
XDM Documentation

Release 0.5.19

Dennis Lutter

August 16, 2015

1	Plugins	3
1.1	DownloadFilter	3
1.2	DownloadType	3
1.3	Downloader	4
1.4	Indexer	4
1.5	MediaAdder	5
1.6	MediaTypeManager	5
1.7	Notifier	6
1.8	PostProcessor	6
1.9	Provider	6
1.10	SearchTermFilter	7
1.11	System	7
1.12	PluginManager	7
1.13	Plugin configuration	8
1.14	Repository	11
1.15	SystemConfig plugin	12
1.16	Getting started	12
1.17	Reserved attributes	14
1.18	Plugin Types	14
1.19	PluginRootLibrarys	14
1.20	Plugin	14
2	API	17
2.1	Access	17
2.2	Namespace: None	17
2.3	Namespace: system	18
2.4	Namespace: plugins	18
3	Core	19
3.1	ORM Classes	19
3.2	Init	23
3.3	GUI Message system	24
3.4	News feed system	24
3.5	Scheduler	24
3.6	Updater	25
3.7	Boot structure	25
3.8	The common object	25

4	Usage	27
5	Libraries in use	29
5.1	Backend (Python)	29
5.2	Frontend (css/js/etc)	29
	Python Module Index	31

XDM is in BETA Official site <http://xdm.lad1337.de> Official main plugin repository at <https://github.com/lad1337/XDM-main-plugin-repo/>

This documentation consists of 3 main sections:

Contents:

1.1 DownloadFilter

class `xm.plugins.bases.DownloadFilter` (*instance='Default'*)

These plugins filter available downloads like video quality.

getConfigHtml ()

The return value of this function is simply injected in the config panel

default is a html comment: `<!-- nothing here -->` you can use this to add dynamic javascript and/or add additional html

Note: The container this is added in is hidden with inline css. to make use of the html use javascript

1.2 DownloadType

class `xm.plugins.bases.DownloadType` (*instance=None*)

These plugins define the type of download like NZB.

extension = ''

The file extension if a file is created for this DownloadType

getConfigHtml ()

The return value of this function is simply injected in the config panel

default is a html comment: `<!-- nothing here -->` you can use this to add dynamic javascript and/or add additional html

Note: The container this is added in is hidden with inline css. to make use of the html use javascript

identifier = ''

A absolute unique identifier in reverse URL style e.g. `de.lad1337.nzb`

1.3 Downloader

class `xdm.plugins.bases.Downloader` (*instance='Default'*)

These plugins of this class send Downloads to another Programs or directly download stuff.

addDownload (*download*)

Add/download a Download to this downloader

Arguments: download – an Download object

return: bool if the adding was successful >>>> False

getConfigHtml ()

The return value of this function is simply injected in the config panel

default is a html comment: <!-- nothing here --> you can use this to add dynamic javascript and/or add additional html

Note: The container this is added in is hidden with inline css. to make use of the html use javascript

getDownloadPercentage (*element*)

“this should return a int between 0 and 100 as the percentage

getElementStaus (*element*)

Get the staus of element that it has in this downloader

Arguments: element – an Element object

return: tuple of Status, Download and a path (str) >>>> (common.UNKNOWN, Download(), ‘’)

1.4 Indexer

class `xdm.plugins.bases.Indexer` (*instance='Default'*)

These plugins perform searches on sites that hold information of available files.

Plugins of this class create elemnts based on mediaType structures

getConfigHtml ()

The return value of this function is simply injected in the config panel

default is a html comment: <!-- nothing here --> you can use this to add dynamic javascript and/or add additional html

Note: The container this is added in is hidden with inline css. to make use of the html use javascript

getDownloadPercentage (*element*)

“this should return a int between 0 and 100 as the percentage

getLatestRss ()

return list of Dowloads

searchForElement (*element*)

Returns a list of Downloads that where found for element

For each download following attributes are set automatically after this is called download.indexer = self.type download.indexer_instance = self.instance download.status = common.UNKNOWN

1.5 MediaAdder

class `xdm.plugins.bases.MediaAdder` (*instance=None*)

These plugins are called periodically and can add/manipulate media entries from other sources.

class `Media` (*mediaTypeIdentifier, externalID, providerTag, elementType, name, additionalData={}*)

Class that holds the information needed by XDM to add the Media

root = None

this is set by XDM of the Media was succesfully added, it is the element(root) that represents the Media in XDM

`MediaAdder.getConfigHtml()`

The return value of this function is simply injected in the config panel

default is a html comment: `<!-- nothing here -->` you can use this to add dynamic javascript and/or add additional html

Note: The container this is added in is hidden with inline css. to make use of the html use javascript

`MediaAdder.runShedule()`

This method is called periodically and has to return a list of Media objects

1.6 MediaTypeManager

class `xdm.plugins.bases.MediaTypeManager` (*instance*)

These plugins define what XDM manages e.g. Movies, Books etc. They also define how each type looks.

Plugins of this type define a “MediaType” e.g. Movies They define a Structure of Classes simple classes that resemble Objects needed for the media that is being reflected.

addConfig = {}

Add configuration to otherplugins

keys a plugin base class reference

value list of dicts `[{'type': 'category', 'default': None, 'prefix': 'Category for', 'sufix': 'Movies'}]`
 TODO: explain more

defaultElements = []

list of dicts Elements to create

e.g. `{'tgdb': {'name': 'Nintendo Wii', 'alias': 'Wii', 'id': 9}}`

Note: each list item is a dict that can contain multiple provider informations for one Element. In the above example we create the Wii with given attributes for the provider that has the tag tgdb (thegamesdb.net)

download = None

The class to download/search for

elementConfigsFor = ()

Tuple of classes that will have a configuration on each plugin e.g. Platform of de.lad1337.games TODO:
 link Platform

getConfigHtml()

The return value of this function is simply injected in the config panel

default is a html comment: <!-- nothing here --> you can use this to add dynamic javascript and/or add additional html

Note: The container this is added in is hidden with inline css. to make use of the html use javascript

identifier = ''

A absolute unique identifier in reverse URL style e.g. de.lad1337.nzb

order = ()

A tuple that defines the order top to bottom / outer to inner of the classes that reflect the media

search (*search_query*)

Search the ONE provider for search_query the provider is either the first one or the provider that is set as the "searcher"

1.7 Notifier

class `xdm.plugins.bases.Notifier` (**args, **kwargs*)

These plugins send out notifications.

getConfigHtml ()

The return value of this function is simply injected in the config panel

default is a html comment: <!-- nothing here --> you can use this to add dynamic javascript and/or add additional html

Note: The container this is added in is hidden with inline css. to make use of the html use javascript

1.8 PostProcessor

class `xdm.plugins.bases.PostProcessor` (*instance='Default'*)

These plugins act on downloaded items like move and rename files.

getConfigHtml ()

The return value of this function is simply injected in the config panel

default is a html comment: <!-- nothing here --> you can use this to add dynamic javascript and/or add additional html

Note: The container this is added in is hidden with inline css. to make use of the html use javascript

postProcessPath (*element, path*)

should return a tuple with overall result bool() new location str() process log str()

1.9 Provider

class `xdm.plugins.bases.Provider` (*instance='Default'*)

These plugins get meta information for the MediaTypes and perform searches.

Plugins of this class create elemnets based on mediaType structures.

creating more providers is definitely more complicated than other things since creating element structures based on the structure defined by the mediaType can be complicated

getConfigHtml ()

The return value of this function is simply injected in the config panel

default is a html comment: <!-- nothing here --> you can use this to add dynamic javascript and/or add additional html

Note: The container this is added in is hidden with inline css. to make use of the html use javascript

searchForElement (*term*='')

Create a MediaType structure of the type of element.mediaType

1.10 SearchTermFilter

class xdm.plugins.bases.**SearchTermFilter** (*instance=None*)

These plugins manipulate the search terms, create new ones or remove others.

getConfigHtml ()

The return value of this function is simply injected in the config panel

default is a html comment: <!-- nothing here --> you can use this to add dynamic javascript and/or add additional html

Note: The container this is added in is hidden with inline css. to make use of the html use javascript

1.11 System

class xdm.plugins.bases.**System** (*instance=None*)

These plugins control global issues such as XDMs config.

getConfigHtml ()

The return value of this function is simply injected in the config panel

default is a html comment: <!-- nothing here --> you can use this to add dynamic javascript and/or add additional html

Note: The container this is added in is hidden with inline css. to make use of the html use javascript

1.12 PluginManager

class xdm.plugins.pluginManager.**PluginManager** (*path='corePlugins'*)

Class that manages/provides and finds plugins

find_subclasses (*cls, reloadModule=False, debug=False, path=''*)

Find all subclass of cls in py files located below path (does look in sub directories)

@param path: the path to the top level folder to walk @type path: str @param cls: the base class that all subclasses should inherit from @type cls: class @rtype: list @return: a list of classes that are subclasses of cls

pylintScoreError = 4

send an error when score is bellow or equal this

pylintScoreWarning = 7

send a warning when score is bellow or equal this

1.13 Plugin configuration

Contents:

XDM provides a framework to store basic configuration and display there form fields on the settings page.

There are three types of plugins config “scopes”:

type/scope	access	defined in	meta defined in
normal	self.c	<code>_config</code>	<code>config_meta</code>
hidden	self.hc	<code>_hidden_config</code>	<code>hidden_config_meta</code>
element	self.e	<code>_config</code>	<code>config_meta</code>

1.13.1 Example

To create settings simply fill the `_config` attribute of your plugin class:

```
from xdm.plugins import *

class Printer(Notifier):
    _config = {'add_message': 'my print plugin'}
    # add_message: same key as in _config
    config_meta = {'add_message': {'human': 'Additional text', # a more human friendly name
                                   'desc': 'This text will be added after every printed message'}}
    # the description will be a tooltip on the input field

    def sendMessage(msg, element=None):
        userAddition = self.c.add_message
        print msg + userAddition
        return True
```

By default settings are ordered alphabetically by name if you want to have a specific order use an `OrderedDict`

```
from collections import OrderedDict
_config = OrderedDict([
    ('default_mt_select', ''),
    ('login_user', ''),
    ('login_password', ''),
    ('port', 8085)
])
```

1.13.2 Setting value types and resulting form fields

Values can be one of the the following type

type	Settings page field
str	Text input field
bool	Checkbox
int / float	Number input field

Note: Only these type are allowed as config values!

Additionally if the following strings are found in the the config name additional modification are made to the input field

string	modification
<i>path</i>	Adds a folder browser
<i>filepath</i>	Adds a file browser
<i>password</i>	Makes the input field an obfuscated password input field

1.13.3 Interaction between settings page and plugin

You can also interact from the settings page with your plugin.

- You can add generic buttons.
- You listen on events of the form fields.
- You can get the data from the config fields.
- You can send data back
- You can invoke custom javascript functions with your send data. See getConfigHtml
- You can trigger global actions

These are defined in the `config_meta` dict:

Buttons

To define single action buttons use the `plugin_buttons` key this holds another dict with button definitions

Button example:

```
config_meta = {'plugin_buttons': {'testConnection': {'action': _testConnection,
                                                    'name': 'Test Connection',
                                                    'desc': 'Test the connection with credentials'}}
               }
```

This will create a button on the settings page with the text “Test Connection” on it and a greyed out text “Test the connection with credentials” behind it. When clicked it will call the function `_testConnection`

Note: Since the “action” is a real python reference the `config_meta` should be at the end your your plugin class.

Events

To add “actions” to changes that are made with the input field (text is changed and focus is lost) you set the “on_live_change” tot the function you want to call

Event example:

```
config_meta = {'plugin_desc': 'Sabnzb downloader. Send Nzbs and check for status',
               'plugin_buttons': {'test_connection': {'action': _testConnection, 'name': 'Test connec
               'host': {'on_live_change': _testConnection},
               'port': {'on_live_change': _testConnection},
               'apikey': {'on_live_change': _testConnection}
               }
```

Now the function `_testConnection` called when either the field `host`, `port` or `apikey` is changed, or when the `test_connection` button is clicked

The functions

The functions that are called should always return a tuple of the following schema:

```
t = (bool, dict, str)
return t
# t[0] is the overall result status
# t[1] is data you want to send / it is converted into json
# t[2] is a human readable message
```

and to get data from the config page add a list named `args` with names of the config fields you want to the function (yes add it as an attribute to the function(object))

```
_config = {'host': 'http://nzbs2go.com',
           'apikey': '',
           'port': None,
           'enabled': True,
           'comment_on_download': False,
           'retention': 900,
           'verify_ssl_certificate': True
          }

def _gatherCategories(self, host, port, verify_ssl_certificate):
    return (True, {}, 'I found 3 categories')
_gatherCategories.args = ['host', 'port', 'verify_ssl_certificate'] # <- this is what i mean
```

Use the data in javascript

To make use of data in javascript / your function you modify the data-dict (`t[1]`) has to in the following structure:

```
dataWrapper = {'callFunction': 'javascript_function_name',
               'functionData': data}
```

Note: since all plugins can have multiple instances the functions should be namespaced with the current instance name. See following example

Full example:

```
class Newznab(Indexer):
    identifier = "de.lad1337.newznab"
    _config = {'host': 'http://nzbs2go.com',
               'apikey': '',
               'port': None,
               'enabled': True,
               'comment_on_download': False,
               'retention': 900,
               'verify_ssl_certificate': True
              }

    def _gatherCategories(self, host, port, verify_ssl_certificate):
        # some magic that does stuff
        data = dict()
        dataWrapper = {'callFunction': 'newsznab_' + self.instance + '_spreadCategories', # use the
```

```

        'functionData': data}

    return (True, dataWrapper, 'I found %s categories' % len(data))
_gatherCategories.args = ['host', 'port', 'verify_ssl_certificate']

def getConfigHtml(self):
    return """<script>
        function newsznab_""" + self.instance + """_spreadCategories(data){ """ # this gives
        """console.log(data);
        $.each(data, function(k,i){
            $('#""" + helper.idSafe(self.name) + """ input[name$="'+k+'"]').val(i)
        });
        };
    </script>

    """

config_meta = {'plugin_desc': 'Generic Newznab indexer. Categories are there numerical id of Newznab',
               'plugin_buttons': {'gather_categories': {'action': _gatherCategories, 'name': 'Get Categories'},
                                'test_connection': {'action': _testConnection, 'name': 'Test Connection'}},
               }

```

1.14 Repository

class xdm.plugins.repository.**RepoManager** (*repos*)

Class to manage all repositories and entry point for installing plugins

checkForUpdate ()

Check on all installed plugins if an update is available stores result in self.updateable_plugins and send messages to GUI

deinstall (*identifier*)

Deinstall a plugin by simply deleting the install folder recaches all plugins at the end

doCleanUp ()

- 1.Recaches plugins
- 2.Updates cherrypys static folders
- 3.cache repositories

hasUpdate (*identifier*)

Check if plugin has an update uses cache

install (*identifier*, *doCleanUp=True*)

install a plugin

- 1.checks if plugin is already installed
- 2.checks if plugins is installed but has an update, if so move the old version to __old__oldFolderName
- 3.chooses install method by the given “format” e.g. “zip”
- 4.use the Appropriate installer
- 5.if doCleanUp was True *doCleanUp* is called

1.15 SystemConfig plugin

1.16 Getting started

Things you should know:

- Each plugin is a subclass one of the *plugin types*.
- A .py file can contain multiple plugins.
- If you have a folder structure the main folder must be a python module.

Plugins are loaded dynamically by traversing the *core plugin path* and the `extraPluginPath`, each .py file is checked for any of the *plugin type* classes.

Basic folderstructure:

```
<extraPluginFolder>--+
|
|   +-<My Plugin Folder>--+
|                           |
|                           +- __init__.py (empty file)
|                           |
|                           +- <My Plugin .py>
|                           |
|                           +- <more .py that the plugin needs> (optional)
|                           |
|                           +- <folder with librarys> (optional)
|                           |
|                           +- pluginRootLibrarys (optional)
```

Note: Not all .py files are checked. Files that start with ‘__’ or files that are in a folder called *pluginRootLibrarys* are not checked for plugins.

1.16.1 A simple notification plugin

The file structure for this would look like:

```
<extraPluginFolder>--+
|
|   +-Printer--+
|               |
|               +- __init__.py
|               |
|               +- Printer.py
```

Contents of `printer.py`:

```
from xdm.plugins import * # by using this one import line you have everything a plugin will need from

class Printer(Notifier): # subclass Notifier

    def sendMessage(msg, element=None)
        print msg
        return True
```

This is absolutely sufficient for a Notifier plugin.

1.16.2 Adding configuration

To create settings simply fill the `_config` attribute of your plugin class:

```
from xdm.plugins import * # by using this one import line you have everything a plugin will need from xdm

class Printer(Notifier): # subclass Notifier
    _config = {'add_message': 'my print plugin'}

    def sendMessage(msg, element=None)
        print msg
        return True
```

This will create a text field in on the settings page and the default value is *my print plugin*.

Each type of default value will determine the input field in the settings

str Text input field

bool Checkbox

int / float Number input field

Note: Only these type are allowed as config values!

To get the config value access the `c` object of you plugin and then the attribute with the name of your config.

```
from xdm.plugins import *

class Printer(Notifier):
    _config = {'add_message': 'my print plugin'}

    def sendMessage(msg, element=None)
        userAddition = self.c.add_message # just make sure its the same spelling as the key in the config
        print msg + userAddition
        return True
```

Note: `self.c` is only available *after* the `__init__()` of *Plugin*.

It was never explained how the `add_message` is used, to change that we use the `config_meta` attribute.

```
from xdm.plugins import *

class Printer(Notifier):
    _config = {'add_message': 'my print plugin'}
    # add_message: same key as in _config
    config_meta = {'add_message': {'human': 'Additional text', # a more human friendly name
                                   'desc': 'This text will be added after every printed message'}}
    # the description will be a tooltip on the input field

    def sendMessage(msg, element=None)
        userAddition = self.c.add_message
        print msg + userAddition
        return True
```

More info on Plugin configuration

1.17 Reserved attributes

Following class / instance attributes are reserved and **are not to be set** by a plugin!

c The configuration manager

hc The hidden configuration manager

type The class name e.g. *Printer*

_type Its the plugin type name e.g. *Downloader*

instance The instance name

name The class name and instance name e.g. *Printer (Default)*

More info see [Plugin](#)

1.18 Plugin Types

DownloadType A simple description of a download type.

Downloader These plugins provider the connection to the downloader programs

Indexer These plugins provide connection to the sites that have information on releases

Notifier These plugins provide means to send notification to third party services or other program's on certain events

Provider These plugins connect to databases and create elements based on the media type they work for.

PostProcessor These plugins move rename or do whatever they like with a given path and element.

System Well there is only one of this kind and maybe there will never be more. The system config is a plugin of this type

DownloadFilter These plugins can filter found downloads as they wish.

SearchTermFilter These plugins can add or remove search terms, before they are used by the Provider

MediaAdder These plugins can add elements. From various sources e.g. [Trakt.tv](#) watchlist.

MediaTypeManager These plugins are the core of XDM. These plugins define the structure of a media type and provide functions to save and store them. The look is also defined here.

1.19 PluginRootLibarys

If you need to include a library that expects to by installed system wide, therefore being in the python path can be used when put in a folder called **pluginRootLibarys**. The absolute path of that folder is added to the python path.

Note: An info level message when the path is added it created.

1.20 Plugin

```
class xdm.plugins.bases.Plugin(instance=None)
    Plugin base class. loads the config on init
```

`_getUseConfigsForElementsAsWrapper` (*element*)

Gets the the config value for given element

`addMediaTypeOptions = True`

Defines if options defined by a MediaType should be added to the plugin.

Note: This is ignored for plugins of the type MediaTypeManager and System since it's **never** done for them.

bool

- True: this will add all configurations defined by a MediaType
- False: No configurations are added

list a list of MediaType identifiers e.g. ['de.lad1337.music', 'de.lad1337.games'], this will add only options from the MediaType with the given identifier

str only str value allowed is `runFor`, this will only add `runFor` options to the plugin.

`config_meta = {}`

Meta information on the the config keys

`getConfigHtml()`

The return value of this function is simply injected in the config panel

default is a html comment: `<!-- nothing here -->` you can use this to add dynamic javascript and/or add additional html

Note: The container this is added in is hidden with inline css. to make use of the html use javascript

`single = False`

if True the gui will not give the option for more configurations. but there is no logic to stop you do it anyways

`useConfigsForElementsAs = 'Category'`

MediaTypeManager can add configurations for elements. this defines how this configuration is used. This string will be used for:

- It will be added to the configurations name.
- A function will be created named `get<useConfigsForElementsAs_value>()` e.g. `getCategory()`

`version = '0.1'`

version string. may only have a major and a minor version number

`xdm_version = (0, 0, 0)`

this is the greater or equal XDM version this plugin needs to function

Note: By default this is (0, 0, 0) but the json builder will return the current version of XDM if it is not specifically set.

API

Things you should know:

- The API is a JSON-RPC api v2.0 more info see <http://en.wikipedia.org/wiki/JSON-RPC>
- It runs on its own port, default is 8086
- The api key MUST be present in all methods except *xdm.api.ping* and *xdm.api.version*

Note: At the time of writing i failed to make a connection to XDM / the api from a webpage due to cross origin domain security issues. I was also not able to add a websocket implementation to the api.

Warning: The api might completely change. goal is to have an api as XBMC provides. (Websockets and HTTP with an JSONRPC protocol)

2.1 Access

You must provider the api key either as the first none keyword argument or as a keyoword argument witht the name *apikey*.

2.1.1 Example in python

Using `JSONRPClib`

```
>>>> import jsonrpclib
>>>> server = jsonrpclib.Server('http://localhost:8086')
>>>> server.ping()
u'pong'
>>>> server.system.listMethods('6vVSMLSDB9vPZkftahVwfBirjzvYzPuih3V6hmO1Nhk')
[u'ping', u'plugins.cache', u'plugins.getActiveMediaTypes', u'system.listMethods', u'system.methodHe...
```

2.2 Namespace: None

`xdm.api.ping(pong='pong')`
Returns pong nice way to test the connections

`xdm.api.version()`
Returns the XDM version tuple as a list e.g. [0, 4, 13, 0]

2.3 Namespace: system

`xdm.api.system.listMethods()`

Return a list of available method names

`xdm.api.system.methodHelp (method_name='system.methodHelp')`

This method returns a text description of a particular method. The method takes one parameter, a string. Its value is the name of the jsonrpc method about which information is being requested. The result is a string. The value of that string is a text description, for human use, of the method in question.

Keyword arguments: `method_name` – str The name of the method (default 'system.methodHelp')

Return: str The documentation text of the method.

`xdm.api.system.methodSignature (method_name='system.methodSignature')`

Returns the signature of `method_name` as an list of list each list item is a list that contains the return value type as the first value all remaining values (if any) are value types that are accepted during call.

`xdm.api.system.reboot()`

just return True and starts a thread to reboot XDM

`xdm.api.system.shutdown()`

just return True and starts a thread to shutdown XDM

2.4 Namespace: plugins

`xdm.api.plugins.cache()`

(Re)cache plugins

`xdm.api.plugins.getActiveMediaTypes()`

Returns a list of active mediatype identifiers

3.1 ORM Classes

```
class xdm.classes.Status(*args, **kwargs)
```

```
    DoesNotExist
        alias of StatusDoesNotExist
    download_set
    download_v0_set
    download_v1_set
    element_set
    hidden = <peewee.BooleanField object>
    id = <peewee.PrimaryKeyField object>
    name = <peewee.CharField object>
    screenName
```

```
class xdm.classes.Config(*args, **kwargs)
```

```
    DoesNotExist
        alias of ConfigDoesNotExist
    copy()
    curType()
    element = <peewee.ForeignKeyField object>
    id = <peewee.PrimaryKeyField object>
    instance = <peewee.CharField object>
    mediaType = <peewee.ForeignKeyField object>
    module = <peewee.CharField object>
    name = <peewee.CharField object>
    section = <peewee.CharField object>
    type = <peewee.CharField object>
```

value

```
class xdm.classes.Download(*args, **kwargs)
```

DoesNotExist

alias of DownloadDoesNotExist

element = <peewee.ForeignKeyField object>

external_id = <peewee.CharField object>

extra_data

id = <peewee.PrimaryKeyField object>

indexer = <peewee.CharField object>

indexer_instance = <peewee.CharField object>

locations

name = <peewee.CharField object>

pp_log = <peewee.TextField object>

size = <peewee.IntegerField object>

status = <peewee.ForeignKeyField object>

type = <peewee.CharField object>

url = <peewee.CharField object>

classmethod where_extra_data (*items*)

let items be a dict with the keys and values you want

```
class xdm.classes.History(*args, **kwargs)
```

DoesNotExist

alias of HistoryDoesNotExist

element = <peewee.ForeignKeyField object>

event = <peewee.CharField object>

id = <peewee.PrimaryKeyField object>

new_obj = <peewee.TextField object>

obj_class = <peewee.CharField object>

obj_id = <peewee.IntegerField object>

obj_type = <peewee.CharField object>

old_obj = <peewee.TextField object>

time = <peewee.DateTimeField object>

```
class xdm.classes.Element(*args, **kwargs)
```

DoesNotExist

alias of ElementDoesNotExist

XDMID

`addLocation` (*path*, *download=None*)
`ancestors`
`buildFieldDict` ()
`buildHtml` (*search=False*, *curIndex=0*)
`children`
`clearCache` ()
`clearLowerTreeCache` ()
`clearTreeCache` ()
`clearUpperTreeCache` ()
`config_set`
`copy` ()
`decendants`
`deleteConfig` ()
`deleteDownloads` ()
`deleteFields` ()
`deleteHistory` ()
`deleteImages` ()
`deleteWithChildren` (*silent=False*)
`delete_instance` (*silent=False*)
`downloadImage` (*imageName*, *provider=''*)
`downloadImages` ()
`downloads`
`fields`
`getAnyImage` ()
`getConfig` (*plugin*, *configName*)
`getConfigs` ()
`getDefinedAttr` ()
`getField` (*name*, *provider=None*, *returnObject=False*)
`getIdentifier` ()
`getImage` (*name*, *provider=''*)
`getImageNames` ()
`getName` ()
`getReleaseDate` ()
`getSearchTemplate` ()
`getSearchTerms` ()
`getTemplate` ()

```
classmethod getWhereField (mt, type, attributes, provider='', parent=0)
getXDMID (tag=None)
history_set
id = <peewee.PrimaryKeyField object>
images
isAncestorOf (jungstar)
isDescendantOf (granny)
isReleaseDateInPast ()
leaf
locations
manager
mediaType = <peewee.ForeignKeyField object>
orderFieldValues
orderFields
orderReverse
paint (search=False, single=False, status=None, curIndex=0, onlyChildren=False)
parent = <peewee.ForeignKeyField object>
save (fixTemp=True)
saveTemp ()
setField (name, value, provider='')
status = <peewee.ForeignKeyField object>
type = <peewee.CharField object>
class xdm.classes.MediaType (*args, **kwargs)

    DoesNotExist
        alias of MediaTypeDoesNotExist
    config_set
    element_set
    id = <peewee.PrimaryKeyField object>
    identifier = <peewee.CharField object>
    manager
    name = <peewee.CharField object>
class xdm.classes.Field (*args, **kwargs)

    DoesNotExist
        alias of FieldDoesNotExist
    element = <peewee.ForeignKeyField object>
```

```

    id = <peewee.PrimaryKeyField object>
    name = <peewee.CharField object>
    provider = <peewee.CharField object>
    value

class xdm.classes.Image(*args, **kwargs)

    DoesNotExist
        alias of ImageDoesNotExist
    delete_instance()
    element = <peewee.ForeignKeyField object>
    id = <peewee.PrimaryKeyField object>
    name = <peewee.TextField object>
    provider = <peewee.TextField object>
    type = <peewee.TextField object>
    url = <peewee.TextField object>

class xdm.classes.Repo(*args, **kwargs)

    DoesNotExist
        alias of RepoDoesNotExist
    id = <peewee.PrimaryKeyField object>
    info_url = <peewee.CharField object>
    name = <peewee.CharField object>
    url = <peewee.CharField object>

class xdm.classes.Location(*args, **kwargs)

    DoesNotExist
        alias of LocationDoesNotExist
    download = <peewee.ForeignKeyField object>
    element = <peewee.ForeignKeyField object>
    extra_data
    id = <peewee.PrimaryKeyField object>
    path = <peewee.TextField object>

```

3.2 Init

```

xdm.init.db()
xdm.init.postDB()
    mostly plugin and common init
xdm.init.preDB(app_path, datadir)

```

```
xm.init.schedule()
```

3.3 GUI Message system

class `xm.message.MessageManager`

Class that manages GUI messages / notifications

createInfo (*message*, *role*='system', *confirm*=None, *deny*=None, *confirmJavascript*=None, *denyJavascript*=None, *uuid*=None)
creates a info message

createWarning (*message*, *role*='system', *confirm*=None, *deny*=None, *confirmJavascript*=None, *denyJavascript*=None, *uuid*=None)
creates a warning message

class `xm.message.SystemMessageManager`

manages system messages mostly used for ajax calls and there messages not thread save! ... it is not checked who/what browser session created the message

3.4 News feed system

class `xm.news.NewsManager`

gets and caches the XDM news feed

cache ()
cache newest feed

3.5 Scheduler

class `xm.scheduler.Scheduler`

Simple manager for recurring tasks

addTask (*action*, *loopdelay*, *initdelay*=0, *name*=None, *description*='')
add a task

Params:

action a function reference

loopdelay delay between calls in seconds

initdelay initial delay in seconds (default = 0)

name name (default = None). will default to action.__name__

description description (default = "")

startAllTasks ()
Start all task threads

stopAllTasks ()
Stop all task threads

3.6 Updater

3.7 Boot structure

```
XDM.py
main() +-
    |
    App() --+
        |
        +- __init__() +-
            |
            |         +-- command line options handling / parsing
            |         +-- xdm.init.preDB(): path and folder setup
            |         +-- xdm.init.db(): database initialisation and migration
            |         +-- xdm.init.postDB(): plugin loading
            |         +-- xdm.init.schedule(): recurring scheduler setup and starting
            |         +-- xdm.init.runTasks(): initial (cleanup) tasks
            |
            +- startWebServer(): (optional)
            |
            |
            +- starting of the API: (optional)
            |
            |
            +- main while loop with KeyboardInterrupt handler
```

3.8 The common object

There is a `xdm.common` object based on *`xdm.Common`* it holds following object references.

Example for PM

```
from xdm import common
for mediaTypeManager in common.PM.getMediaTypeManager():
    print mediaTypeManager.identifier
```

instantiated during import

- MM – `xdm.message.MessageManager` - instance
- SM – `xdm.message.SystemMessageManager` - instance
- NM – `xdm.news.NewsManager` - instance
- SCHEDULER – `xdm.scheduler.Scheduler` - instance

instantiated during boot

- PM – `xdm.plugin.pluginManager.PluginManager` - instance
- SYSTEM – `plugins.system.System.System` - instance
- UPDATER – `xdm.updater.Updater` - instance
- REPOMANAGER – `xdm.plugins.repository.RepoManager` - instance
- APIKEY – str

Statuses available for Elements set during `xdm.init_checkDefaults()` (done in `xdm.init.db()` in `App.__init__()`)

- UNKNOWN – `xdm.classes.Status` - instance # no status
- WANTED – `xdm.classes.Status` - instance # default status
- SNATCHED – `xdm.classes.Status` - instance # well snatched and the downloader is getting it ... so we hope
- DOWNLOADING – `xdm.classes.Status` - instance # its currently downloading woohhooo
- DOWNLOADED – `xdm.classes.Status` - instance # no status thingy
- COMPLETED – `xdm.classes.Status` - instance # downloaded and `pp_success`
- FAILED – `xdm.classes.Status` - instance # download failed
- PP_FAIL – `xdm.classes.Status` - instance # post processing failed
- DELETED – `xdm.classes.Status` - instance # marked as deleted
- IGNORE – `xdm.classes.Status` - instance # ignore this item
- TEMP – `xdm.classes.Status` - instance # temporary item like from a search

Note: Any plugin is prohibited from setting any of these attributes. At time of writing there is no measurement in place to prevent this.

class `xdm.Common`

A class that conveniently holds references to common objects and variables *@DynamicAttrs*

getAllStatus ()

get all available status instances

getCompletedStatuses ()

get statuses that are shown on the completed page

getDownloadTypeExtension (*downloadTypeIdentifier*)

get the (file) of the DownloadType defined by *downloadTypeIdentifier* if none is found will return *txt*

getEveryStatusBut (*notWantedStatuses*)

get all available status instances except statuses in the list *notWantedStatuses*

getHomeStatuses ()

get statuses that are shown on the home/index page

getLocale ()

get the current local string e.g. `en_US` e.g. `de`

this is either the system language / locale or the user set language or the fallback to `en_US`

getStatusByID (*id*)

get a status by the database id

getVersionTuple (*noBuild=False*)

return a tuple of the current version

isThisVersionNewer (*major, minor, revision, build*)

return bool weather the running version is OLDER then the one build by the params

Usage

```
usage: XDM [-h] [-d] [-v] [-D] [-p PIDFILE] [-P PORT] [-n] [-b DATADIR]
          [-c CONFIG] [--noApi] [--apiPort APIPORT] [--noWebServer]
          [--pluginImportDebug] [--profile [PROFILE [PROFILE ...]]]
```

optional arguments:

- h, --help** show this help message and exit
- d, --daemonize** Run the server as a daemon.
- v, --version** Print Version and exit.
- D, --debug** Print debug log to screen.
- p PIDFILE, --pidfile PIDFILE** Store the process id in the given file.
- P PORT, --port PORT** Force webinterface to listen on this port.
- n, --nolaunch** Don't start the browser.
- b DATADIR, --datadir DATADIR** Set the directory for the database.
- c CONFIG, --config CONFIG** Path to config file
- noApi** Disable the api
- apiPort APIPORT** Port the api runs on
- noWebServer** Port the api runs on
- pluginImportDebug** Extra verbosy debug during plugin import is printed.
- profile PROFILE** Wrap a decorated(!) function in a profiler. By default all decorated functions are profiled. Decorate your function with @profileMeMaybe.

Libraries in use

5.1 Backend (Python)

- **CherryPy**: A Minimalist Python Web Framework
- **Requests**: HTTP for Humans
- **pyDes**: This is a pure python implementation of the DES encryption algorithm.
- **profilehooks**: Profiling/tracing wrapper
- **peewee**: a small, expressive orm
- **Jinja2**: Jinja2 is a full featured template engine for Python.
- **pylint**: analyzes Python source code looking for bugs and signs of poor quality
- **astng**: common base representation of python source code for projects such as pychecker, pyreverse, pylint
- **guessit**: a library for guessing information from video files.
- **GitPython**: a python library used to interact with Git repositories.
- **gitdb**: Git Object Database
- **async**: Distribute interdependent tasks to 0 or more threads
- **smmap**: A pure git implementation of a sliding window memory map manager
- **tmdb**: themoviedb.org wrapper for api v3
- **JSONRPClib**: A Python JSON-RPC over HTTP that mirrors xmlrpclib syntax.

5.2 Frontend (css/js/etc)

- **Bootstrap**: Sleek, intuitive, and powerful front-end framework for faster and easier web development.
- **Font Awesome**: The iconic font designed for Bootstrap.
- **jQuery**: is a fast, small, and feature-rich JavaScript library.
- **jQuery UI**: is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library.
- **modernizr**: is a JavaScript library that detects HTML5 and CSS3 features in the user's browser.

- fancyBox: fancyBox is a tool that offers a nice and elegant way to add zooming functionality for images, html content and multi-media on your webpages.
- [Raphaël](#): JavaScript Vector Library
- [noty](#): jquery notification plugin
- [jQuery resize event](#): With jQuery resize event, you can now bind resize event handlers to elements other than window.
- [jQuery Countdown Timer](#): A simple jQuery Countdown Timer with callback

Note: Plugins may use more libraries

C

`corePlugins.system.System`, 12

X

`xdm.api`, 17

`xdm.api.plugins`, 18

`xdm.api.system`, 18

`xdm.classes`, 19

`xdm.init`, 23

`xdm.message`, 24

`xdm.news`, 24

`xdm.plugins.pluginManager`, 7

`xdm.plugins.repository`, 11

`xdm.scheduler`, 24

`xdm.updater`, 25

Symbols

`_getUseConfigsForElementsAsWrapper()`
(`xdm.plugins.bases.Plugin` method), 14

A

`addConfig` (`xdm.plugins.bases.MediaTypeManager` attribute), 5
`addDownload()` (`xdm.plugins.bases.Downloader` method), 4
`addLocation()` (`xdm.classes.Element` method), 20
`addMediaTypeOptions` (`xdm.plugins.bases.Plugin` attribute), 15
`addTask()` (`xdm.scheduler.Scheduler` method), 24
`ancestors` (`xdm.classes.Element` attribute), 21

B

`buildFieldDict()` (`xdm.classes.Element` method), 21
`buildHtml()` (`xdm.classes.Element` method), 21

C

`cache()` (in module `xdm.api.plugins`), 18
`cache()` (`xdm.news.NewsManager` method), 24
`checkForUpdate()` (`xdm.plugins.repository.RepoManager` method), 11
`children` (`xdm.classes.Element` attribute), 21
`clearCache()` (`xdm.classes.Element` method), 21
`clearLowerTreeCache()` (`xdm.classes.Element` method), 21
`clearTreeCache()` (`xdm.classes.Element` method), 21
`clearUpperTreeCache()` (`xdm.classes.Element` method), 21
`Common` (class in `xdm`), 26
`Config` (class in `xdm.classes`), 19
`config_meta` (`xdm.plugins.bases.Plugin` attribute), 15
`config_set` (`xdm.classes.Element` attribute), 21
`config_set` (`xdm.classes.MediaType` attribute), 22
`copy()` (`xdm.classes.Config` method), 19
`copy()` (`xdm.classes.Element` method), 21
`corePlugins.system.System` (module), 12
`createInfo()` (`xdm.message.MessageManager` method), 24

`createWarning()` (`xdm.message.MessageManager` method), 24
`curType()` (`xdm.classes.Config` method), 19

D

`db()` (in module `xdm.init`), 23
`decendants` (`xdm.classes.Element` attribute), 21
`defaultElements` (`xdm.plugins.bases.MediaTypeManager` attribute), 5
`deinstall()` (`xdm.plugins.repository.RepoManager` method), 11
`delete_instance()` (`xdm.classes.Element` method), 21
`delete_instance()` (`xdm.classes.Image` method), 23
`deleteConfig()` (`xdm.classes.Element` method), 21
`deleteDownloads()` (`xdm.classes.Element` method), 21
`deleteFields()` (`xdm.classes.Element` method), 21
`deleteHistory()` (`xdm.classes.Element` method), 21
`deleteImages()` (`xdm.classes.Element` method), 21
`deleteWithChildren()` (`xdm.classes.Element` method), 21
`doCleanUp()` (`xdm.plugins.repository.RepoManager` method), 11
`DoesNotExist` (`xdm.classes.Config` attribute), 19
`DoesNotExist` (`xdm.classes.Download` attribute), 20
`DoesNotExist` (`xdm.classes.Element` attribute), 20
`DoesNotExist` (`xdm.classes.Field` attribute), 22
`DoesNotExist` (`xdm.classes.History` attribute), 20
`DoesNotExist` (`xdm.classes.Image` attribute), 23
`DoesNotExist` (`xdm.classes.Location` attribute), 23
`DoesNotExist` (`xdm.classes.MediaType` attribute), 22
`DoesNotExist` (`xdm.classes.Repo` attribute), 23
`DoesNotExist` (`xdm.classes.Status` attribute), 19
`Download` (class in `xdm.classes`), 20
`download` (`xdm.classes.Location` attribute), 23
`download` (`xdm.plugins.bases.MediaTypeManager` attribute), 5
`download_set` (`xdm.classes.Status` attribute), 19
`download_v0_set` (`xdm.classes.Status` attribute), 19
`download_v1_set` (`xdm.classes.Status` attribute), 19
`Downloader` (class in `xdm.plugins.bases`), 4
`DownloadFilter` (class in `xdm.plugins.bases`), 3
`downloadImage()` (`xdm.classes.Element` method), 21

downloadImages() (xdm.classes.Element method), 21
downloads (xdm.classes.Element attribute), 21
DownloadType (class in xdm.plugins.bases), 3

E

Element (class in xdm.classes), 20
element (xdm.classes.Config attribute), 19
element (xdm.classes.Download attribute), 20
element (xdm.classes.Field attribute), 22
element (xdm.classes.History attribute), 20
element (xdm.classes.Image attribute), 23
element (xdm.classes.Location attribute), 23
element_set (xdm.classes.MediaType attribute), 22
element_set (xdm.classes.Status attribute), 19
elementConfigsFor (xdm.plugins.bases.MediaTypeManager attribute), 5
event (xdm.classes.History attribute), 20
extension (xdm.plugins.bases.DownloadType attribute), 3
external_id (xdm.classes.Download attribute), 20
extra_data (xdm.classes.Download attribute), 20
extra_data (xdm.classes.Location attribute), 23

F

Field (class in xdm.classes), 22
fields (xdm.classes.Element attribute), 21
find_subclasses() (xdm.plugins.pluginManager.PluginManager method), 7

G

getActiveMediaTypes() (in module xdm.api.plugins), 18
getAllStatus() (xdm.Common method), 26
getAnyImage() (xdm.classes.Element method), 21
getCompletedStatuses() (xdm.Common method), 26
getConfig() (xdm.classes.Element method), 21
getConfigHtml() (xdm.plugins.bases.Downloader method), 4
getConfigHtml() (xdm.plugins.bases.DownloadFilter method), 3
getConfigHtml() (xdm.plugins.bases.DownloadType method), 3
getConfigHtml() (xdm.plugins.bases.Indexer method), 4
getConfigHtml() (xdm.plugins.bases.MediaAdder method), 5
getConfigHtml() (xdm.plugins.bases.MediaTypeManager method), 5
getConfigHtml() (xdm.plugins.bases.Notifier method), 6
getConfigHtml() (xdm.plugins.bases.Plugin method), 15
getConfigHtml() (xdm.plugins.bases.PostProcessor method), 6
getConfigHtml() (xdm.plugins.bases.Provider method), 7
getConfigHtml() (xdm.plugins.bases.SearchTermFilter method), 7
getConfigHtml() (xdm.plugins.bases.System method), 7
getConfigs() (xdm.classes.Element method), 21

getDefinedAttr() (xdm.classes.Element method), 21
getDownloadPercentage() (xdm.plugins.bases.Downloader method), 4
getDownloadPercentage() (xdm.plugins.bases.Indexer method), 4
getDownloadTypeExtension() (xdm.Common method), 26
getElementStaus() (xdm.plugins.bases.Downloader method), 4
getEveryStatusBut() (xdm.Common method), 26
getField() (xdm.classes.Element method), 21
getHomeStatuses() (xdm.Common method), 26
getIdentifier() (xdm.classes.Element method), 21
getImage() (xdm.classes.Element method), 21
getImageNames() (xdm.classes.Element method), 21
getLatestRss() (xdm.plugins.bases.Indexer method), 4
getLocale() (xdm.Common method), 26
getName() (xdm.classes.Element method), 21
getReleaseDate() (xdm.classes.Element method), 21
getSearchTemplate() (xdm.classes.Element method), 21
getSearchTerms() (xdm.classes.Element method), 21
getStatusByID() (xdm.Common method), 26
getTemplate() (xdm.classes.Element method), 21
getVersionTuple() (xdm.Common method), 26
getWhereField() (xdm.classes.Element class method), 21
getXDMID() (xdm.classes.Element method), 22

H

hasUpdate() (xdm.plugins.repository.RepoManager method), 11
hidden (xdm.classes.Status attribute), 19
History (class in xdm.classes), 20
history_set (xdm.classes.Element attribute), 22

I

id (xdm.classes.Config attribute), 19
id (xdm.classes.Download attribute), 20
id (xdm.classes.Element attribute), 22
id (xdm.classes.Field attribute), 22
id (xdm.classes.History attribute), 20
id (xdm.classes.Image attribute), 23
id (xdm.classes.Location attribute), 23
id (xdm.classes.MediaType attribute), 22
id (xdm.classes.Repo attribute), 23
id (xdm.classes.Status attribute), 19
identifier (xdm.classes.MediaType attribute), 22
identifier (xdm.plugins.bases.DownloadType attribute), 3
identifier (xdm.plugins.bases.MediaTypeManager attribute), 6
Image (class in xdm.classes), 23
images (xdm.classes.Element attribute), 22
Indexer (class in xdm.plugins.bases), 4
indexer (xdm.classes.Download attribute), 20

indexer_instance (xdm.classes.Download attribute), 20
 info_url (xdm.classes.Repo attribute), 23
 install() (xdm.plugins.repository.RepoManager method), 11
 instance (xdm.classes.Config attribute), 19
 isAncestorOf() (xdm.classes.Element method), 22
 isDescendantOf() (xdm.classes.Element method), 22
 isReleaseDateInPast() (xdm.classes.Element method), 22
 isThisVersionNewer() (xdm.Common method), 26

L

leaf (xdm.classes.Element attribute), 22
 listMethods() (in module xdm.api.system), 18
 Location (class in xdm.classes), 23
 locations (xdm.classes.Download attribute), 20
 locations (xdm.classes.Element attribute), 22

M

manager (xdm.classes.Element attribute), 22
 manager (xdm.classes.MediaType attribute), 22
 MediaAdder (class in xdm.plugins.bases), 5
 MediaAdder.Media (class in xdm.plugins.bases), 5
 MediaType (class in xdm.classes), 22
 mediaType (xdm.classes.Config attribute), 19
 mediaType (xdm.classes.Element attribute), 22
 MediaTypeManager (class in xdm.plugins.bases), 5
 MessageManager (class in xdm.message), 24
 methodHelp() (in module xdm.api.system), 18
 methodSignature() (in module xdm.api.system), 18
 module (xdm.classes.Config attribute), 19

N

name (xdm.classes.Config attribute), 19
 name (xdm.classes.Download attribute), 20
 name (xdm.classes.Field attribute), 23
 name (xdm.classes.Image attribute), 23
 name (xdm.classes.MediaType attribute), 22
 name (xdm.classes.Repo attribute), 23
 name (xdm.classes.Status attribute), 19
 new_obj (xdm.classes.History attribute), 20
 NewsManager (class in xdm.news), 24
 Notifier (class in xdm.plugins.bases), 6

O

obj_class (xdm.classes.History attribute), 20
 obj_id (xdm.classes.History attribute), 20
 obj_type (xdm.classes.History attribute), 20
 old_obj (xdm.classes.History attribute), 20
 order (xdm.plugins.bases.MediaTypeManager attribute), 6
 orderFields (xdm.classes.Element attribute), 22
 orderFieldValues (xdm.classes.Element attribute), 22
 orderReverse (xdm.classes.Element attribute), 22

P

paint() (xdm.classes.Element method), 22
 parent (xdm.classes.Element attribute), 22
 path (xdm.classes.Location attribute), 23
 ping() (in module xdm.api), 17
 Plugin (class in xdm.plugins.bases), 14
 PluginManager (class in xdm.plugins.pluginManager), 7
 postDB() (in module xdm.init), 23
 PostProcessor (class in xdm.plugins.bases), 6
 postProcessPath() (xdm.plugins.bases.PostProcessor method), 6
 pp_log (xdm.classes.Download attribute), 20
 preDB() (in module xdm.init), 23
 Provider (class in xdm.plugins.bases), 6
 provider (xdm.classes.Field attribute), 23
 provider (xdm.classes.Image attribute), 23
 pylintScoreError (xdm.plugins.pluginManager.PluginManager attribute), 7
 pylintScoreWarning (xdm.plugins.pluginManager.PluginManager attribute), 8

R

reboot() (in module xdm.api.system), 18
 Repo (class in xdm.classes), 23
 RepoManager (class in xdm.plugins.repository), 11
 root (xdm.plugins.bases.MediaAdder.Media attribute), 5
 runShedule() (xdm.plugins.bases.MediaAdder method), 5

S

save() (xdm.classes.Element method), 22
 saveTemp() (xdm.classes.Element method), 22
 schedule() (in module xdm.init), 23
 Scheduler (class in xdm.scheduler), 24
 screenName (xdm.classes.Status attribute), 19
 search() (xdm.plugins.bases.MediaTypeManager method), 6
 searchForElement() (xdm.plugins.bases.Indexer method), 4
 searchForElement() (xdm.plugins.bases.Provider method), 7
 SearchTermFilter (class in xdm.plugins.bases), 7
 section (xdm.classes.Config attribute), 19
 setField() (xdm.classes.Element method), 22
 shutdown() (in module xdm.api.system), 18
 single (xdm.plugins.bases.Plugin attribute), 15
 size (xdm.classes.Download attribute), 20
 startAllTasks() (xdm.scheduler.Scheduler method), 24
 Status (class in xdm.classes), 19
 status (xdm.classes.Download attribute), 20
 status (xdm.classes.Element attribute), 22
 stopAllTasks() (xdm.scheduler.Scheduler method), 24
 System (class in xdm.plugins.bases), 7
 SystemMessageManager (class in xdm.message), 24

T

time (xdm.classes.History attribute), 20
type (xdm.classes.Config attribute), 19
type (xdm.classes.Download attribute), 20
type (xdm.classes.Element attribute), 22
type (xdm.classes.Image attribute), 23

U

url (xdm.classes.Download attribute), 20
url (xdm.classes.Image attribute), 23
url (xdm.classes.Repo attribute), 23
useConfigsForElementsAs (xdm.plugins.bases.Plugin attribute), 15

V

value (xdm.classes.Config attribute), 19
value (xdm.classes.Field attribute), 23
version (xdm.plugins.bases.Plugin attribute), 15
version() (in module xdm.api), 17

W

where_extra_data() (xdm.classes.Download class method), 20

X

xdm.api (module), 17
xdm.api.plugins (module), 18
xdm.api.system (module), 18
xdm.classes (module), 19
xdm.init (module), 23
xdm.message (module), 24
xdm.news (module), 24
xdm.plugins.pluginManager (module), 7
xdm.plugins.repository (module), 11
xdm.scheduler (module), 24
xdm.updater (module), 25
xdm_version (xdm.plugins.bases.Plugin attribute), 15
XDMID (xdm.classes.Element attribute), 20