VISAP-Dataaccesslayer modification

Vision and high-level flow

Table of Contents

[Revision History 2](#_Toc412535027)

[1 Introduction 3](#_Toc412535028)

[1.1 Objective 3](#_Toc412535029)

[1.2 Purpose 3](#_Toc412535030)

[1.3 Scope 3](#_Toc412535031)

[2 Executive Summary 3](#_Toc412535032)

[2.1 Key Decisions 3](#_Toc412535033)

[2.2 Core Services 3](#_Toc412535034)

[2.3 High level Workflow 3](#_Toc412535035)

[Visap Workflow 3](#_Toc412535036)

[3 Technical Design and Constraints 5](#_Toc412535037)

[3.1 Design Decisions and Technical Constraints 5](#_Toc412535038)

[4 Features and Services 5](#_Toc412535039)

[4.1 List of Services 5](#_Toc412535040)

[4.2 Flow Control 5](#_Toc412535041)

[5 Architecture 6](#_Toc412535042)

[5.1 Architecture Diagram 6](#_Toc412535043)

[5.2 Design of DataStruture for Timeline 9](#_Toc412535044)

[5.3 Design of DataStruture for Publish 12](#_Toc412535045)

[6 VISAP UI 17](#_Toc412535046)

[7 Risks and Next Steps 17](#_Toc412535047)

[7.1 Risks 17](#_Toc412535048)

[7.2 Next Steps 17](#_Toc412535049)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 1.0 | 03-Mar-2015 | Bharathi | Draft, after initial discussion with Prashant |

# Introduction

This document provides high-level details for modularizing the Dataaccesslayer hence by decentralizing the respective functionality.

This document outlines the Design prospective view of reviewed structure.

This section builds a clear picture on the objective purpose and scope of this document.

## Objective

The objective of his document is to provide

1. Design details of the Layer
2. Flow of the data from the view to database and back to view,Including the Middle managers.
3. Describe a Data structure of Enhanced Functionality.

## Purpose

The purpose of this document is to provide guidelines for Implementation and the better understanding of the flow for new developers.

## Scope

The scope of this document is -

1. To provide high-level Implementation details and flow.
2. To provide an overview how the flow is from services to Dataaccesslayer till the database.
3. To provide a complete list of managers used and data connection class and utility class.
4. To provide a workflow for consumption of dataaccesslayer.

# Executive Summary

## Key Decisions

* Timeline implementation will use video services and video collection.
* Service/handler will talk to manager from dataaccesslayer.
* Utility and dataconnection classes.

## Core Services

## High level Workflow

This section describes the high level workflow for visap from a technical point of view.

### Visap Workflow

1. A logged in user access a Visap product.
2. Video tab will have published videos.
3. Myspace where in logged in user videos will be loaded with the edit option.
4. Collaboration will have all the videos irrespective of users
5. Editing features applicable Under Edit tab
6. Timeline concept either will be tab or button which allows to create timeline.

# Technical Design and Constraints

This chapter focuses on the design constraints and how the Visap service will be used within the system.

## Design Decisions and Technical Constraints

# Features and Services

## List of Services

## Flow Control

Videohandler

BaseEntity

EditableBase

Video

Utilities/Formtating response

MongoDBconnection

# Architecture

This section describes a high level architecture of the proposed product.

## Architecture Diagram

The under given diagram describes the logical architecture of the DataAccessLayer

BaseEntity

Publish

Search

EditableBase

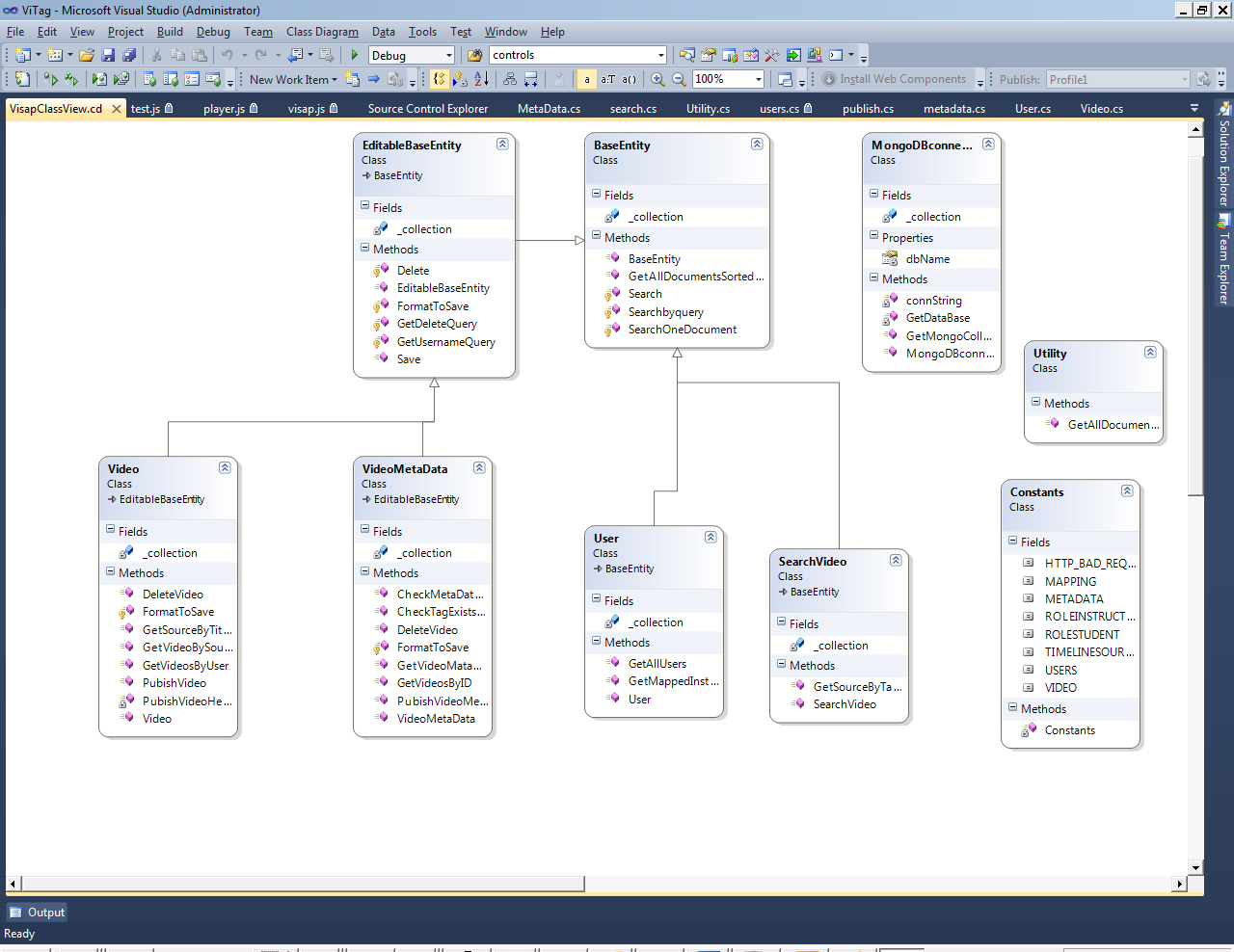
Video

VideoMetadata

MongoDBconnection

Utility

User



**Please find attached class diagram**

****

**Description:**

* Client sends the request to respective handler
* Handler will push the request to the subclass of the respective collection.
* MongoDBconnection will have the responsibility to connect to the database and get the respective collection asked by the manager.
* Baseentity will have get related methods
* EditableBaseEntity will have the methods or operations of the methods which are going to be edited.
* Classes called by handlers can use the relevant base for there operations.

For Instance

Client ajax call will hit the handler based on the config values

Video Handler will pass the collection name to Video class

Video class will hit relavent base classes get the common operations like save,Delete across the collection etc.

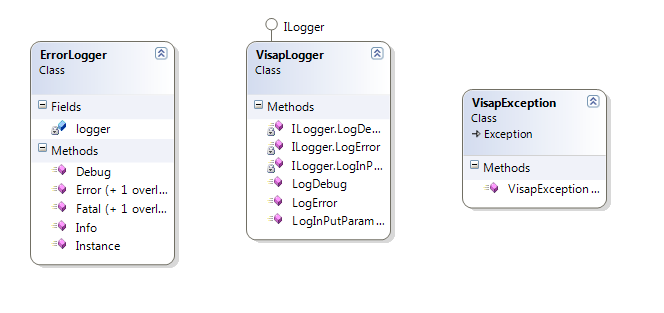
Save is a virtual method in base entity so that respective classes can override it if required.

Utility class will have common methods for formatting the bsondocument according to clients expectation

Utility class can be used from anywhere in the dataaccesslayer

MongoDBconnection class which will wrap all the methods related to mongodatabase like getting the collection and connecting to database.

**Exception and Logging**

****

**Description:**

* Ilogger will have declarations for logging sql and nosql as earlier it was meant for sql too- need to suggest for removal
* VisapLogger Implements Ilogger and can be used to loging at different level of information
* ErrorLogger is related to log4net and is wrapped with Visaplogger
* Errorlogger can itself be used and can be used wrapping on Visaplogger to be more specific
* Visapexception wrapper for exception and will be used to throw

exceptions.

* Further to handler and client exception – yet to implement

## Design of DataStruture for Timeline

Timeline can have one or many videos

VI

Timeline1

V2

V3

V4

One to many

Validation :

* TimelineName will be unique
* Will be listed only in myspace
* Sourcetype will be 2
* Tittle and desc will be given by User while saving
* GUID will be generated while creation of timeline

Based on Id Timeline is going to be accessed.

**Description:**

**Header Information:**

Table Name: **Videos**

Fields**:** username, tittle, desc, src, sourcetype

Json Format :To post the data Structure would be

{

"\_id" : "54c7381f93180dbb5e1835ef",

"src" : [{ "squence": 1, "data": { "timelineSource": "HTML5\_1.mp4", "starttime": 0, "duration": 5} },

{ "squence": 2, "data": { " timelineSource ": "Moving\_digits.mp4", " starttime ": 0, "duration”: 5 } }],

"username" : "Instructor2",

"tittle" : "Timeline1",

"desc" : "About the content",

"sourcetype" : 2,

"date" : ISODate("2015-01-27T07:02:54.798Z")

}

**Service for TimeLine used is video:**

EditableBaseEntity

Utilities/Formtating response

MongoDBconnection

BaseEntity

Video

**Services :**

Videos-Pointing to video collection

Metadata –Pointing for the operations on metadatacollection

Publish- will use video and metadata object

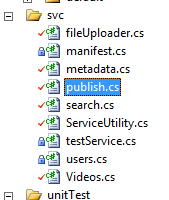
Users- used to get the list of users

Search-uses video and Metadata for loading videos according to search key

Fileuploader-To read configuration from config and accordingly upload video

Testservice- to post the junit report to menation drive

Schema looks like below:



**Description:**

1.GET(Existing call will do)

Instance1: when user clicks on myspace

All videos will be loaded including timeline videos

Timeline Videos will have Icon depicting for Timeline

loadData: function (autoplay, mode, user) {

}

2. onclick of save of timeline

Post call to post the timeline date

ns.SaveTimeLineVid = function () {

};

## Design of DataStruture for Publish

Publish video option can have one or more videos.

V1 V1

Collaboration

Myspace

V2 V2 V3

Publish

V3 V3

V4 V4

Validation :

* Initially My space will be empty.
* When user selects video from collaboration then clicks publish videos

then selected videos will be added to myspace.

* Both Header and detail information will flow while publishing videos

**Description:**

**Header Information:**

Table Name: **Videos**

Fields**:** username, tittle, desc, src, sourcetype

Json Format :To post the data Structure would be

{

"\_id" : "54c7381f93180dbb5e1835ef",

"src" :”nxt\_moondemo.mp4”,

"username" : "Instructor2",

"tittle" : "Timeline1",

"desc" : "About the content",

"sourcetype" : 1,

"date" : ISODate("2015-01-27T07:02:54.798Z")

}

{

"\_id" : "54c7381f93180dbb5e183ab",

RefID: “54c7381f93180dbb5e1835ef”,

"src" :”nxt\_moondemo.mp4”,

"username" : "Instructor2",

"tittle" : "Timeline1",

"desc" : "About the content",

"sourcetype" : 1,

"date" : ISODate("2015-01-27T07:02:54.798Z")

}

**Metadata Information:**

Table Name: **Metadata**

Json Format :To post the data Structure would be

{

"\_id" : "54c7381f93180dbb5e1835ef "

"username" : "stage",

"tags" : "[]",

"links" : "About the content",

“actions”:”[]”,

"sourcetype" : 1,//accordingly

"date" : ISODate("2015-01-27T07:02:54.798Z")

}

Json Format :To post the data Structure would be

{

"\_id" : "54c7381f93180dbb5e1835ef"

"username" : "Instructor2",

"tags" : "[]",

"links" : "About the content",

“actions”:”[]”,

"sourcetype" : 1,//accordingly

"date" : ISODate("2015-01-27T07:02:54.798Z")

}

Logic Used:

Publish:

* Check For refid for the first time publish RefId will not be there
* Create new GUID and existing Id will go as refid
* Remove exiting ID before saving
* Based on id and refid insert a record

Republish**:**

* Data will have RefID and latest generated ID is passed
* Add the data with based on sent generated ID
* RefId is not required Hence remove

Note: Publish Video class not needed as we are using video and metadata class for respective publishing.—suggestion expected

Utilities/Formtating response

MongoDBconnection

BaseEntity

PublishVideo

EditableBaseEntity

# VISAP UI

Timeline UI description can be given

# Risks and Next Steps

## Risks

## Next Steps