Zend_Media - Sven Vollbehr

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend_Media Component Proposal

Proposed Component Name	Zend_Media
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Media
Proposers	Sven Vollbehr Jon Whitcraft
Zend Liaison	TBD
Revision	1.0 - 3 May 2009: Initial Draft. (wiki revision: 5)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

1. Overview

Zend_Media is a collection of classes representing various media file formats. Classes structure information of the files according to the file format standards and provide means to manipulate the information in an object oriented fashion.

2. References

- PHP Reader Project

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** structure information in various media file formats.
- This component **will** provide means to read and write the header information of a supported media file.

4. Dependencies on Other Framework Components

- Zend_Exception
- Zend_Io

5. Theory of Operation

6. Milestones / Tasks

- Milestone 1: Working prototype transformed from existing code and checked into the incubator (necessary tasks: conform to Zend naming conventions, and refactor to support the new API described here and use the new Zend_Io library).
- Milestone 2: Unit tests exist, work, and are checked into SVN.
- Milestone 3: Initial documentation exists.
- Milestone 4: Moved to core.

7. Class Index

- Zend_Media_Exception

- Zend_Media_Asf
- Zend_Media_Asf_Exception
- Zend_Media_Asf_Object
- Zend_Media_Asf_Object_AdvancedContentEncryption
- Zend_Media_Asf_Object_AdvancedMutualExclusion
- Zend_Media_Asf_Object_BandwidthSharing
- Zend_Media_Asf_Object_BitrateMutualExclusion
- Zend_Media_Asf_Object_CodecList
- Zend_Media_Asf_Object_Compatibility
- Zend_Media_Asf_Object_Container
- Zend_Media_Asf_Object_ContentBranding
- Zend_Media_Asf_Object_ContentDescription
- Zend_Media_Asf_Object_ContentEncryption
- Zend_Media_Asf_Object_Data
- Zend_Media_Asf_Object_DigitalSignature
- Zend_Media_Asf_Object_ErrorCorrection
- Zend_Media_Asf_Object_ExtendedContentDescription
- Zend_Media_Asf_Object_ExtendedContentEncryption
- Zend_Media_Asf_Object_ExtendedStreamProperties
- Zend_Media_Asf_Object_FileProperties
- Zend_Media_Asf_Object_GroupMutualExclusion
- Zend_Media_Asf_Object_Header
- Zend_Media_Asf_Object_HeaderExtension
- Zend_Media_Asf_Object_Index
- Zend_Media_Asf_Object_IndexParameters
- Zend_Media_Asf_Object_LanguageList
- Zend_Media_Asf_Object_Marker
- Zend_Media_Asf_Object_MediaObjectIndex
- Zend_Media_Asf_Object_MediaObjectIndexParameters
- Zend_Media_Asf_Object_Metadata
- Zend_Media_Asf_Object_MetadataLibrary
- Zend_Media_Asf_Object_Padding
- Zend_Media_Asf_Object_ScriptCommand
- Zend_Media_Asf_Object_SimpleIndex
- Zend_Media_Asf_Object_StreamBitrateProperties
- Zend_Media_Asf_Object_StreamPrioritization
- Zend_Media_Asf_Object_StreamProperties
- Zend_Media_Asf_Object_TimecodeIndex
- Zend_Media_Asf_Object_TimecodeIndexParameters
- Zend_Media_Asf_Object_Unknown

- Zend_Media_Id3v1
- Zend_Media_Id3v2
- Zend_Media_Id3_Encoding
- Zend_Media_Id3_Exception
- Zend_Media_Id3_ExtendedHeader
- Zend_Media_Id3_Frame
- Zend_Media_Id3_Frame_AbstractLink
- Zend_Media_Id3_Frame_AbstractText
- Zend_Media_Id3_Frame_Aenc
- Zend_Media_Id3_Frame_Apic
- Zend_Media_Id3_Frame_Aspi
- Zend_Media_Id3_Frame_Comm
- Zend_Media_Id3_Frame_Comr
- Zend_Media_Id3_Frame_Encr
- Zend_Media_Id3_Frame_Equ2
- Zend_Media_Id3_Frame_Equa
- Zend_Media_Id3_Frame_Etco
- Zend_Media_Id3_Frame_Geob
- Zend_Media_Id3_Frame_Grid
- Zend_Media_Id3_Frame_Ipls
- Zend_Media_Id3_Frame_Isrc
- Zend_Media_Id3_Frame_Link
- Zend_Media_Id3_Frame_Mcdi
- Zend_Media_Id3_Frame_Mllt
- Zend_Media_Id3_Frame_Owne
- Zend_Media_Id3_Frame_Pcnt
- Zend_Media_Id3_Frame_Popm
- Zend_Media_Id3_Frame_Poss
- Zend_Media_Id3_Frame_Priv
- Zend_Media_Id3_Frame_Rbuf
- Zend_Media_Id3_Frame_Rva2
- Zend_Media_Id3_Frame_Rvad
- Zend_Media_Id3_Frame_Rvrb
- Zend_Media_Id3_Frame_Seek
- Zend_Media_Id3_Frame_Sign
- Zend_Media_Id3_Frame_Sylt
- Zend_Media_Id3_Frame_Sytc
- Zend_Media_Id3_Frame_Talb
- Zend_Media_Id3_Frame_Tbpm
- Zend_Media_Id3_Frame_Tcom
- Zend_Media_Id3_Frame_Tcon
- Zend_Media_Id3_Frame_Tcop
- Zend_Media_Id3_Frame_Tdat
- Zend_Media_Id3_Frame_Tden
- Zend_Media_Id3_Frame_Tdly
- Zend_Media_Id3_Frame_Tdor
- Zend_Media_Id3_Frame_Tdrc
- Zend_Media_Id3_Frame_Tdrl
- Zend_Media_Id3_Frame_Tdtg
- Zend_Media_Id3_Frame_Tenc
- Zend_Media_Id3_Frame_Text
- Zend_Media_Id3_Frame_Tflt
- Zend_Media_Id3_Frame_Time
- Zend_Media_Id3_Frame_Tipl
- Zend_Media_Id3_Frame_Tit1
- Zend_Media_Id3_Frame_Tit2
- Zend_Media_Id3_Frame_Tit3
- Zend_Media_Id3_Frame_Tkey
- Zend_Media_Id3_Frame_Tlan
- Zend_Media_Id3_Frame_Tlen
- Zend_Media_Id3_Frame_Tmcl
- Zend_Media_Id3_Frame_Tmed
- Zend_Media_Id3_Frame_Tmoo
- Zend_Media_Id3_Frame_Toal
- Zend_Media_Id3_Frame_Tofn
- Zend_Media_Id3_Frame_Toly
- Zend_Media_Id3_Frame_Tope
- Zend_Media_Id3_Frame_Tory
- Zend_Media_Id3_Frame_Town
- Zend_Media_Id3_Frame_Tpe1
- Zend_Media_Id3_Frame_Tpe2
- Zend_Media_Id3_Frame_Tpe3
- Zend_Media_Id3_Frame_Tpe4
- Zend_Media_Id3_Frame_Tpos
- Zend_Media_Id3_Frame_Tpro

- Zend_Media_Id3_Frame_Tpub
- Zend_Media_Id3_Frame_Trck
- Zend_Media_Id3_Frame_Trda
- Zend_Media_Id3_Frame_Trsn
- Zend_Media_Id3_Frame_Trso
- Zend_Media_Id3_Frame_Tsiz
- Zend_Media_Id3_Frame_Tsoa
- Zend_Media_Id3_Frame_Tsop
- Zend_Media_Id3_Frame_Tsot
- Zend_Media_Id3_Frame_Tsse
- Zend_Media_Id3_Frame_Tsst
- Zend_Media_Id3_Frame_Txxx
- Zend_Media_Id3_Frame_Tyer
- Zend_Media_Id3_Frame_Unknown
- Zend_Media_Id3_Frame_User
- Zend_Media_Id3_Frame_Uslt
- Zend_Media_Id3_Frame_Wcom
- Zend_Media_Id3_Frame_Wcop
- Zend_Media_Id3_Frame_Woaf
- Zend_Media_Id3_Frame_Woar
- Zend_Media_Id3_Frame_Woas
- Zend_Media_Id3_Frame_Wors
- Zend_Media_Id3_Frame_Wpay
- Zend_Media_Id3_Frame_Wpub
- Zend_Media_Id3_Frame_Wxxx
- Zend_Media_Id3_Header
- Zend_Media_Id3_Language
- Zend_Media_Id3_Object
- Zend_Media_Id3_Timing

- Zend_Media_Iso14496
- Zend_Media_Iso14496_Box
- Zend_Media_Iso14496_Box_Bxml
- Zend_Media_Iso14496_Box_Cdsc
- Zend_Media_Iso14496_Box_Co64
- Zend_Media_Iso14496_Box_Cprt
- Zend_Media_Iso14496_Box_Ctts
- Zend_Media_Iso14496_Box_Data
- Zend_Media_Iso14496_Box_Dinf
- Zend_Media_Iso14496_Box_Dref
- Zend_Media_Iso14496_Box_Edts
- Zend_Media_Iso14496_Box_Elst
- Zend_Media_Iso14496_Box_Free
- Zend_Media_Iso14496_Box_Frma
- Zend_Media_Iso14496_Box_Ftyp
- Zend_Media_Iso14496_Box_Full
- Zend_Media_Iso14496_Box_Hdlr
- Zend_Media_Iso14496_Box_Hint
- Zend_Media_Iso14496_Box_Hmhd
- Zend_Media_Iso14496_Box_Id32
- Zend_Media_Iso14496_Box_Iinf
- Zend_Media_Iso14496_Box_Iloc
- Zend_Media_Iso14496_Box_Ilst
- Zend_Media_Iso14496_Box_Imif
- Zend_Media_Iso14496_Box_Infe
- Zend_Media_Iso14496_Box_Ipmc
- Zend_Media_Iso14496_Box_Ipro
- Zend_Media_Iso14496_Box_Mdat
- Zend_Media_Iso14496_Box_Mdhd
- Zend_Media_Iso14496_Box_Mdia
- Zend_Media_Iso14496_Box_Mehd
- Zend_Media_Iso14496_Box_Meta
- Zend_Media_Iso14496_Box_Mfhd
- Zend_Media_Iso14496_Box_Mfra
- Zend_Media_Iso14496_Box_Mfro
- Zend_Media_Iso14496_Box_Minf
- Zend_Media_Iso14496_Box_Moof
- Zend_Media_Iso14496_Box_Moov
- Zend_Media_Iso14496_Box_Mvex
- Zend_Media_Iso14496_Box_Mvhd
- Zend_Media_Iso14496_Box_Nmhd
- Zend_Media_Iso14496_Box_Padb
- Zend_Media_Iso14496_Box_Pdin
- Zend_Media_Iso14496_Box_Pitm
- Zend_Media_Iso14496_Box_Sbgp

- Zend_Media_Iso14496_Box_Schi
- Zend_Media_Iso14496_Box_Schm
- Zend_Media_Iso14496_Box_Sdtp
- Zend_Media_Iso14496_Box_Sgpd
- Zend_Media_Iso14496_Box_Sinf
- Zend_Media_Iso14496_Box_Skip
- Zend_Media_Iso14496_Box_Smhd
- Zend_Media_Iso14496_Box_Stbl
- Zend_Media_Iso14496_Box_Stco
- Zend_Media_Iso14496_Box_Stdp
- Zend_Media_Iso14496_Box_Stsc
- Zend_Media_Iso14496_Box_Stsd
- Zend_Media_Iso14496_Box_Stsh
- Zend_Media_Iso14496_Box_Stss
- Zend_Media_Iso14496_Box_Stsz
- Zend_Media_Iso14496_Box_Stts
- Zend_Media_Iso14496_Box_Stz2
- Zend_Media_Iso14496_Box_Subs
- Zend_Media_Iso14496_Box_Tfhd
- Zend_Media_Iso14496_Box_Tfra
- Zend_Media_Iso14496_Box_Tkhd
- Zend_Media_Iso14496_Box_Traf
- Zend_Media_Iso14496_Box_Trak
- Zend_Media_Iso14496_Box_Tref
- Zend_Media_Iso14496_Box_Trex
- Zend_Media_Iso14496_Box_Trun
- Zend_Media_Iso14496_Box_Udta
- Zend_Media_Iso14496_Box_Url
- Zend_Media_Iso14496_Box_Urn
- Zend_Media_Iso14496_Box_Vmhd
- Zend_Media_Iso14496_Box_Xml
- Zend_Media_Iso14496_Exception

- Zend_Media_Mpeg_Abs
- Zend_Media_Mpeg_Ps
- Zend_Media_Mpeg_Abs_Frame
- Zend_Media_Mpeg_Abs_LameHeader
- Zend_Media_Mpeg_Abs_Object
- Zend_Media_Mpeg_Abs_VbriHeader
- Zend_Media_Mpeg_Abs_XingHeader
- Zend_Media_Mpeg_Exception
- Zend_Media_Mpeg_Object

8. Use Cases

9. Class Skeletons

Zend_Media_Id3v1

```
final class Zend_Media_Id3v1
{
    /**
     * The genre list.
     *
     * @var Array
     */
    public static $genres = array
        ('Blues', 'Classic Rock', 'Country', 'Dance', 'Disco', 'Funk', 'Grunge',
         'Hip-Hop', 'Jazz', 'Metal', 'New Age', 'Oldies', 'Other', 'Pop', 'R&B',
         'Rap', 'Reggae', 'Rock', 'Techno', 'Industrial', 'Alternative', 'Ska',
         'Death Metal', 'Pranks', 'Soundtrack', 'Euro-Techno', 'Ambient',
```

```
     'Trip-Hop', 'Vocal', 'Jazz+Funk', 'Fusion', 'Trance', 'Classical',
     'Instrumental', 'Acid', 'House', 'Game', 'Sound Clip', 'Gospel',
     'Noise', 'AlternRock', 'Bass', 'Soul', 'Punk', 'Space', 'Meditative',
     'Instrumental Pop', 'Instrumental Rock', 'Ethnic', 'Gothic',
     'Darkwave', 'Techno-Industrial', 'Electronic', 'Pop-Folk', 'Eurodance',
     'Dream', 'Southern Rock', 'Comedy', 'Cult', 'Gangsta', 'Top 40',
     'Christian Rap', 'Pop/Funk', 'Jungle', 'Native American', 'Cabaret',
     'New Wave', 'Psychadelic', 'Rave', 'Showtunes', 'Trailer', 'Lo-Fi',
     'Tribal', 'Acid Punk', 'Acid Jazz', 'Polka', 'Retro', 'Musical',
     'Rock & Roll', 'Hard Rock', 'Folk', 'Folk-Rock', 'National Folk',
     'Swing', 'Fast Fusion', 'Bebob', 'Latin', 'Revival', 'Celtic',
     'Bluegrass', 'Avantgarde', 'Gothic Rock', 'Progressive Rock',
     'Psychedelic Rock', 'Symphonic Rock', 'Slow Rock', 'Big Band',
     'Chorus', 'Easy Listening', 'Acoustic', 'Humour', 'Speech', 'Chanson',
     'Opera', 'Chamber Music', 'Sonata', 'Symphony', 'Booty Bass', 'Primus',
     'Porn Groove', 'Satire', 'Slow Jam', 'Club', 'Tango', 'Samba',
     'Folklore', 'Ballad', 'Power Ballad', 'Rhythmic Soul', 'Freestyle',
     'Duet', 'Punk Rock', 'Drum Solo', 'A capella', 'Euro-House',
     'Dance Hall', 255 => 'Unknown');

/**
 * Constructs the Id3v1 class with given file. The file is not mandatory
 * argument and may be omitted as a new tag can be written to a file also by
 * giving the filename to the {@link #write} method of this class.
 *
 * @param string|resource|Zend_Io_Reader $filename The path to the file,
 *  file descriptor of an opened file, or a {@link Zend_Io_Reader} instance.
 * @throws Zend_Media_Id3_Exception if given file descriptor is not valid
 */
public function __construct($filename = null);

/**
 * Returns the title field.
 *
 * @return string
 */
public function getTitle();

/**
 * Sets a new value for the title field. The field cannot exceed 30
 * characters in length.
 *
 * @param string $title The title.
 */
public function setTitle($title);

/**
 * Returns the artist field.
 *
 * @return string
 */
public function getArtist();

/**
 * Sets a new value for the artist field. The field cannot exceed 30
 * characters in length.
 *
 * @param string $artist The artist.
 */
public function setArtist($artist);

/**
 * Returns the album field.
 *
 * @return string
 */
public function getAlbum();

/**
 * Sets a new value for the album field. The field cannot exceed 30
```

```
     * characters in length.
     *
     * @param string $album The album.
     */
    public function setAlbum($album);

    /**
     * Returns the year field.
     *
     * @return string
     */
    public function getYear();

    /**
     * Sets a new value for the year field. The field cannot exceed 4
     * characters in length.
     *
     * @param string $year The year.
     */
    public function setYear($year);

    /**
     * Returns the comment field.
     *
     * @return string
     */
    public function getComment();

    /**
     * Sets a new value for the comment field. The field cannot exceed 30
     * characters in length.
     *
     * @param string $comment The comment.
     */
    public function setComment($comment);

    /**
     * Returns the track field.
     *
     * @since ID3v1.1
     * @return integer
     */
    public function getTrack();

    /**
     * Sets a new value for the track field. By setting this field you enforce
     * the 1.1 version to be used.
     *
     * @since ID3v1.1
     * @param integer $track The track number.
     */
    public function setTrack($track);

    /**
     * Returns the genre.
     *
     * @return string
     */
    public function getGenre();

    /**
     * Sets a new value for the genre field. The value may either be a numerical
     * code representing one of the genres, or its string variant.
     *
     * The genre is set to unknown (code 255) in case the string is not found
     * from the static {@link $genres} array of this class.
     *
     * @param integer $genre The genre.
     */
    public function setGenre($genre);
```

```
    /**
     * Writes the possibly altered ID3v1 tag back to the file where it was read.
     * If the class was constructed without a file name, one can be provided
     * here as an argument. Regardless, the write operation will override
     * previous tag information, if found.
     *
     * @param string $filename The optional path to the file.
     * @throws Zend_Media_Id3_Exception if there is no file to write the tag to
     */
    public function write($filename = null);

    /**
     * Magic function so that $obj->value will work.
     *
     * @param string $name The field name.
     * @return mixed
     */
    public function __get($name);

    /**
     * Magic function so that assignments with $obj->value will work.
     *
     * @param string $name  The field name.
     * @param string $value The field value.
     * @return mixed
```

```
     */
    public function __set($name, $value);
}
```

Zend_Media_Id3v2

```
final class Zend_Media_Id3v2 extends Zend_Media_Id3_Object
{
    /**
     * Constructs the Zend_Media_Id3v2 class with given file and options. The
     * options array may also be given as the only parameter.
     *
     * The following options are currently recognized:
     *   o encoding -- Indicates the encoding that all the texts are presented
     *     with. See the documentation of iconv for supported values. Please
     *     note that write operations do not convert string and thus encodings
     *     are limited to those supported by the {@link Zend_Media_Id3_Encoding}
     *     interface.
     *   o version -- The ID3v2 tag version to use in write operation. This
     *     option is automatically set when a tag is read from a file and
     *     defaults to version 4.0 for tag write.
     *   o readonly -- Indicates that the tag is read from a temporary file or
     *     another source it cannot be written back to. The tag can, however,
     *     still be written to another file.
     *
     * @todo  Only limited subset of flags are processed.
     * @todo  Utilize the SEEK frame and search for a footer to find the tag
     * @todo  Utilize the LINK frame to fetch frames from other sources
     * @param string|resource|Zend_Io_Reader $filename The path to the file,
     *  file descriptor of an opened file, or a {@link Zend_Io_Reader} instance.
     * @param Array                          $options  The options array.
     * @throws Zend_Media_Id3_Exception if given file descriptor is not valid
     */
    public function __construct($filename = null, $options = array());

    /**
     * Returns the header object.
     *
     * @return Zend_Media_Id3_Header
     */
    public function getHeader();

    /**
     * Checks whether there is an extended header present in the tag. Returns
     * <var>true</var> if the header is present, <var>false</var> otherwise.
     *
     * @return boolean
     */
    public function hasExtendedHeader();

    /**
     * Returns the extended header object if present, or <var>false</var>
     * otherwise.
     *
     * @return Zend_Media_Id3_ExtendedHeader|false
     */
    public function getExtendedHeader();

    /**
     * Sets the extended header object.
     *
     * @param Zend_Media_Id3_ExtendedHeader $extendedHeader The header object
     */
    public function setExtendedHeader($extendedHeader);
```

```
/**
 * Checks whether there is a frame given as an argument defined in the tag.
 * Returns <var>true</var> if one ore more frames are present,
 * <var>false</var> otherwise.
 *
 * @param string $identifier The frame name.
 * @return boolean
 */
public function hasFrame($identifier);

/**
 * Returns all the frames the tag contains as an associate array. The frame
 * identifiers work as keys having an array of frames as associated value.
 *
 * @return Array
 */
public function getFrames();

/**
 * Returns an array of frames matching the given identifier or an empty
 * array if no frames matched the identifier.
 *
 * The identifier may contain wildcard characters '*' and '?'. The asterisk
 * matches against zero or more characters, and the question mark matches
 * any single character.
 *
 * Please note that one may also use the shorthand $obj->identifier to
 * access the first frame with the identifier given. Wildcards cannot be
 * used with the shorthand method.
 *
 * @param string $identifier The frame name.
 * @return Array
 */
public function getFramesByIdentifier($identifier);

/**
 * Removes any frames matching the given object identifier.
 *
 * The identifier may contain wildcard characters '*' and '?'. The asterisk
 * matches against zero or more characters, and the question mark matches
 * any single character.
 *
 * One may also use the shorthand unset($obj->identifier) to achieve the
 * same result. Wildcards cannot be used with the shorthand method.
 *
 * @param string $identifier The frame name.
 */
public final function removeFramesByIdentifier($identifier);

/**
 * Adds a new frame to the tag and returns it.
 *
 * @param Zend_Media_Id3_Frame $frame The frame to add.
 * @return Zend_Media_Id3_Frame
 */
public function addFrame($frame);

/**
 * Remove the given frame from the tag.
 *
 * @param Zend_Media_Id3_Frame $frame The frame to remove.
 */
public function removeFrame($frame);

/**
 * Checks whether there is a footer present in the tag. Returns
 * <var>true</var> if the footer is present, <var>false</var> otherwise.
 *
 * @return boolean
```

```
     */
    public function hasFooter();

    /**
     * Returns the footer object if present, or <var>false</var> otherwise.
     *
     * @return Zend_Media_Id3_Header|false
     */
    public function getFooter();

    /**
     * Sets whether the tag should have a footer defined.
     *
     * @param boolean $useFooter Whether the tag should have a footer
     */
    public function setFooter($useFooter);

    /**
     * Writes the possibly altered ID3v2 tag back to the file where it was read.
     * If the class was constructed without a file name, one can be provided
     * here as an argument. Regardless, the write operation will override
     * previous tag information, if found.
     *
     * If write is called without setting any frames to the tag, the tag is
     * removed from the file.
     *
     * @param string $filename The optional path to the file.
     */
    public function write($filename);

    /**
     * Magic function so that $obj->value will work. The method will attempt to
     * return the first frame that matches the identifier.
     *
     * If there is no frame or field with given name, the method will attempt to
     * create a frame with given identifier.
     *
     * If none of these work, an exception is thrown.
     *
     * @param string $name The frame or field name.
     * @return mixed
     */
    public function __get($name);

    /**
     * Magic function so that isset($obj->value) will work. This method checks
     * whether the frame matching the identifier exists.
     *
     * @param string $name The frame identifier.
     * @return boolean
     */
    public function __isset($name);

    /**
     * Magic function so that unset($obj->value) will work. This method removes
     * all the frames matching the identifier.
     *
     * @param string $name The frame identifier.
```

```
      */
     public function __unset($name);
 }
```

]]></ac:plain-text-body></ac:macro>

]]></ac:plain-text-body></ac:macro>