// Module identifier: "mawg"

// Qualified name: "::mawg"

module mawg {

exception WrongMethod{

const DOMString errorMessage = ”Property undefined for this media type.”

};

exception NoValue{

const DOMString errorMessage = ”No value available for this property.”

};

// Interface identifier: "System"

// Qualified name: "::framework::System"

interface MediaResource {

// An attribute whose value cannot be assigned to, and which can raise an exception in some circumstances.

// Identification

readonly attribute DOMString identifier getraises (NoValue);

readonly attribute DOMString title getraises (NoTitle);

readonly attribute DOMString language getraises (NoValue);

readonly attribute DOMString locator getraises (NoValue);

// Creation

readonly attribute object contributor getraises (NoValue);

readonly attribute DOMString[] creator getraises (NoValue);

//problem, currently there are no provisions for the Date type

readonly attribute object createDate getraises (NoValue);

readonly attribute object location getraises (NoValue);

//Content Description

readonly attribute DOMString description getraises (NoValue);

readonly attribute DOMString[] keyword getraises (NoValue);

readonly attribute DOMString genre getraises (NoValue);

readonly attribute object rating getraises (NoValue);

// Relational

readonly attribute object relation getraises (NoValue);

readonly attribute DOMString collection getraises (NoValue);

// Rights

readonly attribute object[] copyright getraises (NoValue);

readonly attribute object license getraises (NoValue);

// Distribution

readonly attribute DOMString publisher getraises (NoValue);

readonly attribute object targetAudience getraises (NoValue);

// Fragments

readonly attribute object[] fragments getraises (NoValue);

readonly attribute object[] namedFragments getraises (NoValue);

// Technical Properties

readonly attribute object frameSize getraises (NoValue);

readonly attribute DOMString compression getraises (NoValue);

//how to denote time interval? Assumption: use msecs

readonly attribute unsigned long duration getraises (NoValue);

readonly attribute DOMString format getraises (NoValue);

readonly attribute unsigned long samplingrate getraises (NoValue);

//remark the unit is frames per second

readonly attribute float framerate getraises (NoValue);

//remark, what would be the unit? (we assume kbps)

readonly attribute float bitrate getraises (NoValue);

readonly attribute unsigned short numTracks getraises (NoValue);

};

// Module identifier: " returnValues "

// Qualified name: "::mawg:: returnValues "

module returnValues {

interface Unstructured {

attribute DOMString unstructuredValue;

};

interface Contributor: Unstructured {

attribute DOMString id;

attribute DOMString role;

};

interface Location: Unstructured {

// to be defined what the Location is …

};

interface Rating: Unstructured {

/\*example: [http://www.individuals.com/ChrisPoppe, 10.0, 0, 10.0, "quality"]

(Rating issuer: http://www.individuals.com/ChrisPoppe, Rating value: 10.0, Rating min: 0, Rating max: 10.0, Rating context: "quality") \*/

attribute DOMString issuer;

attribute short value;

attribute short minimum;

attribute short maximum;

attribute DOMString context;

};

interface Relation: Unstructured {

attribute DOMString id;

attribute DOMString relationship;

};

interface Copyright: Unstructured {

attribute DOMString statement;

attribute DOMString[] holder;

};

interface License: Unstructured {

attribute DOMString statement;

attribute DOMString organization;

};

interface TargetAudience: Unstructured {

attribute DOMString issuer;

attribute DOMString classification;

};

interface Fragment: Unstructured {

attribute DOMString role;

attribute DOMString identifier;

};

interface NamedFragment: Unstructured {

attribute DOMString name;

attribute DOMString identifier;

};

//note we are assuming the use of pixels…

interface FrameSize: Unstructured {

attribute unsigned long width;

attribute unsigned long height;

};

interface Duration: Unstructured {

attribute DOMString statement;

attribute DOMString holder;

};

};

};