\u00A0-\
uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])))\.?$/;

var MIN_PASSWORD_LENGTH = 5;
var MAX_PASSWORD_LENGTH = 128;
var SAFE_PASSWORD_REGEX = /^[a-zA-Z0-9_]$|^[a-zA-Z0-9_]+[a-zA-Z0-9_ ]*[a-zA-Z0-9_]+$/;

var MIN_PROJECT_NAME_LENGTH = 1;
var MAX_PROJECT_NAME_LENGTH = 32;
var SAFE_PROJECT_NAME_REGEX = SAFE_PASSWORD_REGEX;

var MIN_VIDEO_NAME_LENGTH = 1;
var MAX_VIDEO_NAME_LENGTH = 32;
var SAFE_VIDEO_NAME_REGEX = SAFE_PASSWORD_REGEX;

jQuery(function($) {
    includeHeader(); // TODO Fix for index.jsp (which is different)
    includeFooter();
});

function includeHeader() {
    $.get('/olive/header.jsp', function(data) {
        $('#header').append($(data));
        attachDialogToLink('help');
    });
}

function includeFooter() {
    $.get('/olive/footer.jsp', function(data) {
        $('#footer').append($(data));
        attachDialogToLink('about');
    });
}

function attachDialogToLink(linkName) {
    $('#' + linkName + '-dialog').dialog( {
        autoOpen : false,
        buttons: {
            OK: function() {
                $(this).dialog('close');
            }
        }
    });

    $('#' + linkName + '-dialog-opener').click(function() {
        $('#' + linkName + '-dialog').dialog('open');
        return false;
    });
}

function refresh() {
    window.location.reload();
}

function logout() {
    // See: http://stackoverflow.com/questions/503093/how-can-i-make-a-redirect-page-in-jque
ry
    window.location.replace('logout.jsp');
}

function makeAjaxPostRequest(requestData, onSuccess, onError) {
    // Domain: http://stackoverflow.com/questions/2300771/jquery-domain-get-url
    // E.g. 'http:' + '//' + 'olive.readytalk.com' + '/olive/OliveServlet'
    var postUrl = location.protocol + '//' + location.host + '/olive/OliveServlet';

    // Encoding: http://stackoverflow.com/questions/26620/how-to-set-encoding-in-getjson-jqu
ery
    $.ajax({
        type: 'POST',
        url: postUrl,
        async: true,
        contentType: 'application/json; charset=utf-8',
        data: requestData,
        success: function (responseData) {
            if (onSuccess) {
                onSuccess(responseData);
            }
        },
        error: function (XMLHttpRequest, textStatus, errorThrown) {
            if (onError) {
                onError(XMLHttpRequest, textStatus, errorThrown);
            }
        }
    });
}

/*
 * The following functions are used for input validation
 */

function updateTips(t) {
    var tips = $('.validateTips');

    tips.text(t).addClass('ui-state-highlight');
    setTimeout(function() {
        tips.removeClass('ui-state-highlight', 1500);
    }, 500);
}

function checkLength(o, n, min, max) {
    if (o.val().length > max || o.val().length < min) {
        o.addClass('ui-state-error');
        updateTips('Length of ' + n + ' must be between ' + min + ' and '
```

```
                + max + '.');
        return false;
    } else {
        return true;
    }
}

function checkRegexp(o, regexp, n) {
    if (!(regexp.test(o.val()))) {
        o.addClass('ui-state-error');
        updateTips(n);
        return false;
    } else {
        return true;
    }
}

function checkPasswordsEqual(o,p,n){
    if(!(o.val() == p.val())){
        o.addClass('ui-state-error');
        updateTips(n);
        return false;
    } else {
        return true;
    }
}
```

```
/*
 * This is Olive's JavaScript file for projects.jsp only.
 * Dependencies: "/olive/scripts/master.js"
 */

// Called once the DOM is ready but before the images, etc. load.
// Failsafe jQuery code modified from: http://api.jquery.com/jQuery/#jQuery3
jQuery(function($) {
    populateProjects();
});

function attachHandlers() {
    printFirstSignInMessage();
    attachCreateNewProjectHandlers();
    attachDeleteProjectHandlers();
    attachRenameProjectHandlers();
    enableDragAndDrop();
}

function createNewProjectContainer(projectName, projectNum, projectIcon) {
    projectIcon = '/olive/images/SPANISH OLIVES.jpg';
    var projectContainer = '<div id="project-container-'
        + projectNum
        + '" class="project-container"><a href="OliveServlet?projectName='
        + encodeURI(projectName)    // Ensure a safe link.
        + '"><img id="project-image-'
        + projectNum
        + '" class="project-image" src="'
        + projectIcon
        + '" alt="project-image-'
        + projectNum
        + '" /></a><div class="project-name">'
        + projectName
        + '</div><div class="project-controls"><small><a id="delete-project-link-'
        + projectNum
        + '" class="delete-project-link warning">Delete</a></small></div></div>';

    $('#projects').append(projectContainer);

    // Return the object that was just appended (with the jQuery wrapper stripped off).
    return $('#project-container-' + projectNum).get(0);
}

function createProjectSpinner() {

}

function populateProjects() {
    $('.project-container').hide();

    var requestData = '{'
        +    '"command" : "getProjectInformation"'
        +  '}';
    makeAjaxPostRequest(requestData, function (responseData) {
        var poolPositions = [];
        for (var i = 0; i < responseData.length; ++i) {
            var element = createNewProjectContainer(responseData[i].name, i, responseData[i]
.icon);
            $(element).data('name', responseData[i].name);
            $(element).data('icon', responseData[i].icon);

            // Modified from: http://stackoverflow.com/questions/600700/jquery-javascript-re
ordering-rows/617349#617349
            if (responseData[i].poolPosition != -1) {
                $(element).data('poolPosition', responseData[i].poolPosition);
                poolPositions[(responseData.length - 1) - responseData[i].poolPosition] = el
ement;  // Sort in reverse order to work with prepending.
            }
        }
        // Append in the sorted order
        for (var poolIndex = 0; poolIndex < poolPositions.length; ++poolIndex) {
            $('#projects').prepend(poolPositions[poolIndex]);   // Prepend to keep unsorted
elements (poolPosition == -1) at the end.
        }

        $('.project-container').show();

        attachHandlers();   // This must go inside the ajax callback or it will be called to
o early.
    }, null);   // Defined in "/olive/scripts/master.js".
}

function printFirstSignInMessage() {
    var requestData = '{'
        +    '"command" : "isFirstSignIn"'
        +  '}';
    makeAjaxPostRequest(requestData, function (responseData) {
        if (responseData === true) {
            var welcomeMessage = '<p>Welcome to Olive. This is the projects '
                + 'page where you can add, edit, and delete your projects. We '
                + 'would like to remind you to go to the Account Information '
                + 'page by clicking on your username at the top right and '
                + 'adding a security question and answer in case you forget '
                + 'your password.<br /><br />Thanks!<br /><br />The Olive '
                + 'Team</p>';
            $('#projects').append(welcomeMessage);
        }
    }, null);   // Defined in "/olive/scripts/master.js".
}

function attachCreateNewProjectHandlers() {
    var newProjectName = $('#new-project-name'),
        allFields = $([]).add(newProjectName);

    $('#new-project-dialog-form').dialog({
        autoOpen : false,
        height : 350,
        width : 400,
        modal : true,
        buttons : {
            'Create New Project' : function () {
                var bValid = true;
                allFields.removeClass('ui-state-error');

                bValid = bValid
                        && checkLength(newProjectName,
                                'new-project-name',
                                MIN_PROJECT_NAME_LENGTH,
                                MAX_PROJECT_NAME_LENGTH);
                bValid = bValid
                        && checkRegexp(newProjectName,
                                SAFE_PROJECT_NAME_REGEX,
                                'Project name may consist of letters, numbers, underscores,
and spaces.');
                if (bValid) {
                    $('#new-project-form').submit();
```

```
                    $(this).dialog('close');
                }
            },
            Cancel : function () {
                $(this).dialog('close');
            }
        },
        close : function () {
            allFields.val('').change().removeClass('ui-state-error');
        }
    });

    $('#create-new-project').click(function() {
        $('#new-project-dialog-form').dialog('open');
    });
}

function attachRenameProjectHandlers() {
    // Downloaded from: http://www.arashkarimzadeh.com/jquery/7-editable-jquery-plugin.html
    $('.project-name').editable({
        type: 'text',
        submit: 'Save',
        cancel: 'Cancel',
        onEdit: function () {
            // Restrict input length
            // Another way: http://www.arashkarimzadeh.com/jquery/9-how-to-extend-jquery-edi
table.html
            var maxProjectNameLength = 32;
            $(this).children('input').attr('maxlength', maxProjectNameLength);
        },
        onSubmit: function (content) {
            renameProject(content.previous, content.current);
        },
        onCancel: function (content) {
        }
    });
}

//Perform a renameProject request
function renameProject(oldProjectName, newProjectName) {
    var requestData = '{'
        +    '"command" : "renameProject",'
        +    '"arguments" : {'
        +        '"oldProjectName" : "' + oldProjectName + '",'
        +        '"newProjectName" : "' + newProjectName + '"'
        +    '}'
        +  '}';
    makeAjaxPostRequest(requestData, refresh, null);    // Defined in "/olive/scripts/master
.js".
}

function attachDeleteProjectHandlers() {
    var projectToDelete;

    $('.delete-project-link').click(function () {
        $('#confirm-delete-project-dialog').dialog('open');
        projectToDelete = $(this).parent().parent().parent();
    });

    $('#confirm-delete-project-dialog').dialog({
        autoOpen: false,
        resizable: false,
        height: 275,
        modal: true,
        buttons: {
            'Delete': function () {
                deleteProject($(projectToDelete).data('name'));
                $(this).dialog('close');
            },
            Cancel: function () {
                $(this).dialog('close');
            }
        }
    });
}

function enableDragAndDrop() {
    $('#projects').sortable( {
        appendTo: 'body',
        helper: 'clone',
        items: '.project-container',
        revert: true,
        scroll: false,
        tolerance: 'pointer',
        update: function(event, ui) {
            updateProjectsPosition();
        }
    });
}

//Perform an update<command>Position request
function updatePosition(command, collectionItems) {
    var requestData = '{'
        +    '"command" : "' + command + '",'
        +    '"arguments" : {'
        +      '"projects" : [';

    if ($(collectionItems).length > 0) {
        $(collectionItems).each(function(index) {
            requestData += '{'
            +          '"project" : "' + $(this).data('name') + '",'
            +          '"position" : ' + index
            +        '},';  // This will result in an extra comma.
        });

        // Strip off the extra comma.
        requestData = requestData.substring(0, requestData.length - 1);
    }

    requestData += ']}}';

    makeAjaxPostRequest(requestData, null, null);   // Defined in "/olive/scripts/master.js"
.
}

//Perform an updateProjectsPosition request
function updateProjectsPosition() {
    updatePosition('updateProjectsPosition', '#projects > .project-container');
}

// Perform a deleteProject request
function deleteProject(projectName) {
    var requestData = '{'
            +    '"command" : "deleteProject",'
            +    '"arguments" : {'
            +        '"project" : "' + projectName + '"'
```

```
            +      '}'
            +  '}';
    makeAjaxPostRequest(requestData, refresh, null);    // Defined in "/olive/scripts/master
.js".
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Application Server 9.0 Servlet
2.5//EN" "http://www.sun.com/software/appserver/dtds/sun-web-app_2_5-0.dtd">
<sun-web-app error-url="">
    <context-root>/olive</context-root>
    <class-loader delegate="true" />
    <jsp-config>
        <property name="keepgenerated" value="true">
            <description>Keep a copy of the generated servlet class' java code.</description
>
        </property>
    </jsp-config>
</sun-web-app>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/w
eb-app_2_5.xsd"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/
web-app_2_5.xsd"
    version="2.5">
    <servlet>
        <servlet-name>OliveServlet</servlet-name>
        <servlet-class>com.readytalk.olive.servlet.OliveServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>OliveServlet</servlet-name>
        <url-pattern>/OliveServlet</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>30</session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
    <resource-ref>
        <description>Database</description>
        <res-ref-name>jdbc/OliveData</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <res-auth>Container</res-auth>
    </resource-ref>
</web-app>
```

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="com.readytalk.olive.util.Attribute"%>
<!doctype html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Olive | The Online Video Editor</title>

<link rel="shortcut icon" href="/olive/images/olive.ico" />

<link rel="stylesheet" type="text/css" href="/olive/css/reset.css" />
<link rel="stylesheet" type="text/css"
    href="/olive/scripts/jquery-ui-1.8.9.custom/css/ui-lightness/jquery-ui-1.8.9.custom.css"
 />
<link rel="stylesheet" type="text/css" href="/olive/css/master.css" />
<link rel="stylesheet" type="text/css" href="/olive/css/account.css" />

<script src="/olive/scripts/jquery-1.5.min.js"></script>
<script
    src="/olive/scripts/jquery-ui-1.8.9.custom/js/jquery-ui-1.8.9.custom.min.js"></script>
<script src="/olive/scripts/master.js"></script>
<script src="/olive/scripts/account.js"></script>
<script src="/olive/scripts/google-analytics.js"></script>
</head>
<body>
<%
    Boolean isAuthorized = (Boolean) session
            .getAttribute(Attribute.IS_AUTHORIZED.toString()); // Nasty cast
    if (isAuthorized == null) {
        response.sendRedirect("index.jsp");
    } else if (!isAuthorized) {
        response.sendRedirect("index.jsp");
    }

    Boolean editSuccessfully = (Boolean) session
            .getAttribute(Attribute.EDIT_SUCCESSFULLY.toString());
    Boolean passwordsMatch = (Boolean) session
            .getAttribute(Attribute.PASSWORDS_MATCH.toString());
    String editConfirmation;
    if (editSuccessfully == null) {
        editConfirmation = "";
    } else if (editSuccessfully) {
        editConfirmation = "Your information has been changed successfully.";
    } else {
        if (passwordsMatch == null) { // Not safe
            editConfirmation = "Invalid characters or length on one or more fields.";
        } else { // Safe, but differing passwords
            editConfirmation = "Passwords do not match.";
        }
    }
    session.removeAttribute(Attribute.EDIT_SUCCESSFULLY.toString());
    session.removeAttribute(Attribute.PASSWORDS_MATCH.toString());

    String username = (String) session.getAttribute(Attribute.USERNAME
            .toString());
%>
<div id="header">
<div id="header-left">
<img id="olive-icon"
    src="/olive/images/olive.png" />
<h1 id= "olive-title">Olive</h1>
</div>
<!-- end #header-left -->
<div id="header-right">
<div>Welcome, <%=username%>! <a href="logout.jsp">Logout</a></div>
<div><strong><a href="projects.jsp">My Projects</a></strong> <span
    id="help-dialog-opener"><a href="">Help</a></span></div>
</div>
<!-- end #header-right --></div>
<!-- end #header -->

<div class="clear"></div>

<div id="main">

<div id="edit-account-container">
<h2>Edit account information</h2>
<form id="edit-account-form" action="OliveServlet" name="process"
    method="post">
<p><label for="new-name">Name</label> <input type="text"
    name="new-name" id="new-name" size="32" maxlength="32" /></p>
<p><label for="new-email">Email</label> <input type="text"
    name="new-email" id="new-email" size="32" maxlength="64" /></p>
<p><label for="new-password">Password</label> <input type="password"
    name="new-password" id="new-password" size="32" maxlength="128" /></p>
<p><label for="confirm-new-password">Confirm password</label> <input
    type="password" name="confirm-new-password" id="confirm-new-password"
    size="32" maxlength="128" /></p>
<p><label for="new-security-question">Security Question</label> <input
    type="text" name="new-security-question" id="new-security-question"
    size="32" maxlength="128" /></p>
<p><label for="new-security-answer">Security Answer</label> <input
    type="text" name="new-security-answer" id="new-security-answer"
    size="32" maxlength="128" /></p>
<input type="hidden" name="FormName" value="EditUser"></input> <input
    type="submit" value="Update information" /><span><%=editConfirmation%></span></form>
<p><small><a id="delete-account"
    class="warning delete-project">Delete account</a></small></p>
</div>
<!-- end #login-form-container --></div>
<!-- end #main -->

<div class="clear"></div>

<div id="footer"></div>

<!-- Everything below this line will be hidden and inserted in pop-ups. -->
<div id="help-dialog" class="hidden" title="How to use Olive">
<ul>
    <li>1. Create a new account.</li>
    <li>2. Create a new project.</li>
    <li>3. Upload your videos.</li>
    <li>4. Edit your videos.</li>
    <li>5. Export to your computer.</li>
</ul>
</div>

<div id="confirm-delete-account-dialog" class="hidden" title="Warning!">
<p>Delete account? This will also delete the account’s projects and
videos.</p>
</div>

</body>
</html>
```

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="com.readytalk.olive.logic.DatabaseApi"%>
<%@ page import="com.readytalk.olive.util.Attribute"%>
<!doctype html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Edit Project | Olive</title>

<link rel="shortcut icon" href="/olive/images/olive.ico">

<link rel="stylesheet" type="text/css" href="/olive/css/reset.css" />
<link rel="stylesheet" type="text/css"
    href="/olive/scripts/jquery-ui-1.8.9.custom/css/ui-lightness/jquery-ui-1.8.9.custom.css"
 />
<link rel="stylesheet" type="text/css" href="/olive/css/master.css" />
<link rel="stylesheet" type="text/css" href="/olive/css/editor.css" />

<script src="/olive/scripts/jquery-1.5.min.js"></script>
<script
    src="/olive/scripts/jquery-ui-1.8.9.custom/js/jquery-ui-1.8.9.custom.min.js"></script>
<script src="/olive/scripts/contextMenu.js"></script>
<script src="/olive/scripts/jquery.editable-1.3.3.min.js"></script>
<script src="/olive/scripts/master.js"></script>
<script src="/olive/scripts/editor.js"></script>
<script src="/olive/scripts/google-analytics.js"></script>
</head>
<body>
<%
    Boolean isAuthorized = (Boolean) session
            .getAttribute(Attribute.IS_AUTHORIZED.toString()); // Nasty cast
    String username = "";
    String projectName = "";
    if (isAuthorized == null || !isAuthorized) { // Short-circuiting
        response.sendRedirect("index.jsp");
    } else {
        projectName = (String) session
                .getAttribute(Attribute.PROJECT_NAME.toString());
        if (projectName == null) {
            response.sendRedirect("projects.jsp");
        } else {
            username = (String) session.getAttribute(Attribute.USERNAME
                    .toString());
        }
    }
%>
<div id="header">
<div id="header-left">
<img id="olive-icon"
    src="/olive/images/olive.png" />
<h1 id= "olive-title">Olive</h1>
</div>
<!-- end #header-left -->
<div id="header-right">
<div>Welcome, <a href="account.jsp"><%=username%>!</a> <a
    href="logout.jsp">Logout</a></div>
<div><strong><a href="projects.jsp">My Projects</a></strong> <span
    id="help-dialog-opener"><a href="">Help</a></span></div>
</div>
<!-- end #header-right --></div>
<!-- end #header -->

<div class="clear"></div>

<div id="main">

<div id="videos-container">
<div id="videos-header"><span id="videos-title"> <%=projectName%>
</span> <span id="videos-controls">
<button id="upload-new-video-button" type="button"
    class="ui-button ui-widget ui-state-default ui-corner-all ui-button-text-only ui-state-h
over"><span
    class="ui-button-text">Upload New Video</span></button>
</span></div>
<div id="videos"></div>
<!-- end #videos --></div>
<!-- end #videos-container -->

<div class="contextMenu" id="video-context-menu">
<ul>
    <li id="split-video-menu-item">Split Video</li>
</ul>
</div>
<!-- end #context-menu -->

<div id="player-container"></div>

<div class="clear"></div>

<div id="timeline">
<div id="ticks-container"></div>
</div>

<div class="clear"></div>

<div id="export">
<form id="combine-and-export-form" action="OliveServlet" name="process"
    method="post">
<input type="hidden" name="FormName" value="combine-form"></input>
<input type="submit" value="Export to Computer"></input>
</form>

</div>
<!-- end #export --></div>
<!-- end #main -->

<div class="clear"></div>

<div id="footer"></div>

<!-- Everything below this line will be hidden and inserted in pop-ups. -->
<div id="help-dialog" class="hidden" title="How to use Olive">
<ul>
    <li>1. Create a new account.</li>
    <li>2. Create a new project.</li>
    <li>3. Upload your videos.</li>
    <li>4. Edit your videos.</li>
    <li>5. Export to your computer.</li>
</ul>
</div>

<div id="confirm-delete-video-dialog" class="hidden" title="Warning!">
<p>Delete video?</p>
</div>

<div id="confirm-add-to-timeline-dialog" class="hidden"
```

```
    title="Attention!">
<p>Add Video to Timeline</p>
</div>

<div id="invalid-split-dialog" class="hidden" title="Warning">
<p>Please pause the video at a valid split location.</p>
</div>
<!-- end #dialog-form -->

<div id="new-video-dialog-form" class="hidden" title="Create new user">
<p class="validateTips">All form fields are required.</p>
<form id="new-video-form" action="OliveServlet" name="process"
    enctype="multipart/form-data" method="post">
<fieldset><input type="hidden" name="FormName"
    value="UploadVideo"></input><input type="file" id="new-video-file"
    name="file" /> <br />
<label for="new-video-name">Give the video a new name</label> <input
    type="text" name="new-video-name" id="new-video-name"
    class="text ui-widget-content ui-corner-all" maxlength="32" /></fieldset>
</form>
</div>
<!-- end #new-video-dialog-form -->

</body>
</html>
```

```
<div id="footer-left"><span id="about-dialog-opener"><a
    href="">About Us</a></span></div>
<div id="about-dialog" title="About Olive">
<p>&copy; 2010 ReadyTalk</p>
</div>
<!-- Everything below this line will be hidden and inserted in pop-ups. -->
<div id="footer-right" class="hidden">&copy; 2010 ReadyTalk</div>
```

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="com.readytalk.olive.util.Attribute"%>
<!doctype html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Forgot Password? | The Online Video Editor</title>

<link rel="shortcut icon" href="/olive/images/olive.ico" />

<link rel="stylesheet" type="text/css" href="/olive/css/reset.css" />
<link rel="stylesheet" type="text/css"
    href="/olive/scripts/jquery-ui-1.8.9.custom/css/ui-lightness/jquery-ui-1.8.9.custom.css"
 />
<link rel="stylesheet" type="text/css" href="/olive/css/master.css" />
<link rel="stylesheet" type="text/css" href="/olive/css/account.css" />

<script src="/olive/scripts/jquery-1.5.min.js"></script>
<script
    src="/olive/scripts/jquery-ui-1.8.9.custom/js/jquery-ui-1.8.9.custom.min.js"></script>
<script src="/olive/scripts/master.js"></script>
<script src="/olive/scripts/account.js"></script>
<script src="/olive/scripts/google-analytics.js"></script>
</head>
<body>
<%
    Boolean isSafe = (Boolean) session.getAttribute(Attribute.IS_SAFE
            .toString());
    Boolean isCorrect = (Boolean) session
            .getAttribute(Attribute.IS_CORRECT.toString());
    String confirmation = "";
    if (isCorrect == null) {
        confirmation = "";
    } else if (isCorrect == false) {
        if (isSafe == null) {
            confirmation = "";
        } else if (isSafe) {
            confirmation = "Username does not exist or a security question has not been ente
red yet";
        } else {
            confirmation = "Unsafe input";
        }
    } else if (isCorrect) {
        confirmation = "";
        response.sendRedirect("securityQuestion.jsp");
    }
    session.removeAttribute(Attribute.IS_SAFE.toString());
    session.removeAttribute(Attribute.IS_CORRECT.toString());
%>
<div id="header">
<div id="header-left">
<img id="olive-icon"
    src="/olive/images/olive.png" />
<h1 id= "olive-title">Olive</h1>
</div>
<!-- end #header-left -->
<div id="header-right">
<div><strong><a href="index.jsp">Home</a></strong> <span
    id="help-dialog-opener"><a href="">Help</a></span></div>
</div>
<!-- end #header-right --></div>

<!-- end #header -->

<div class="clear"></div>

<div id="main">
<div id="edit-account-container">
<h2>Forgot Password?</h2>
<p>Please enter your username to retrieve your security question.
Thank you</p>
<!-- end #about-title -->

<form id="security-question-form" action="OliveServlet" name="process"
    method="post">
<p><label for="username">Username</label><br />
<input type="text" name="username" id="username" value="" size="32"
    maxlength="32" /></p>
<input type="hidden" name="FormName" value="security-question-form"></input><br />
<input type="submit" value="Get Security Question" /><span><%=confirmation%></span>
</form>
</div>
<!-- end #main --></div>

<div id="footer"></div>

<!-- Everything below this line will be hidden and inserted in pop-ups. -->
<div id="help-dialog" class="hidden" title="How to use Olive">
<ul>
    <li>1. Create a new account.</li>
    <li>2. Create a new project.</li>
    <li>3. Upload your videos.</li>
    <li>4. Edit your videos.</li>
    <li>5. Export to your computer.</li>
</ul>
</div>

</body>
</html>
```

```
<!-- This can’t be used yet because of JSP complications (how to pass around the Java variab
les). -->
```

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="com.readytalk.olive.util.Attribute"%>
<!doctype html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Olive | The Online Video Editor</title>

<link rel="shortcut icon" href="/olive/images/olive.ico" />

<link rel="stylesheet" type="text/css" href="/olive/css/reset.css" />
<link rel="stylesheet" type="text/css"
    href="/olive/scripts/jquery-ui-1.8.9.custom/css/ui-lightness/jquery-ui-1.8.9.custom.css"
 />
<link rel="stylesheet" type="text/css" href="/olive/css/master.css" />
<link rel="stylesheet" type="text/css" href="/olive/css/index.css" />
<link href="/olive/css/skin/jplayer.blue.monday.css" rel="stylesheet" type="text/css" />

<script src="/olive/scripts/jquery-1.5.min.js"></script>
<script
    src="/olive/scripts/jquery-ui-1.8.9.custom/js/jquery-ui-1.8.9.custom.min.js"></script>
<script src="/olive/scripts/jquery.jplayer.min.js"></script>
<script src="/olive/scripts/master.js"></script>
<script src="/olive/scripts/index.js"></script>
<script src="/olive/scripts/google-analytics.js"></script>
</head>
<body>
<%
    Boolean isAuthorized = (Boolean) session
            .getAttribute(Attribute.IS_AUTHORIZED.toString()); // Nasty cast
    String loginMessage;
    if (isAuthorized == null) {
        loginMessage = "Please log in.";
    } else if (isAuthorized) {
        loginMessage = "You have been successfully logged in.";
    } else {
        loginMessage = "Incorrect username and/or password.";
    }
%>
<div id="header">
<div id="header-left">
<img id="olive-icon"
    src="/olive/images/olive.png" />
<h1 id= "olive-title">Olive</h1>
</div>
<!-- end #header-left -->
<div id="header-right"><span id="help-dialog-opener"><a
    href="">Help</a></span></div>
<!-- end #header-right --></div>
<!-- end #header -->

<div class="clear"></div>

<div id="main">
<div id="screencast-container">
        <div class="jp-video jp-video-360p">
            <div class="jp-type-playlist">
                <div id="jquery_jplayer_1" class="jp-jplayer"></div>
                <div id="jp_interface_1" class="jp-interface">
                    <div class="jp-video-play"></div>
                    <ul class="jp-controls">

                        <li><a href="#" class="jp-play" tabindex="1">play</a></li>
                        <li><a href="#" class="jp-pause" tabindex="1">pause</a></li>
                        <li><a href="#" class="jp-stop" tabindex="1">stop</a></li>
                        <li><a href="#" class="jp-mute" tabindex="1">mute</a></li>
                        <li><a href="#" class="jp-unmute" tabindex="1">unmute</a></li>
                        <li><a href="#" class="jp-previous" tabindex="1">previous</a></li>

                        <li><a href="#" class="jp-next" tabindex="1">next</a></li>
                    </ul>
                    <div class="jp-progress">
                        <div class="jp-seek-bar">
                            <div class="jp-play-bar"></div>
                        </div>
                    </div>
                    <div class="jp-volume-bar">

                        <div class="jp-volume-bar-value"></div>
                    </div>
                    <div class="jp-current-time"></div>
                    <div class="jp-duration"></div>
                </div>
                <div id="jp_playlist_1" class="jp-playlist">
                    <ul>
                        <!-- The method Playlist.displayPlaylist() uses this unordered list
-->
                        <li></li>

                    </ul>
                </div>
            </div>
        </div>


    </div>
<!-- end #splash-container -->

<div id="login-form-container">
<form id="login-form" action="OliveServlet" name="process" method="post">
<p><label for="username">Username</label> <input type="text"
    name="username" id="login-username" size="32" maxlength="16" /></p>
<p><label for="password">Password</label> <input type="password"
    name="password" id="login-password" size="32" maxlength="128" /></p>
<input type="hidden" name="FormName" value="LoginUser"></input> <input
    type="submit" value="Login" /><br />
<span><%=loginMessage%> <a href="forgot.jsp">Forgot password?</a></span></form>
<p>Don’t have an account? <a id="create-user" href="javascript:;"
    title="">Sign up!</a></p>
</div>
<!-- end #login-form-container --></div>
<!-- end #main -->

<div id="footer"></div>

<!-- Everything below this line will be hidden and inserted in pop-ups. -->
<div id="help-dialog" class="hidden" title="How to use Olive">
<ul>
    <li>1. Create a new account.</li>
    <li>2. Create a new project.</li>
    <li>3. Upload your videos.</li>
    <li>4. Edit your videos.</li>
    <li>5. Export to your computer.</li>
</ul>
</div>
```

```
<div id="dialog-form" title="Create new user">
<p class="validateTips">All form fields are required.</p>
<form id="register-form" action="OliveServlet" name="process"
    method="post">
<fieldset><label for="name">Username</label> <input
    type="text" name="name" id="register-name"
    class="text ui-widget-content ui-corner-all" maxlength="16" /> <label
    for="email">Email</label> <input type="text" name="email"
    id="register-email" value=""
    class="text ui-widget-content ui-corner-all" maxlength="64" /> <label
    for="password">Password</label> <input type="password" name="password"
    id="register-password" value=""
    class="text ui-widget-content ui-corner-all" maxlength="128" /> <label
    for="confirm_password">Confirm Password</label> <input type="password"
    name="confirm_password" id="confirm-register-password" value=""
    class="text ui-widget-content ui-corner-all" maxlength="128" /></fieldset>
<input type="hidden" name="FormName" value="AddUser"></input></form>
</div>
<!-- end #dialog-form -->

</body>
</html>
```

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="com.readytalk.olive.util.Attribute"%>
<!doctype html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Logout | Olive</title>
<script src="/olive/scripts/google-analytics.js"></script>
</head>
<body>
<%
    // Modified from: http://stackoverflow.com/questions/1104975/for-loop-to-interate-over-e
num-in-java
    for (Attribute attribute : Attribute.values()) {
        session.removeAttribute(attribute.toString());
    }
    response.sendRedirect("index.jsp");
%>
</body>
</html>
```

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="com.readytalk.olive.util.Attribute"%>
<!doctype html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Forgot Password? | The Online Video Editor</title>

<link rel="shortcut icon" href="/olive/images/olive.ico" />

<link rel="stylesheet" type="text/css" href="/olive/css/reset.css" />
<link rel="stylesheet" type="text/css"
    href="/olive/scripts/jquery-ui-1.8.9.custom/css/ui-lightness/jquery-ui-1.8.9.custom.css"
 />
<link rel="stylesheet" type="text/css" href="/olive/css/master.css" />
<link rel="stylesheet" type="text/css" href="/olive/css/account.css" />

<script src="/olive/scripts/jquery-1.5.min.js"></script>
<script
    src="/olive/scripts/jquery-ui-1.8.9.custom/js/jquery-ui-1.8.9.custom.min.js"></script>
<script src="/olive/scripts/master.js"></script>
<script src="/olive/scripts/account.js"></script>
<script src="/olive/scripts/google-analytics.js"></script>
</head>
<body>
<%
    Boolean isSafe = (Boolean) session.getAttribute(Attribute.IS_SAFE
            .toString());
    Boolean passwordsMatch = (Boolean) session
            .getAttribute(Attribute.PASSWORDS_MATCH.toString());
    Boolean editSuccess = (Boolean) session
            .getAttribute(Attribute.EDIT_SUCCESSFULLY.toString());
    String confirmation = "";
    String index = "'index.jsp'";
    if (editSuccess == null) {
        confirmation = "";
    } else if (editSuccess == false) {
        if (isSafe) {
            if (passwordsMatch) {
                confirmation = "We're sorry, there is an error in inputting to the database.
 Please try again";
            } else {
                confirmation = "Passwords do not match";
            }
        } else {
            confirmation = "Unsafe input";
        }
    } else if (editSuccess) {
        confirmation = "Congratulations. Your new password has been set. You "
                + "may now sign in to <a href = "
                + index
                + ">Olive</a>";
    }
    session.removeAttribute(Attribute.IS_CORRECT.toString());
    session.removeAttribute(Attribute.IS_SAFE.toString());
    session.removeAttribute(Attribute.PASSWORDS_MATCH.toString());
    session.removeAttribute(Attribute.EDIT_SUCCESSFULLY.toString());
%>
<div id="header">
<div id="header-left">
<img id="olive-icon"
    src="/olive/images/olive.png" />
<h1 id= "olive-title">Olive</h1>
</div>
<!-- end #header-left -->
<div id="header-right"><span id="help-dialog-opener"><a
    href="">Help</a></span></div>
<!-- end #header-right --></div>
<!-- end #header -->

<div class="clear"></div>

<div id="main">
<div id="edit-account-container">
<h2>New Password</h2>
<p>Please enter a new password for your account</p>
<!-- end #about-title -->

<form id="password-form" action="OliveServlet" name="process"
    method="post">
<p><label for="password">Password</label><br />
<input type="password" name="password" id="password" value="" size="32"
    maxlength="128" /></p>
<p><label for="password">Confirm Password</label><br />
<input type="password" name="confirm_password" id="confirm_password"
    value="" size="32" maxlength="128" /></p>
<input type="hidden" name="FormName" value="new_password"></input><br />
<input type="submit" value="Submit New Password" /><span><%=confirmation%></span></form>
</div>
<!-- end #main --></div>

<div id="footer"></div>

<!-- Everything below this line will be hidden and inserted in pop-ups. -->
<div id="help-dialog" class="hidden" title="How to use Olive">
<ul>
    <li>1. Create a new account.</li>
    <li>2. Create a new project.</li>
    <li>3. Upload your videos.</li>
    <li>4. Edit your videos.</li>
    <li>5. Export to your computer.</li>
</ul>
</div>

</body>
</html>
```

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="com.readytalk.olive.logic.DatabaseApi"%>
<%@ page import="com.readytalk.olive.util.Attribute"%>
<!doctype html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>My Projects | Olive</title>

<link rel="shortcut icon" href="/olive/images/olive.ico">

<link rel="stylesheet" type="text/css" href="/olive/css/reset.css" />
<link rel="stylesheet" type="text/css"
    href="/olive/scripts/jquery-ui-1.8.9.custom/css/ui-lightness/jquery-ui-1.8.9.custom.css"
 />
<link rel="stylesheet" type="text/css" href="/olive/css/master.css" />
<link rel="stylesheet" type="text/css" href="/olive/css/projects.css" />

<script src="/olive/scripts/jquery-1.5.min.js"></script>
<script
    src="/olive/scripts/jquery-ui-1.8.9.custom/js/jquery-ui-1.8.9.custom.min.js"></script>
<script src="/olive/scripts/jquery.editable-1.3.3.min.js"></script>
<script src="/olive/scripts/master.js"></script>
<script src="/olive/scripts/projects.js"></script>
<script src="/olive/scripts/google-analytics.js"></script>
</head>
<body>
<%
    Boolean isAuthorized = (Boolean) session
            .getAttribute(Attribute.IS_AUTHORIZED.toString()); // Nasty cast
    String username = "";
    if (isAuthorized == null) {
        response.sendRedirect("index.jsp");
    } else if (!isAuthorized) {
        response.sendRedirect("index.jsp");
    } else {
        username = (String) session.getAttribute(Attribute.USERNAME
                .toString());
    }
%>
<div id="header">
<div id="header-left">
<img id="olive-icon"
    src="/olive/images/olive.png" />
<h1 id= "olive-title">Olive</h1>
</div>
<div id="header-right">
<div>Welcome, <a href="account.jsp"><%=username%>!</a> <a
    href="logout.jsp">Logout</a></div>
<div><strong><a href="projects.jsp">My Projects</a></strong> <span
    id="help-dialog-opener"><a href="">Help</a></span></div>
</div>
</div>

<div class="clear"></div>

<div id="main">
<div id="projects-container">

<div id="projects-title">
<h1>My Projects</h1>
</div>
<!-- end #projects-title -->

<div id="projects-controls">
<button id="create-new-project" type="button"
    class="ui-button ui-widget ui-state-default ui-corner-all ui-button-text-only ui-state-h
over"><span
    class="ui-button-text">Create New Project</span></button>
</div>
<!-- end #controls -->

<div class="clear"></div>
<div id="projects"></div>
<!-- end #projects --></div>
<!-- end #projects-container --></div>
<!-- end #main -->

<div class="clear"></div>

<div id="footer"></div>

<!-- Everything below this line will be hidden and inserted in pop-ups. -->
<div id="first-sign-in-dialog" class="hidden" title="Welcome to Olive!">
<p>Welcome to Olive. This is the projects page where you
can add, edit, and delete your projects. We would like to
remind you to go to the Account Information page by clicking
on your username at the top right and adding a security question
and answer in case you forget your password.<br />Thanks!<br />The Olive Team</p>
</div>

<div id="help-dialog" class="hidden" title="How to use Olive">
<ul>
    <li>1. Create a new account.</li>
    <li>2. Create a new project.</li>
    <li>3. Upload your videos.</li>
    <li>4. Edit your videos.</li>
    <li>5. Export to your computer.</li>
</ul>
</div>

<div id="confirm-delete-project-dialog" class="hidden" title="Warning!">
<p>Delete project? This will also delete the project’s videos.</p>
</div>

<div id="new-project-dialog-form" class="hidden" title="Create new user">
<p class="validateTips">All form fields are required.</p>
<form id="new-project-form" action="OliveServlet" name="process"
    method="post">
<fieldset><label for="new-project-name">Project Name</label>
<input type="text" name="new-project-name" id="new-project-name"
    class="text ui-widget-content ui-corner-all" maxlength="32" /></fieldset>
<input type="hidden" name="FormName" value="AddProject"></input></form>
</div>
<!-- end #new-project-dialog-form -->

</body>
</html>
```