

02/26/11  
01:45:15

# Olive README

1

Olive makes it easy to edit your videos.

```
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-Inner-Class
    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-Inner-Class
    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-Inner-Class
    public static class Arguments {

        // No-args constructor
        public Arguments() {

        }

    }

}
```

```
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-Inner-Class
    public static class Arguments {

        // No-args constructor
        public Arguments() {

        }

        public String video;
        public double splitTimeInSeconds;

    }

}
```

## src/com/readytalk/olive/logic/HttpSenderReceiver.java

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

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

import com.readytalk.olive.model.Video;

// Modified from: http://www.exampledepot.com/egs/java.net/post.html
public class HttpSenderReceiver {
    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.");

        // Get the response
        System.out.println("Receiving data from Zencoder...");
        BufferedReader bufferedReader = new BufferedReader(
            new InputStreamReader(conn.getInputStream()));
        String line;
        String response = "";
        while ((line = bufferedReader.readLine()) != null) {
            response += line + System.getProperty("line.separator");
        }
        outputStreamWriter.close();
        bufferedReader.close();
        System.out.println(response);
        System.out.println("Data received from Zencoder.");

        return response;
    }

    public static Video[] split(int videoId, double splitTimeInSeconds)
        throws IOException {
        String originalVideoUrl = OliveDatabaseApi.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 underst
and this.
        Video[] videoFragments = new Video[2];

        // Request the first and second halves of the original video.
        for (int i = 0; i < 2; ++i) {
            String videoFragmentFileName = S3Api
                .getNameFromUrlWithNewTimeStamp(originalVideoUrl);
            HttpSenderReceiver.sendReceive(HttpSenderReceiver.getZencoderJson(
                originalVideoUrl, awsBaseUrl, videoFragmentFileName,
                splitStartInSeconds[i], clipLengthInSeconds[i]), new URL(
                    getZencoderUrl()));
            videoFragments[i] = new Video(
```

```
                OliveDatabaseApi.getVideoName(videoId) + i,
                S3Api.AWS_URL_PREFIX + videoFragmentFileName,
                "/olive/images/bbb480.jpg", -1, -1); // TODO Get icon from Zencoder
        }

        return videoFragments;
    }

    private static String getZencoderJson(String input, String baseUrl,
        String filename, double startClip, double clipLength) {
        String data = "{\"api_key\":\"" + OliveDatabaseApi.getZencoderApiKey()
            + "\",\"input\":\"" + input + "\",\"
            + \"output\":{\"\"base_url\":\"" + baseUrl + "\",\"
            + \"filename\":\"" + filename + "\",\"public\":1,\"
            + \"start_clip\":\"" + startClip + "\",\"clip_length\":\"
            + clipLength + "\"}}";
        System.out.println(data);
        return data;
    }

    private static String getZencoderUrl() {
        return "https://app.zencoder.com/api/jobs";
    }
}
```

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

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
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;

public class OliveDatabaseApi {

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

    // CAUTION: closeConnection() must be called sometime after this method.
    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;
    }

    public static void closeConnection(Connection c) {
        try {
            c.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    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!
}

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;"
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;
}

public static int getAccountId(String username) {
    return Integer.parseInt(getUnknownValueFromTable("AccountId",
        "Accounts", "Username", username));
}

public static String getAccountUsername(int accountId) {
    return getUnknownValueFromTable("Username", "Accounts", "AccountId",
        Integer.toString(accountId));
}

public static String getAccountPassword(int accountId) {
    return getUnknownValueFromTable("Password", "Accounts", "AccountId",
        Integer.toString(accountId));
}
```

```
public static String getAccountName(int accountId) {
    return getUnknownValueFromTable("Name", "Accounts", "AccountId",
        Integer.toString(accountId));
}

public static String getAccountEmail(int accountId) {
    return getUnknownValueFromTable("Email", "Accounts", "AccountId",
        Integer.toString(accountId));
}

public static String getAccountSecurityQuestion(int accountId) {
    return getUnknownValueFromTable("SecurityQuestion", "Accounts",
        "AccountId", Integer.toString(accountId));
}

public static String getAccountSecurityAnswer(int accountId) {
    return getUnknownValueFromTable("SecurityAnswer", "Accounts",
        "AccountId", Integer.toString(accountId));
}

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;
}

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);
    }
}

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;
}

public static String getProjectName(int projectId) {
```



```

        return getUnknownValueFromTable("Name", "Projects", "ProjectID",
            Integer.toString(projectId));
    }

    public static int getProjectAccountId(int projectId) {
        return Integer.parseInt(getUnknownValueFromTable("AccountID",
            "Projects", "ProjectID", Integer.toString(projectId)));
    }

    public static String getProjectIcon(int projectId) {
        return getUnknownValueFromTable("Icon", "Projects", "ProjectID",
            Integer.toString(projectId));
    }

    public static String populateProjects(int accountId) {
        String projects = "";
        Connection conn = getDBConnection();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);
            s = "SELECT * FROM Projects WHERE AccountID = '" + accountId + "'";
            ResultSet r = st.executeQuery(s);
            if (r.first()) {
                int projectNum = 0;
                do {
                    projectNum += 1; // TODO This is never used.
                    String projectName = r.getString("Name");
                    String projectIcon = "/olive/images/SPANISH OLIVES.jpg";
                    projects += "<div id=\""
                        + projectName
                        + "\" class=\"project-icon-container\">"
                        + "\n"
                        + "<a href=\"OliveServlet?projectName="
                        + projectName
                        + "\"><img src=\""
                        + projectIcon
                        + "\" class=\"project-icon\" alt=\""
                        + projectName
                        + "\" /></a>"
                        + "\n"
                        + "<p><a href=\"OliveServlet?projectName="
                        + projectName
                        + "\">"
                        + projectName
                        + "</a></p>"
                        + "\n"
                        + "<p><small><a id=\"" // TODO Assign the videoName elsewhere fo
r the JavaScript to access.
                        + projectName
                        + "\" class=\"warning delete-project\">Delete</a></small></p>"
                        + "\n" + "</div>" + "\n";
                } while (r.next());
            }
            return projects;
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            closeConnection(conn);
        }
        return projects;
    }
    // TODO change db to have unique usernames for accounts and names for
    // both projects and videos in one project

```

```

    }

    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;
    }

    public static boolean AddProject(Project project) {
        Connection conn = getDBConnection();
        try {
            Statement st = conn.createStatement();
            String s = "USE OliveData;";
            st.executeUpdate(s);
            Boolean exists = projectExists(project.getName(),
                project.getAccountId());
            if (exists) {
                return false;
            }

            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;
    }

    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

```

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

// 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;
}

public static String getVideoName(int videoId) {
    return getUnknownValueFromTable("Name", "Videos", "VideoID",
        Integer.toString(videoId));
}

public static String getVideoUrl(int videoId) {
    return getUnknownValueFromTable("URL", "Videos", "VideoID",
        Integer.toString(videoId));
}

public static String getVideoIcon(int videoId) {
    return getUnknownValueFromTable("Icon", "Videos", "VideoID",
        Integer.toString(videoId));
}

public static int getVideoProjectId(int videoId) {
    return Integer.parseInt(getUnknownValueFromTable("ProjectID", "Videos",
        "VideoID", Integer.toString(videoId)));
}

public static int getVideoStartTimeStoryboard(int videoId) {
    String videoStartTimeStoryboard = getUnknownValueFromTable(
        "StartTimeStoryboard", "Videos", "VideoID",
        Integer.toString(videoId));
    if (videoStartTimeStoryboard == null) {
        return -1; // TODO Is this a good idea?
    }
    return Integer.parseInt(videoStartTimeStoryboard);
}
```

```
public static String populateVideos(int projectId) {
    String videos = "";
    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);
        if (r.first()) {
            int videoNum = 0;
            do {
                videoNum += 1; // TODO This is inconsistent with projects.
                String videoName = r.getString("Name");
                String videoIcon = "/olive/images/olive.png";

                videos += "<span id=\""
                    + videoName
                    + "\" class=\"video-container\"><img id=\"olive\"
                    + videoNum
                    + "\" class=\"video-icon\"
                    + "\n"
                    + "src=\""
                    + videoIcon
                    + "\" alt=\"olive\"
                    + videoNum
                    + "\" /><br />"
                    + "\n"
                    + videoName
                    + "<br />"
                    + "\n"
                    + "<small><a id=\"" // TODO Assign the videoName elsewhere for t
                    + videoName
                    + "\" class=\"warning delete-video\">Delete</a></small> </span>"
                    + "\n";
            } while (r.next());
        }
        return videos;
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        closeConnection(conn);
    }
    return videos;
    // TODO change db to have unique usernames for accounts and names for
    // both projects and videos in one project
}

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;
}

public static void AddVideo(String name, String url, int projectId,
    String icon) {
    Connection conn = getDBConnection();
    try {
        Statement st = conn.createStatement();
        String s = "USE OliveData;";
        st.executeUpdate(s);
        s = "INSERT INTO Videos (Name, URL, ProjectID, Icon) VALUES ('"
            + name + "', '" + url + "', '" + projectId + "', '" + icon
            + "');"
        st.executeUpdate(s);
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        closeConnection(conn);
    }
}

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

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);
}

public static String getZencoderApiKey() {
    String zencoderApiKey = "";
    Connection conn = OliveDatabaseApi.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;
}

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!
}

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;
}
```

## src/com/readytalk/olive/logic/S3Api.java

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

import java.io.File;
import java.io.IOException;
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.Video;
import com.readytalk.olive.util.InvalidFileSizeException;

// JetS3t code samples (in Java): https://bitbucket.org/jmurty/jets3t/src/Release-0_8_0/src/org/jets3t/samples/CodeSamples.java
// JetS3t code samples (in HTML): http://jets3t.s3.amazonaws.com/toolkit/code-samples.html#downloading
// JetS3t JavaDocs: http://jets3t.s3.amazonaws.com/api/org/jets3t/service/model/StorageObject.html
public class S3Api {
    private static final String BUCKET_NAME = "test-bucket-Olive";
    private 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 RestS3Service getS3Service() throws S3ServiceException {
        AWSCredentials awsCredentials = OliveDatabaseApi.getAwsCredentials();

        // RestS3Service is similar to HttpClient
        RestS3Service s3Service = new RestS3Service(awsCredentials);
        return s3Service;
    }

    // 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());
    }

    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 videoUrl = AWS_URL_PREFIX + fileNameOnS3;
        return videoUrl;
    } 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();
    }
}

public static void deleteFileInS3(String fileName) {
    log.severe("deleteFileInS3 has not yet been implemented.");
    // Do something like this:
    // RestS3Service s3Service = getS3Service();
    // s3Service.deleteObject(BUCKET_NAME, objectKey); // TODO Store objectKey in the da
tabase
}

public static String getNameFromUrl(String videoUrl) {
    return videoUrl.substring(AWS_URL_PREFIX.length());
}

public static String getNameFromUrlWithNewTimeStamp(String videoUrl) {
    return S3Api.getTime()
        + getNameFromUrl(videoUrl).substring(S3Api.getTime().length()); // TODO Fix
fugly code
}

public static String getVideoInformation(int projectId) {
}
```

```
int[] videoIds = OliveDatabaseApi.getVideoIds(projectId);
Video[] videos = new Video[videoIds.length];

for (int videoIndex = 0; videoIndex < videoIds.length; ++videoIndex) {
    String videoName = OliveDatabaseApi
        .getVideoName(videoIds[videoIndex]);
    String videoUrl = OliveDatabaseApi
        .getVideoUrl(videoIds[videoIndex]);
    String videoIcon = OliveDatabaseApi
        .getVideoIcon(videoIds[videoIndex]);
    int startTimeStoryboard = OliveDatabaseApi
        .getVideoStartTimeStoryboard(videoIds[videoIndex]);

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

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



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

return output;
}
}
```



```
package com.readytalk.olive.model;

public class Project {

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

    public Project(String name, int accountId, String icon) {
        this.name = name;
        this.accountId = accountId;
        this.icon = icon;
    }

    @Override
    public String toString() {
        return name;
    }

    public String getName() {
        return name;
    }

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

    public int getAccountId() {
        return accountId;
    }

    public void setAccountId(int accountId) {
        this.accountId = accountId;
    }

    public String getIcon() {
        return icon;
    }

    public void setIcon(String icon) {
        this.icon = icon;
    }
}
```

```
package com.readytalk.olive.model;
```

```
public class User {
```

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

```
    public User(String username, String password, String name, String email) {  
        this.username = username;  
        this.password = password;  
        this.name = name;  
        this.email = email;  
    }
```

```
    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);  
    }
```

```
    @Override  
    public String toString() {  
        return username;  
    }
```

```
    public String getUsername() {  
        return username;  
    }
```

```
    public void setUsername(String username) {  
        this.username = username;  
    }
```

```
    public String getPassword() {  
        return password;  
    }
```

```
    public void setPassword(String password) {  
        this.password = password;  
    }
```

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

```
    public String getName() {  
        return name;  
    }
```

```
    public void setEmail(String email) {  
        this.email = email;  
    }
```

```
    public String getEmail() {
```

```
        return email;  
    }
```

```
    public void setSecurityQuestion(String securityQuestion) {  
        this.securityQuestion = securityQuestion;  
    }
```

```
    public String getSecurityQuestion() {  
        return securityQuestion;  
    }
```

```
    public void setSecurityAnswer(String securityAnswer) {  
        this.securityAnswer = securityAnswer;  
    }
```

```
    public String getSecurityAnswer() {  
        return securityAnswer;  
    }
```

```
}
```

```
package com.readytalk.olive.model;

public class Video {

    private String name;
    private String url;
    private String icon;
    private int projectId;
    private int startTimeStoryboard;

    public Video(String name, String url, String icon, int projectId,
        int startTimeStoryboard) {
        this.name = name;
        this.url = url;
        this.icon = icon;
        this.projectId = projectId;
        this.setStartTimeStoryboard(startTimeStoryboard);
    }

    public String getName() {
        return name;
    }

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

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    public String getIcon() {
        return icon;
    }

    public void setIcon(String icon) {
        this.icon = icon;
    }

    public int getProjectId() {
        return projectId;
    }

    public void setProjectId(int project) {
        this.projectId = project;
    }

    public void setStartTimeStoryboard(int startTimeStoryboard) {
        this.startTimeStoryboard = startTimeStoryboard;
    }

    public int getStartTimeStoryboard() {
        return startTimeStoryboard;
    }

}
```

## src/com/readytalk/olive/servlet/OliveServlet.java

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

import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.util.Iterator;
import java.util.List;
import java.util.logging.Logger;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
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 com.google.gson.Gson;
import com.readytalk.olive.json.DeleteProjectRequest;
import com.readytalk.olive.json.DeleteVideoRequest;
import com.readytalk.olive.json.GeneralRequest;
import com.readytalk.olive.json.SplitVideoRequest;
import com.readytalk.olive.logic.HttpSenderReceiver;
import com.readytalk.olive.logic.OliveDatabaseApi;
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;

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 this.
    public static File destinationDir; // TODO Make a getter for this.

    // Modified from: http://www.jsptube.com/servlet-tutorials/servlet-file-upload-example.html
    // Also see: http://stackoverflow.com/questions/4101960/storing-image-using-htm-input-type-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 getProjectIdFromSessionAttributes(HttpSession session) {
        String sessionUsername = (String) session.getAttribute(Attribute.USERNAME.toString());
        int accountId = OliveDatabaseApi.getAccountId(sessionUsername);
        String sessionProjectName = (String) session.getAttribute(Attribute.PROJECT_NAME.toString());
        int projectId = OliveDatabaseApi.getProjectId(sessionProjectName, accountId);
        return projectId;
    }

    @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-form-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")) {
                handleSecurityQuestion(request, response, session);
            } else if (id.equals("new_password")) {
                handleNewPassword(request, response, session);
            } else if (id.equals("DeleteAccount")) {
                handleDeleteAccount(request, response, session);
            } 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");
            handleJsonPostRequest(request, response, session);
        }
    }
}
```

```
} else {
    log.severe("Unknown content type");
}

private void handleDeleteAccount(HttpServletRequest request,
    HttpServletResponse response, HttpSession session)
    throws UnsupportedEncodingException, IOException {
    String username = (String) session.getAttribute(Attribute.USERNAME
        .toString());
    int accountId = OliveDatabaseApi.getAccountId(username);
    OliveDatabaseApi.deleteAccount(accountId);
    response.sendRedirect("logout.jsp");
}

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 = OliveDatabaseApi.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 handleSecurityQuestion(HttpServletRequest request,
    HttpServletResponse response, HttpSession session)
    throws UnsupportedEncodingException, IOException {
    // TODO Auto-generated method stub
    String username = request.getParameter("username");
    String securityQuestion = request.getParameter("security_question");
    String securityAnswer = request.getParameter("security_answer");
    Boolean isCorrect;
    if (Security.isSafeUsername(username)
        && Security.isSafeSecurityQuestion(securityQuestion)
        && Security.isSafeSecurityAnswer(securityAnswer)) {
        session.setAttribute(Attribute.IS_SAFE.toString(), true);
        isCorrect = OliveDatabaseApi.isCorrectSecurityInfo(username,
            securityQuestion, securityAnswer);
        session.setAttribute(Attribute.IS_CORRECT.toString(), isCorrect);
        if (isCorrect) {
```

```
        session.setAttribute(Attribute.USERNAME.toString(), username);
        response.sendRedirect("new-password-form.jsp");
        session.removeAttribute(Attribute.IS_SAFE.toString()); // Cleared so as to n
ot interfere with any other form.
    } else {
        response.sendRedirect("forgot.jsp");
    }
} else {
    session.setAttribute(Attribute.IS_SAFE.toString(), false);
    session.setAttribute(Attribute.IS_CORRECT.toString(), false);
    response.sendRedirect("forgot.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 = OliveDatabaseApi.getAccountId((String) session
        .getAttribute(Attribute.USERNAME.toString()));
    if (projectName != null && Security.isSafeProjectName(projectName)
        && OliveDatabaseApi.projectExists(projectName, accountId)) { // Short-circui
ting
        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 = OliveDatabaseApi.isAuthorized(username, password);
        session.setAttribute(Attribute.IS_AUTHORIZED.toString(),
            isAuthorized);
        if (isAuthorized) { // Take the user to the projects page.
            int accountId = OliveDatabaseApi.getAccountId(username);
            session.setAttribute(Attribute.USERNAME.toString(),
                OliveDatabaseApi.getAccountUsername(accountId));
            session.setAttribute(Attribute.PASSWORD.toString(), password);
            session.setAttribute(Attribute.EMAIL.toString(),
                OliveDatabaseApi.getAccountEmail(accountId));
            session.setAttribute(Attribute.NAME.toString(),
                OliveDatabaseApi.getAccountName(accountId));
            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("security_question");
    String securityAnswer = request.getParameter("security_answer");
    log.info("Security question: " + securityQuestion
        + ". Security Answer: " + securityAnswer);
    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 = OliveDatabaseApi
                .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 = OliveDatabaseApi.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);
    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 {
    String projectName = request.getParameter("ProjectName");
    if (Security.isSafeProjectName(projectName)) {
        session.setAttribute(Attribute.IS_SAFE.toString(), true);

        String sessionUsername = (String) session
            .getAttribute(Attribute.USERNAME.toString());
        int accountId = OliveDatabaseApi.getAccountId(sessionUsername);
        String icon = ""; // TODO Get this from user input.
        Project project = new Project(projectName, accountId, icon);
        Boolean added = OliveDatabaseApi.AddProject(project);
        if (!added) {
            session.setAttribute(Attribute.ADD_SUCCESSFULLY.toString(),
                false);
        } else {
            session.setAttribute(Attribute.ADD_SUCCESSFULLY.toString(),
                true);
        }
    } else {
        session.setAttribute(Attribute.IS_SAFE.toString(), false);
    }
    response.sendRedirect("new-project-form.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("VideoName")) { // Short-circuitry
        // 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)) {
    String videoUrl = S3Api.uploadFile(file);
    if (videoUrl != null) {
        OliveDatabaseApi.AddVideo(videoName, videoUrl, projectId,
            "/olive/images/bbb480.jpg"); // TODO Get icon from Zencoder.
        // 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();
    } 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 type is invalid.");
        log.warning("Upload Failed. Video type is invalid.");
        //response.sendError(HttpServletResponse.SC_UNSUPPORTED_MEDIA_TYPE);
        return;
    } else {
        out.println("Upload Failed. Video name is invalid.");
        log.warning("Upload Failed. Video name is invalid.");
        //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/convertng-json-to-java/1688182#1688182
private void handleJsonPostRequest(HttpServletRequest request,
    HttpServletResponse response, HttpSession session)
    throws IOException {
    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("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("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("getVideoInformation")) {
        getVideoInformation(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 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 = OliveDatabaseApi.getAccountId(sessionUsername);
    String projectToDelete = deleteProjectRequest.arguments.project;
    int projectId = OliveDatabaseApi.getProjectId(projectToDelete,
        accountId);
    OliveDatabaseApi.deleteProject(projectId);

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

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

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 = OliveDatabaseApi.getVideoId(
        deleteVideoRequest.arguments.video, projectId);
    OliveDatabaseApi.deleteVideo(videoId);

    S3Api.deleteFileInS3(OliveDatabaseApi.getVideoName(videoId));
    S3Api.deleteFileInS3(OliveDatabaseApi.getVideoName(videoId));

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

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

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 {
```



```
        log.severe("handleAddToSelected has not yet been implemented.");
    }

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

    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.");
            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.");
            log.warning("Split time (in seconds) is invalid.");
            response.sendError(HttpServletResponse.SC_BAD_REQUEST);
            return;
        }

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

        for (Video videoFragment : videoFragments) { // foreach-loop
            OliveDatabaseApi.AddVideo(videoFragment.getName(),
                videoFragment.getUrl(), projectId, videoFragment.getIcon()); // projectI
d 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 {
        log.severe("handleCombineVideos has not yet been implemented.");
    }

    private void getVideoInformation(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();
    }
}
```

02/26/11  
01:45:15

Olive

1

src/com/readytalk/olive/util/Attribute.java

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

public enum Attribute {
    IS_AUTHORIZED, USERNAME, PASSWORD, EDIT_SUCCESSFULLY, ADD_SUCCESSFULLY, PROJECT_NAME, PA
    SSWORDS_MATCH, IS_SAFE, EMAIL, NAME, SECURITY_QUESTION, SECURITY_ANSWER, IS_CORRECT
}
```

```
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 = 1699131254533294330L;

    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: 0px;
    right: 70%;
    bottom: 160px;
    left: 0px;
    text-align: center;
    /* 100% - (video width) - (2 x (sum of padding, border, margin)) */
}

#videos-title {
    position: absolute;
    top: 0%;
    right: 0%;
    bottom: 10%;
    left: 0%;
    margin: 5px;
}

#videos-controls {
    position: absolute;
    top: 10%;
    right: 0%;
    bottom: 20%;
    left: 0%;
    border-top: 1px solid gray;
    border-right: 1px solid gray;
    border-bottom: 0px solid gray; /* different */
    border-left: 1px solid gray;
}

#videos {
    position: absolute;
    top: 20%;
    right: 0%;
    bottom: 0%;
    left: 0%;
    text-align: left;
    border: 1px solid gray;
    overflow: auto;
}

.video-container {
    display: block;
    float: left;
    width: 100px;
    margin: 10px;
    padding: 10px;
    border: 1px solid green;
    background-color: #FFFFFFE;
    text-align: center;
}

.video-icon {
    display: block;
    border: 1px solid gray;
    width: 50px;
    height: 50px;

```

```
    margin-left: auto;
    margin-right: auto;
    padding: 5px;
}

#player-div {
    position: absolute;
    top: 0px;
    right: 0px;
    bottom: 160px;
    left: 30%;
    /* A bit smaller than #videos-container's right edge (compensate for border, etc. */
    border: 1px solid gray;
}

#player-container {
    position: absolute;
    top: 0%;
    right: 0%;
    bottom: 10%;
    left: 0%;
}

#player-controls {
    position: absolute;
    bottom: 0%;
    left: 0%;
    right: 0%;
    height: 10%;
}

video {
    float: right;
    width: 100%;
    height: 100%;
}

#timeline {
    position: absolute;
    right: 200px;
    bottom: 0px;
    left: 0px;
    height: 160px;
    border: 1px solid gray;
    overflow-x: auto;
}

#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: 50%;
    padding: 10% 10% 10% 10%;
}

#login-form {
}
```

```
@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: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;
}

/*
-----
Common elements to every page
-----
*/
#header {
position: absolute;
```

```
    top: 0px;
    right: 0px;
    left: 0px;
    height: 50px;
    border-bottom: 2px solid #888888;
    min-width: 1000px;
}

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

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

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

#footer {
    position: absolute;
    right: 0px;
    bottom: 00px;
    left: 0px;
    height: 50px;
    border-top: 2px 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: 15%;
    left: 0%;
}

#projects-controls {
    position: absolute;
    top: 15%;
    right: 0%;
    bottom: 20%;
    left: 0%;
}

#project-clips {
    position: absolute;
    top: 20%;
    right: 0%;
    bottom: 0%;
    left: 0%;
    overflow: auto;
}

.project-icon-container {
    float: left;
    width: 100px;
    padding-bottom: 40px;
    padding-right: 40px;
}

#project-clips div img {
    display: block;
    padding: 5px;
    border: 1px solid #000000;
    width: 100px;
    height: 100px;
    border: 1px solid #000000;
}
```

```
@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,del,
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;
}
```

02/26/11  
01:45:15

Olive

1

## WebContent/META-INF/context.xml

```
<?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>
```

02/26/11  
01:45:15

Olive

WebContent/temp/maintainer.txt

1

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

02/26/11  
01:45:15

Olive

WebContent/scripts/account.js

1

```
/*
 * This is Olive's JavaScript file for account.js 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($) {

});
```

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

var deleteVideoDialogContext; // TODO Remove this global variable.
var video; // Global

// 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($) {
    attachDeleteVideoHandlers();
    attachVideoMenuHandlers();
    attachVideoClickHandlers();
    attachPlayerHandlers();
    enableDragAndDrop();
    attachContextMenuHandlers();
    getVideoInformation();
});

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

    $('#confirm-delete-video-dialog').dialog({
        autoOpen: false,
        resizable: false,
        height: 275,
        modal: true,
        buttons: {
            'Delete': function () {
                deleteVideo.call(deleteVideoDialogContext); // 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');
            }
        }
    });
}

function attachVideoMenuHandlers() {
    $('#upload-new-button').click(function () {
        openNewVideoForm();
    });

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

function attachVideoClickHandlers() {
    $('#video-container').click(function () {
        if ($(this).data('isSelected')) {
            unselect(this);
        } else {
            // First, unselect all
            $('#video-container').each(function () {
                unselect(this); // 'this' is a different 'this' than outside .each()
            });
        }
    });
}
```

```
});
// Then, select this
select(this);
});
}

function select(element) {
    $(element).data('isSelected', true);
    $(element).css( {
        'background-color': '#edf4e6' // A lighter version of the Olive color
    });
    addToSelected($(element).attr('id'));
    updatePlayerWithNewElement(element);
}

function unselect(element) {
    $(element).data('isSelected', false);
    $(element).css( {
        'background-color': ''
    });
    removeFromSelected($(element).attr('id'));
    updatePlayerWithNoElements();
}

//Perform an addToSelected request
function addToSelected(id) {
    var videoName = id; // TODO This works by definition (but the definition should probably
    // change).
    var requestData = '{'
        + '"command" : "addToSelected",'
        + '"arguments" : {'
        + '"video" : "' + videoName + '"'
        + '}'
        + '}'
    makeAjaxPostRequest(requestData, null, null); // Defined in "/olive/scripts/master.js"
}

// Perform a removeFromSelected request
function removeFromSelected(id) {
    var videoName = id; // TODO This works by definition (but the definition should probably
    // change).
    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) {
    $('#player-video').attr('type', 'video/mp4'); // TODO Get this from the database.
    $('#player-video').attr('poster', $(element).data('icon'));
    $('#player-video').attr('src', $(element).data('url'));
}

function updatePlayerWithNoElements() {
    $('#player-video').removeAttr('type');
```

```

    $('#player-video').removeAttr('poster');
    $('#player-video').removeAttr('src');
}

function attachPlayerHandlers() {
    video = document.getElementById('player-video');

    $('#videos-playpause').click(function () {
        if (video.paused) {
            video.play();
        } else {
            video.pause();
        }
    });

    $('#videos-volume-up').click(function () {
        if (video.volume < 0.85) { // Account for rounding errors
            video.volume += 0.1;
        } else {
            video.volume = 1.0; // Don't allow rounding errors
            $('#videos-volume-up').attr('disabled', 'disabled'); // Disable
        }
        $('#videos-volume-down').removeAttr('disabled'); // Enable
    });

    $('#videos-volume-down').click(function () {
        if (video.volume > 0.15) { // Account for rounding errors
            video.volume -= 0.1;
        } else {
            video.volume = 0.0; // Don't allow rounding errors
            $('#videos-volume-down').attr('disabled', 'disabled'); // Disable
        }
        $('#videos-volume-up').removeAttr('disabled'); // Enable
    });
}

function enableDragAndDrop() {
    // Modified from: http://jqueryui.com/demos/draggable/
    $('#video-container').draggable({
        appendTo: 'body',
        scroll: false,
        connectToSortable: '#timeline',
        helper: 'clone',
        revert: 'invalid',
        snap: '#timeline'
    });

    $('#timeline').sortable({
        revert: true,
        sort: function() {
            if($('#timeline').sortable('items').length > 0){
                $('#export-button').removeAttr('disabled');
            } else {
                $('#export-button').attr('disabled', 'disabled');
            }
        }
    });
}

function attachContextMenuHandlers() {
    $('#video-container').contextMenu('video-context-menu', {
        menuStyle: {
            border: '1px solid #000'

```

```

    },
    itemStyle: {
        fontFamily: 'Arial',
        backgroundColor: '#fff',
        color: 'black',
        border: 'none',
        padding: '1px'
    },
    itemHoverStyle: {
        color: '#fff',
        backgroundColor: '#00c',
        border: 'none'
    },
    bindings: {
        'split-video-menu-item': function(t) {
            $('#split-video-dialog-form').dialog('open');
        }
    });
}

// Perform a deleteVideo request
function deleteVideo() {
    var requestData = '{'
        + '"command": "deleteVideo",'
        + '"arguments": {'
        + '"video": "' + $(this).attr('id') + '" '
        + '},'
        + '}'
        + ' ';

    makeAjaxPostRequest(requestData, function(responseData) {location.reload();}, null);
    // Defined in "/olive/scripts/master.js".
}

//Perform a splitVideo request
function splitVideo(videoName, splitTimeInSeconds) {
    var requestData = '{'
        + '"command": "splitVideo",'
        + '"arguments": {'
        + '"video": "' + videoName + '" '
        + '"splitTimeInSeconds": ' + splitTimeInSeconds + ' '
        + '},'
        + '}'
        + ' ';

    makeAjaxPostRequest(requestData, function(responseData) {location.reload();}, null);
    // Defined in "/olive/scripts/master.js".
}

function attachSplitHandlers() {
    var videoName = $('#video-name'), splitTimeInSeconds = $('#split-time-in-seconds'), allF
    ields = $(

        []).add(videoName).add(splitTimeInSeconds), tips = $('#.validateTips');

    function updateTips(t) {
        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 checkCondition(o, condition, n) {
    if (!condition) {
        o.addClass("ui-state-error");
        updateTips(n);
        return false;
    } else {
        return true;
    }
}

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

            bValid = bValid
                && checkLength(videoName,
                    'video-name', 1, 32);

            bValid = bValid
                && checkLength(splitTimeInSeconds, 'split-time-in-seconds',
                    0, 14400);

            bValid = bValid
                && checkRegexp(videoName,
                    /^[0-9a-zA-Z]+$/i,
                    'Video name may consist of a-z, 0-9; and must begin with a l
etter.');
```

```

        window.location.reload(true);
        $(this).dialog('close');
    }
},
close : function() {
    allFields.val('').removeClass('ui-state-error');
}
});

function getVideoInformation() {
    var requestData = '{'
        + '"command" : "getVideoInformation"'
        + '}'
    makeAjaxPostRequest(requestData, function (responseData) {
        for (var i = 0; i < responseData.length; ++i) {
            $('#'+ responseData[i].name).data('url', responseData[i].url);
            $('#'+ responseData[i].name).data('icon', responseData[i].icon);
            $('#'+ responseData[i].name).data('startTimeStoryboard', responseData[i].startT
imeStoryboard);
        }
    }, null);    // Defined in "/olive/scripts/master.js".
}

function openNewVideoForm() {
    window.open("new-video-form.jsp", "videoUploadForm",
        "menubar=no,width=320,height=200,toolbar=no");
}

```





```
/*
 * 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
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
    });

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

function openHelpWindow() {
    window.open("help.jsp", "HelpWindow",
        "menubar=no,width=500,height=500,toolbar=no");
}

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,
        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);
            }
        }
    })
}
```

```
    });
}
```

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

var deleteProjectDialogContext; // 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($) {
    attachDeleteProjectHandlers();
});

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

    $('#confirm-delete-project-dialog').dialog({
        autoOpen: false,
        resizable: false,
        height: 275,
        modal: true,
        buttons: {
            'Delete': function () {
                deleteProject.call(deleteProjectDialogContext); // 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 deleteProject request
function deleteProject() {
    var requestData = '{'
        + '"command" : "deleteProject",'
        + '"arguments" : {'
        + '"project" : "' + $(this).attr('id') + '"',
        + '}'
        + '>';

    makeAjaxPostRequest(requestData, function (responseData) {location.reload(); }, null);
} // Defined in "/olive/scripts/master.js".

function openNewProjectForm() {
    window.open("new-project-form.jsp", "newProjectForm",
        "menubar=no,width=320,height=200,toolbar=no");
}
```

```
<?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/web-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>
```

## WebContent/account.jsp

```
<%@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());
    String name = (String) session.getAttribute(Attribute.NAME
        .toString());
    String email = (String) session.getAttribute(Attribute.EMAIL
        .toString());
    String password = (String) session.getAttribute(Attribute.PASSWORD
        .toString());
    String securityQuestion = (String) session
        .getAttribute(Attribute.SECURITY_QUESTION.toString());
```

```
        String securityAnswer = (String) session
            .getAttribute(Attribute.SECURITY_ANSWER.toString());
    %>
    <div id="header">
    <div id="header-left">
    <h1>Olive</h1>
    </div>
    <!-- end #header-left -->
    <div id="header-right">
    <div>Welcome, <%=username%>!&nbsp;<a href="logout.jsp">Logout</a></div>
    <div><strong><a href="projects.jsp">My Projects</a></strong>&nbsp;<span
        id="help-dialog-opener"><a href="">Help</a></span></div>
    <div id="help-dialog" 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>
    <!-- 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><br />
    <input type="text" name="new-name" id="new-name" value="<%=name%>"
        size="32" maxlength="32" /></p>
    <p><label for="new-email">Email</label><br />
    <input type="text" name="new-email" id="new-email" value="<%=email%>"
        size="32" maxlength="64" /></p>
    <p><label for="new-password">Password</label><br />
    <input type="password" name="new-password" id="new-password"
        value="<%=password%>" size="32" maxlength="128" /></p>
    <p><label for="confirm-new-password">Confirm password</label><br />
    <input type="password" name="confirm-new-password"
        id="confirm-new-password" value="<%=password%>" size="32"
        maxlength="128" /></p>
    <p><label for="security_question">Security Question</label><br />
    <input type="text" name="security_question" id="security_question"
        value="<%=securityQuestion%>" size="32" maxlength="128" /></p>
    <p><label for="security_answer">Security Answer</label><br />
    <input type="text" name="security_answer" id="security_question_answer"
        value="<%=securityAnswer%>" size="32" maxlength="128" /></p>
    <input type="hidden" name="FormName" value="EditUser"></input><br />
    <input type="submit" value="Update information" /><span><%=editConfirmation%></span></form>
    <form id="delete-account" action="OliveServlet" name="process"
        method="post"><input type="hidden" name="FormName"
        value="DeleteAccount"></input><br />
    <input type="submit" value="Delete Account" /> <!-- TODO: Add a "Are you sure you want to de
    lete your account window?" here -->
    </form>
    </div>
    <!-- end #login-form-container --></div>
    <!-- end #main -->
```

02/26/11  
01:45:15

Olive

WebContent/account.jsp

2

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

```
<div id="footer"></div>  
</body>  
</html>
```

## WebContent/editor.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="com.readytalk.olive.logic.OliveDatabaseApi"%>
<%@ 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/master.js"></script>
<script src="/olive/scripts/editor.js"></script>
<script src="/olive/scripts/google-analytics.js"></script>
<script src="/olive/scripts/contextMenu.js"></script>
</head>
<body>
<%
    String username = "";
    Boolean isAuthorized = (Boolean) session
        .getAttribute(Attribute.IS_AUTHORIZED.toString()); // Nasty cast
    String projectName = "";
    String videosHtml = "";
    if (isAuthorized == null) {
        response.sendRedirect("index.jsp");
    } else if (!isAuthorized) {
        response.sendRedirect("index.jsp");
    } else {
        username = (String) session.getAttribute(Attribute.USERNAME
            .toString());
        projectName = (String) session
            .getAttribute(Attribute.PROJECT_NAME.toString());
        if (projectName == null) {
            response.sendRedirect("projects.jsp");
        }

        int accountId = OliveDatabaseApi.getAccountId(username);
        int projectId = OliveDatabaseApi.getProjectId(projectName,
            accountId);
        videosHtml = OliveDatabaseApi.populateVideos(projectId);
    }
%>
<div id="header">
<div id="header-left">
<h1>Olive</h1>
</div>
<!-- end #header-left -->
<div id="header-right">
<div>Welcome, <a href="account.jsp"><%=username%>!</a>&nbsp;<a
    href="logout.jsp">Logout</a></div>
<div><strong><a href="projects.jsp">My Projects</a></strong>&nbsp;<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-title">
<h3><%=projectName%></h3>
</div>
<!-- end #videos-title -->
<div id="videos-controls"><!-- http://stackoverflow.com/questions/1106720/how-to-display-htm
l-form-as-inline-element/1106747#1106747 -->
<button id="upload-new-button" type="button">Upload New Video</button>
<button id="split-button" type="button">Split Video</button>
</div>
<!-- end #videos-controls -->
<div id="videos"><%=videosHtml%></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-div">
<div id="player-container"><video id="player-video"
    preload="preload"></video></div>
<div id="player-controls" class="center-text">
<button id="videos-playpause">Play/pause</button>
<button id="videos-volume-down">Volume down</button>
<button id="videos-volume-up" disabled="disabled">Volume up</button>
</div>
</div>
<!-- end #player -->

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

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

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

<div id="export">
<button id="export-button" type="button" disabled="disabled"
    onclick="alert('Export to Computer');">Export to Computer</button>

</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" 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" title="Warning!">
<p>Delete video?</p>
</div>
<!-- type="number", min, and max are valid in HTML5: http://dev.w3.org/html5/markup/input.number.html -->
<div id="split-video-dialog-form" title="Split video">
<p class="validateTips">All form fields are required.</p>
<form id="split-video-form" action="OliveServlet" name="process"
  method="post">
<fieldset><label for="video-name">Name of video to
split</label> <input type="text" name="video-name" id="video-name"
  class="text ui-widget-content ui-corner-all" maxlength="32" /> <label
  for="split-time-in-seconds">Split time (in seconds)</label> <input
  type="number" min=0 max=14400 name="split-time-in-seconds"
  id="split-time-in-seconds" value=""
  class="text ui-widget-content ui-corner-all" /></fieldset>
<input type="hidden" name="FormName" value="SplitVideo"></input></form>
</div>
<!-- end #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>  
<div id="footer-right">&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 editConfirmation = "";
    if(isCorrect == null){
        editConfirmation = "";
    }
    else if(isCorrect == false){
        if(isSafe == null){
            editConfirmation = "";
        }
        else if(isSafe){
            editConfirmation = "Question or answer do not match database records";
        }
        else{
            editConfirmation = "Unsafe input";
        }
    }
    else if(isCorrect){
        editConfirmation = "";
        response.sendRedirect("new-password-form.jsp");
    }
    session.removeAttribute(Attribute.IS_SAFE.toString());
    session.removeAttribute(Attribute.IS_CORRECT.toString());
%>
<div id="header">
<div id="header-left">
<h1>Olive</h1>
</div>
<!-- end #header-left -->
<div id="header-right"><span
    id="help-dialog-opener"><a href="">Help</a></span></div>
<div id="help-dialog" 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>
<!-- 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 and your security question and answer. 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>
    <p><label for="security_question">Security Question</label><br />
<input type="text" name="security_question"
    id="security_question" value="" size="32"
    maxlength="128" /></p>
    <p><label for="security_answer">Security Answer</label><br />
<input type="text" name="security_answer"
    id="security_question_answer" value="" size="32"
    maxlength="128" /></p>
<input type="hidden" name="FormName" value="security-question-form"></input><br />
<input type="submit" value="Recover Password" /><span><%=editConfirmation%></span>
</form>
</div>
<!-- end #main --></div>

<div id="footer"></div>
</body>
</html>
```

02/26/11  
01:45:15

Olive

WebContent/header.jsp

1

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

## WebContent/index.jsp

```
%@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" />

<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/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">
<h1>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="splash-container"></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 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_pas
sword"
    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 --></div>
<!-- end #main -->

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

<div id="help-dialog" 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.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-iterate-over-e
num-in-java
    for (Attribute attribute : Attribute.values()) {
        session.removeAttribute(attribute.toString());
    }
    response.sendRedirect("index.jsp");
%>
</body>
</html>
```

## WebContent/new-password-form.jsp

```
<%@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.toStrin
g());
    Boolean editSuccess = (Boolean) session.getAttribute(Attribute.EDIT_SUCCESSFULLY.toStrin
g());
    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_SAFE.toString());
    session.removeAttribute(Attribute.PASSWORDS_MATCH.toString());
    session.removeAttribute(Attribute.EDIT_SUCCESSFULLY.toString());
%>
<div id="header">
<div id="header-left">
<h1>Olive</h1>
</div>
```

```
<!-- end #header-left -->
<div id="header-right"><span id="help-dialog-opener"><a
    href="">Help</a></span></div>
<div id="help-dialog" 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>
<!-- 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>
</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>Create new project | Olive</title>
<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 isSafe = (Boolean) session.getAttribute(Attribute.IS_SAFE
        .toString()); // Nasty cast
    Boolean addedSuccessfully = (Boolean) session
        .getAttribute(Attribute.ADD_SUCCESSFULLY.toString());
    String safeMessage;
    if (isSafe == null) {
        safeMessage = "";
    } else if (isSafe) {
        if(addedSuccessfully){
            // Syntax: http://www.infimum.dk/HTML/JSwindows.html
            safeMessage = "<script> window.opener.location.reload(); window.close(); </scrip
t>";
        }
        else{
            safeMessage = "Project already exists";
        }
    } else {
        safeMessage = "Project name must be between 1 and 32 characters; must consist of a-z
, 0-9, underscores; and must begin with a letter.";
    }
    session.removeAttribute(Attribute.ADD_SUCCESSFULLY.toString());
    session.removeAttribute(Attribute.IS_SAFE.toString());
%>
<form id="add-project-form" action="OliveServlet" name="process"
    method="post">
<p><label for="ProjectName">Project Name</label> <input type="text"
    name="ProjectName" id="ProjectName" value="" size="32" maxlength="32" />
</p>
<p><%=safeMessage%></p>
<input type="hidden" name="FormName" value="AddProject"></input> <input
    type="submit" value="Create Project" /></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>Upload new video | Olive</title>
<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");
    }
%>
<form id="UploadForm" action="OliveServlet" name="process"
    enctype="multipart/form-data" method="post"><input type="hidden"
    name="FormName" value="UploadVideo"></input> <input type="file"
    name="file" />
<p><label for="VideoName">Video Name</label> <input type="text"
    name="VideoName" id="VideoName" value="" size="32" maxlength="32" /></p>
<input type="submit" /></form>
</body>
</html>
```

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="com.readytalk.olive.logic.OliveDatabaseApi"%>
<%@ 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/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 = "";
    String projectsHtml = "";
    if (isAuthorized == null) {
        response.sendRedirect("index.jsp");
    } else if (!isAuthorized) {
        response.sendRedirect("index.jsp");
    } else {
        username = (String) session.getAttribute(Attribute.USERNAME
            .toString());
        int accountId = OliveDatabaseApi.getAccountId(username);
        projectsHtml = OliveDatabaseApi.populateProjects(accountId);
    }
%>
<div id="header">
<div id="header-left">
<h1>Olive</h1>
</div>
<div id="header-right">
<div>Welcome, <a href="account.jsp"><%=username%>!</a>&nbsp;<a
    href="logout.jsp">Logout</a></div>
<div><strong><a href="projects.jsp">My Projects</a></strong>&nbsp;<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 type="button" onclick="openNewProjectForm()">Create New
Project</button>
</div>
<!-- end #controls -->

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

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

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

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

</body>
</html>
```