pyexistdb Documentation

Release 1.0.1

The Research Software Company

Jun 05, 2018

Contents

1 Contents **3**

1.1 pyexistdb – Store and retrieve data in an eXist database . . . 3

1.1.1 Direct database access . . . 3

1.1.2 Object-based searching . . . 11

1.1.3 Django tie-ins for pyexistdb . . . 14

1.2 Change & Version Information . . . 17

1.2.1 1.0.1 . . . 17

1.2.2 1.0.0 . . . 17

1.2.3 0.21.1 . . . 17

1.2.4 0.21 . . . 17

1.2.5 0.20 . . . 18

1.2.6 0.19.2 . . . 18

1.2.7 0.19.1 . . . 18

1.2.8 0.19 . . . 18

1.2.9 0.18 . . . 19

1.2.10 0.17 . . . 19

1.2.11 0.16 . . . 19

1.2.12 0.15.2 . . . 19

1.2.13 0.15.1 - Bugfix Release . . . 19

1.2.14 0.15.0 - Initial Release . . . 19

2 Indices and tables **21**

Python Module Index **23**

Pyexistdb is a library for accessing, querying, and updating an eXist-db XML database using idiomatic Python. *ExistDB* provides basic read-write access. pyexistdb doesn't require Django, but when they're used together, developers can define *XmlModel* subclasses to automate XQuery searching of the eXist database based on the model's XPath fields. Even without Django configuration, developers can utilize this automatic XQuery generation by constructing a *QuerySet* referencing an `XmlObject` subclass and a database.

CHAPTER 1

Contents

1.1 pyexistdb – Store and retrieve data in an eXist database

Interact with eXist-db XML databases.

This package provides classes to ease interaction with eXist XML databases. It contains the following modules:

- *pyexistdb.db* – Connect to the database and query
- *pyexistdb.query* – Query `XmlObject` models from eXist with semantics like a Django QuerySet

1.1.1 Direct database access

Connect to an eXist XML database and query it.

This module provides *ExistDB* and related classes for connecting to an eXist-db database and executing XQuery queries against it.

When used with Django, *ExistDB* can pull configuration settings directly from Django settings. If you create an instance of *ExistDB* without specifying a server url, it will attempt to configure an eXist database based on Django settings, using the configuration names documented below.

Projects that use this module should include the following settings in their `settings.py`:

```
# Exist DB Settings
EXISTDB_SERVER_USER = 'user'
EXISTDB_SERVER_PASSWORD = 'password'
EXISTDB_SERVER_URL = "http://megaserver.example.com:8042/exist"
EXISTDB_ROOT_COLLECTION = "/sample_collection"
```

To configure a timeout for most eXist connections, specify the desired time in seconds as `EXISTDB_TIMEOUT`; if none is specified, the global default socket timeout will be used.

Note: Any configured `EXISTDB_TIMEOUT` will be ignored by the **existdb** management command, since reindexing a large collection could take significantly longer than a normal timeout would allow for.

If you are using an eXist index configuration file, you can add another setting to specify your configuration file:

```
EXISTDB_INDEX_CONFIGFILE = "/path/to/my/exist_index.xconf"
```

This will allow you to use the `existdb` management command to manage your index configuration file in eXist.

If you wish to specify options for fulltext queries, you can set a dictionary of options like this:

```
EXISTDB_FULLTEXT_OPTIONS = {'default-operator': 'and'}
```

Note: Python `xmlrpclib` does not support extended types, some of which are used in eXist returns. This does not currently affect the functionality exposed within *ExistDB*, but may cause issues if you use the `ExistDB.server` XML-RPC connection directly for other available eXist XML-RPC methods. If you do make use of those, you may want to enable XML-RPC patching to handle the return types:

```
from pyexistdb import patch
patch.request_patching(patch.XMLRpcLibPatch)
```

—

If you are writing unit tests against code that uses *pyexistdb*, you may want to take advantage of *pyexistdb.testutil.TestCase* for loading fixture data to a test eXist-db collection, and `pyexistdb.testutil.ExistDBTestSuiteRunner`, which has logic to set up and switch configurations between a development and test collections in eXist.

class pyexistdb.db.**ExistDB**(*server_url*[, *resultType*[, *encoding*[, *verbose*]]])

Connect to an eXist database, and manipulate and query it.

Construction doesn't initiate server communication, only store information about where the server is, to be used in later communications.

Parameters

- **server_url** – The eXist server URL. New syntax (as of 0.20) expects primary eXist url and *not* the `/xmlrpc` endpoint; for backwards compatibility, urls that include *'/xmlrpc'* are still handled, and will be parsed to set exist server path as well as username and password if specified. Note that username and password parameters take precedence over username and password in the server url if both are specified.
- **username** – exist username, if any
- **password** – exist user password, if any
- **resultType** – The class to use for returning *query()* results; defaults to *QueryResult*
- **encoding** – The encoding used to communicate with the server; defaults to "UTF-8"
- **verbose** – When True, print XML-RPC debugging messages to stdout
- **timeout** – Specify a timeout for xmlrpc connection requests. If not specified, the global default socket timeout value will be used.

- **keep_alive** – Optional parameter, to disable requests built-in session handling; can also be configured in django settings with EXISTDB_SESSION_KEEP_ALIVE

getDocument(*name*)
Retrieve a document from the database.

Parameters **name** – database document path to retrieve

Return type string contents of the document

createCollection(*collection_name*[, *overwrite*])
Create a new collection in the database.

Parameters

- **collection_name** – string name of collection
- **overwrite** – overwrite existing document?

Return type boolean indicating success

removeCollection(*collection_name*)
Remove the named collection from the database.

Parameters **collection_name** – string name of collection

Return type boolean indicating success

hasCollection(*collection_name*)
Check if a collection exists.

Parameters **collection_name** – string name of collection

Return type boolean

load(*xml*, *path*[, *overwrite*])
Insert or overwrite a document in the database.

Note: This method will automatically overwrite existing content at the same path without notice. This is a change from versions prior to 0.20.

Parameters

- **xml** – string or file object with the document contents
- **path** – destination location in the database

Return type boolean indicating success

query(*xquery*[, *start*[, *how_many*]])
Execute an XQuery query, returning the results directly.

Parameters

- **xquery** – a string XQuery query
- **start** – first index to return (1-based)
- **how_many** – maximum number of items to return
- **cache** – boolean, to cache a query and return a session id (optional)
- **session** – session id, to retrieve a cached session (optional)
- **release** – session id to be released (optional)

Return type the resultType specified at the creation of this ExistDB; defaults to *QueryResult*.

executeQuery(*xquery*)
Execute an XQuery query, returning a server-provided result handle.

Parameters **xquery** – a string XQuery query

Return type an integer handle identifying the query result for future calls

querySummary(*result_id*)
Retrieve results summary from a past query.

Parameters **result_id** – an integer handle returned by *executeQuery()*

Return type a dict describing the results

The returned dict has four fields:

- *queryTime*: processing time in milliseconds
- *hits*: number of hits in the result set
- *documents*: a list of lists. Each identifies a document and takes the form [*doc_id*, *doc_name*, *hits*], where:
  - *doc_id*: an internal integer identifier for the document
  - *doc_name*: the name of the document as a string
  - *hits*: the number of hits within that document
- ***doctype*: a list of lists. Each contains a doctype public** identifier and the number of hits found for this doctype.

getHits(*result_id*)
Get the number of hits in a query result.

Parameters **result_id** – an integer handle returned by *executeQuery()*

Return type integer representing the number of hits

retrieve(*result_id*, *position*)
Retrieve a single result fragment.

Parameters

- **result_id** – an integer handle returned by *executeQuery()*
- **position** – the result index to return
- **highlight** – enable search term highlighting in result; optional, defaults to False

Returns the query result item as a string or XMLRPC Binary

Return type string | xmlrpc.client.Binary

releaseQueryResult(*result_id*)
Release a result set handle in the server.

Parameters **result_id** – an integer handle returned by *executeQuery()*

createCollection(*collection_name*, *overwrite=False*)
Create a new collection in the database.

Parameters

- **collection_name** – string name of collection

- **overwrite** – overwrite existing document?

Return type boolean indicating success

create_account (*username*, *password*, *groups*)
Create a user account; returns true if the user was created, false if the user already exists. Any other exist exception is re-raised.

create_group (*group*)
Create a group; returns true if the group was created, false if the group already exists. Any other exist exception is re-raised.

describeDocument (*document_path*)
Return information about a document in eXist. Includes name, owner, group, created date, permissions, mime-type, type, content-length. Returns an empty dictionary if document is not found.

Parameters **document_path** – string full path to document in eXist

Return type dictionary

executeQuery (*xquery*)
Execute an XQuery query, returning a server-provided result handle.

Parameters **xquery** – a string XQuery query

Return type an integer handle identifying the query result for future calls

getCollectionDescription (*collection_name*)
Retrieve information about a collection.

Parameters **collection_name** – string name of collection

Return type boolean

getDoc (*name*)
Alias for *getDocument()*.

getDocument (*name*)
Retrieve a document from the database.

Parameters **name** – database document path to retrieve

Return type string contents of the document

getHits (*result_id*)
Get the number of hits in a query result.

Parameters **result_id** – an integer handle returned by *executeQuery()*

Return type integer representing the number of hits

getPermissions (*resource*)
Retrieve permissions for a resource in eXist.

Parameters **resource** – full path to a collection or document in eXist

Return type ExistPermissions

hasCollection (*collection_name*)
Check if a collection exists.

Parameters **collection_name** – string name of collection

Return type boolean

hasCollectionIndex(*collection_name*)
Check if the specified collection has an index configuration in eXist.

Note: according to eXist documentation, index config file does not *have* to be named *collection.xconf* for reasons of backward compatibility. This function assumes that the recommended naming conventions are followed.

Parameters **collection** – name of the collection with an index to be removed

Return type boolean indicating collection index is present

hasDocument(*document_path*)
Check if a document is present in eXist.

Parameters **document_path** – string full path to document in eXist

Return type boolean

load(*xml*, *path*)
Insert or overwrite a document in the database.

Note: This method will automatically overwrite existing content at the same path without notice. This is a change from versions prior to 0.20.

Parameters

- **xml** – string or file object with the document contents
- **path** – destination location in the database

Return type boolean indicating success

loadCollectionIndex(*collection_name*, *index*)
Load an index configuration for the specified collection. Creates the eXist system config collection if it is not already there, and loads the specified index config file, as per eXist collection and index naming conventions.

Parameters

- **collection_name** – name of the collection to be indexed
- **index** – string or file object with the document contents (as used by *load()*)

Return type boolean indicating success

moveDocument(*from_collection*, *to_collection*, *document*)
Move a document in eXist from one collection to another.

Parameters

- **from_collection** – collection where the document currently exists
- **to_collection** – collection where the document should be moved
- **document** – name of the document in eXist

Return type boolean

query(*xquery=None*, *start=1*, *how_many=10*, *cache=False*, *session=None*, *release=None*, *result_type=None*)
Execute an XQuery query, returning the results directly.

Parameters

- **xquery** – a string XQuery query
- **start** – first index to return (1-based)
- **how_many** – maximum number of items to return
- **cache** – boolean, to cache a query and return a session id (optional)
- **session** – session id, to retrieve a cached session (optional)
- **release** – session id to be released (optional)

Return type the resultType specified at the creation of this ExistDB; defaults to *QueryResult*.

querySummary(*result_id*)
Retrieve results summary from a past query.

Parameters **result_id** – an integer handle returned by *executeQuery()*

Return type a dict describing the results

The returned dict has four fields:

- *queryTime*: processing time in milliseconds
- *hits*: number of hits in the result set
- *documents*: a list of lists. Each identifies a document and takes the form [*doc_id*, *doc_name*, *hits*], where:
  - *doc_id*: an internal integer identifier for the document
  - *doc_name*: the name of the document as a string
  - *hits*: the number of hits within that document
- ***doctype*: a list of lists. Each contains a doctype public** identifier and the number of hits found for this doctype.

reindexCollection(*collection_name*)
Reindex a collection. Reindex will fail if the eXist user does not have the correct permissions within eXist (must be a member of the DBA group).

Parameters **collection_name** – string name of collection

Return type boolean success

releaseQueryResult(*result_id*)
Release a result set handle in the server.

Parameters **result_id** – an integer handle returned by *executeQuery()*

removeCollection(*collection_name*)
Remove the named collection from the database.

Parameters **collection_name** – string name of collection

Return type boolean indicating success

removeCollectionIndex(*collection_name*)
Remove index configuration for the specified collection. If index collection has no documents or subcollections after the index file is removed, the configuration collection will also be removed.

Parameters **collection** – name of the collection with an index to be removed

Return type boolean indicating success

removeDocument(*name*)
Remove a document from the database.

Parameters **name** – full eXist path to the database document to be removed

Return type boolean indicating success

retrieve(*result_id*, *position*, *highlight=False*, ***options*)
Retrieve a single result fragment.

Parameters

- **result_id** – an integer handle returned by *executeQuery()*
- **position** – the result index to return
- **highlight** – enable search term highlighting in result; optional, defaults to False

Returns the query result item as a string or XMLRPC Binary

Return type string | xmlrpc.client.Binary

retrieve_text(*result_id*, *position*, *highlight=False*, ***options*)
Retrieve a single result fragment, making sure it is returned as text.

Parameters

- **result_id** – an integer handle returned by *executeQuery()*
- **position** – the result index to return
- **highlight** – enable search term highlighting in result; optional, defaults to False

Returns the query result item as a string

Return type string

This function fixes an inconvenience with the original retrieve function. In some cases eXist-db returns base64 encoded strings, and xmlrpc thinks the response is binary, leaving the decoding to the caller. retrieve_text always decodes binaries to strings based on the default encoding.

setPermissions(*resource*, *permissions*)
Set permissions on a resource in eXist.

Parameters

- **resource** – full path to a collection or document in eXist
- **permissions** – int or string permissions statement

class pyexistdb.db.**QueryResult**(*node=None*, *context=None*, ***kwargs*)
The results of an eXist XQuery query

count
The number of results returned in this chunk

hits
The total number of hits found by the search

results
The result documents themselves as nodes, starting at `start` and containing *count* members

exception pyexistdb.db.**ExistDBException**
A handy wrapper for all errors returned by the eXist server.

1.1.2 Object-based searching

Provide a prettier, more Pythonic approach to eXist-db access.

This module provides *QuerySet* modeled after Django QuerySet objects. It's not dependent on Django at all, but it aims to function as a stand-in replacement in any context that expects one.

class pyexistdb.query.**QuerySet**(*model=None*, *xpath=None*, *using=None*, *collection=None*, *xquery=None*, *fulltext_options=None*)

Lazy eXist database lookup for a set of objects.

Parameters

- **model** – the type of object to return from __getitem__(). If set, the resulting xml nodes will be wrapped in objects of this type. Some methods, like *filter()* and *only()* only make sense if this is set. While this argument can be any callable object, it is typically a subclass of XmlObject.
- **xpath** – an XPath expression where this *QuerySet* will begin filtering. Typically this is left out, beginning with an unfiltered collection: Filtering is then added with *filter()*.
- **using** – The *ExistDB* to query against.
- **collection** – If set, search only within a particular eXist-db collection. Otherwise search all collections.
- **xquery** – Override the entire Xquery object used for internal query serialization. Most code will leave this unset, which uses a default Xquery.
- **fulltext_options** – optional dictionary of fulltext options to be used as settings for any full-text queries. See http://demo.exist-db.org/lucene.xml#N1047C for available options. Requires a version of eXist that supports this feature.

all()

Return all results.

This method returns an identical copy of the QuerySet.

also(**fields*)

Return additional data in results.

Parameters **fields** – names of fields in the QuerySet's model

This method returns an updated copy of the QuerySet: It does not modify the original. When results are returned from the updated copy, they will contain the specified additional fields.

For special fields available, see *only()*.

For performance considerations, see note on *only()*.

also_raw(***fields*)

Return an additional field by raw xpath. Similar to (and can be combined with) *also()*, but xpath is not pulled from the model. Use this when you want to retrieve a field with a different xpath than the one configured in your model. See Xquery.return_only() for details on specifying xpaths in raw mode.

Parameters

- **fields** – field name and xpath in keyword-args notation. If **field** is the name of a field on the associated model, the result of the raw xpath should be accessible on the return object as the normal property.
- **xpath** – xpath for retrieving the specified field

Can be combined with *also()*.

Example usage:

```
qs.also_raw(field_matches='count(util:expand(%(xq_var)s//field)//exist:match)
↪')
```

count()
Return the cached query hit count, executing the query first if it has not yet executed.

distinct()
Return distinct results.

This method returns an updated copy of the QuerySet: It does not modify the original. When results are returned from the updated copy, they will contain only distinct results.

exclude(***kwargs*)
Filter the QuerySet to return a subset of the documents that do **not** contain any of the filters. Uses the same syntax and allows for the same filters as *filter()*.

filter(*combine='AND'*, ***kwargs*)
Filter the QuerySet to return a subset of the documents.

Arguments take the form `lookuptype` or `field__lookuptype`, where `field` is the name of a field in the QuerySet's `model` and `lookuptype` is one of:

- `exact` – The field or object matches the argument value.
- `contains` – The field or object contains the argument value.
- `startswith` – The field or object starts with the argument value.
- `fulltext_terms` – the field or object contains any of the the argument terms anywhere in the full text; requires a properly configured lucene index. By default, highlighting is enabled when this filter is used. To turn it off, specify an additional filter of highlight=False. Recommend using `fulltext_score` for ordering, in return fields.
- `highlight` - highlight search terms; when used with `fulltext_terms`, should be specified as a boolean (enabled by default); when used separately, takes a string using the same search format as `fulltext_terms`, but content will be returned even if it does not include the search terms. Requires a properly configured lucene index.
- `in` - field or object is present in a list of values
- **`exists` - field or object is or is not present in the document;** if True, field must be present; if False, must not be present.
- `document_path` - restrict the query to a single document; this must be a document path as returned by eXist, with full db path
- **`gt`, `gte`, `lt`, `lte` - greater than, greater than or equal,** less than, less than or equal

Field may be in the format of field__subfield when field is an NodeField or NodeListField and subfield is a configured element on that object.

Field may also be one of the prefined 'special' fields; see *only()* for the list of fields.

Any number of these filter arguments may be passed. This method returns an updated copy of the QuerySet: It does not modify the original.

Parameters **`combine`** – optional; specify how the filters should be combined. Defaults to `AND`; also supports `OR`.

get(***kwargs*)
Get a single result identified by filter arguments.

Takes any number of *filter()* arguments. Unlike *filter()*, though, this method returns exactly one item. If the filter expressions match no items, or if they match more than one, this method throws an exception.

Raises a `pyexistdb.exceptions.DoesNotExist` exception if no matches are found; raises a `pyexistdb.exceptions.ReturnedMultiple` exception if more than one match is found.

getDocument(*docname*)
Get a single document from the server by filename.

only(**fields*)
Limit results to include only specified fields.

Parameters **fields** – names of fields in the QuerySet's `model`

This method returns an updated copy of the QuerySet: It does not modify the original. When results are returned from the updated copy, they will contain only the specified fields.

Special fields available:

- `fulltext_score` - lucene query; should only be used when a fulltext query has been used
- `document_name`, `collection_name` - document or collection name where xml content is stored in eXist
- `hash` - generate and return a SHA-1 checksum of the root element being queried
- `last_modified` - `DateTimeField` for the date the document the xml element belongs to was last modified

NOTE: Be aware that this will result in an XQuery with a constructed return. For large queries, this may have a significant impact on performance. For more details, see http://exist.sourceforge.net/tuning.html#N103A2 .

only_raw(***fields*)
Limit results to include only specified fields, and return the specified field by xpath. Similar to (and can be combined with) *only()*. See `Xquery.return_only()` for details on specifying xpaths in raw mode.

See *also_raw()* for more details and usage example.

or_filter(***kwargs*)
Filter the QuerySet to return a subset of the documents, but combine the filters with OR instead of AND. Uses the same syntax and allows for the same filters as *filter()* with the exception that currently predefined special fields (see *only()*) are not supported.

order_by(*field*)
Order results returned according to a specified field. By default, all sorting is case-sensitive and in ascending order.

Parameters **field** – the name (a string) of a field in the QuerySet's `model`. If the field is prefixed with '-', results will be sorted in descending order. If the field is prefixed with '~', results will use a case-insensitive sort. The flags '-' and '~' may be combined in any order.

Example usage:

```
queryset.filter(fulltext_terms='foo').order_by('-fulltext_score')
queryset.order_by('~name')
```

This method returns an updated copy of the QuerySet. It does not modify the original.

order_by_raw(*xpath*, *ascending=True*)
Order results returned by a raw XPath.

Parameters **xpath** – the xpath to be used

This method returns an updated copy of the QuerySet. It does not modify the original.

Example usage:

```
qs.order_by_raw('min(%(xq_var)s//date/string())')
```

query_result_type
Custom query result return type used to access a batch of results wrapped in an exist result as returned by the REST API. Extends *pyexistdb.db.QueryResult* to add an item-level result mapping based, using *return_type* if appropriate.

reset()
Reset filters and cached results on the QuerySet.

This modifies the current query set, removing all filters, and resetting cached results.

result_id
Return the cached server result id, executing the query first if it has not yet executed.

return_type
Return type that will be used for initializing results returned from eXist queries. Either the subclass of XmlObject passed in to the constructor as model, or, if *only()* or *also()* has been used, a dynamically created instance of XmlObject with the xpaths modified based on the constructed xml return.

using(*collection*)
Specify the eXist collection to be queried.

If you are using an *pyexistdb.models.XmlModel* to generate queries against an eXist collection other than the one defined in settings.EXISTDB_ROOT_COLLECTION, you should use this function.

class pyexistdb.query.**XmlQuery**(*node=None*, *context=None*, ***kwargs*)
XmlObject class to allow describing queries in xml.

1.1.3 Django tie-ins for pyexistdb

class pyexistdb.manager.**Manager**(*xpath*)
Connect an *XmlModel* to an *ExistDB* for easy querying.

Typically each *XmlModel* will have one or more Manager members. Like Django Manager objects these offer a convenient way to access model-based queries. Like Django Manager objects, developers can derive a child class and override *get_query_set()* to modify the default QuerySet. Unlike Django, this implementation does not currently provide a default Manager for every XmlModel.

Developers should consult *pyexistdb.query.QuerySet* for a complete list of its methods. Manager directly exposes these methods, forwarding them to the QuerySet returned by its own *get_query_set()*.

get_query_set()
Get the default pyexistdb.db.QuerySet returned by this Manager. Typically this returns a QuerySet based on the Manager's *xpath*, evaluated in the settings.EXISTDB_ROOT_COLLECTION on a default *pyexistdb.db.ExistDB*.

This is a convenient point for developers to customize an object's managers. Deriving a child class from Manager and overriding or extending this method is a handy way to create custom queries accessible from an XmlModel.

class pyexistdb.models.**XmlModel**(*node=None*, *context=None*, ***kwargs*)

An XmlObject in an *pyexistdb.db.ExistDB*.

XmlModel is derived from XmlObject and thus has access to all the field logic provided by that class. Additionally, since XmlModel objects are stored in an *ExistDB*, they can define *Manager* members for easy access to stored models.

Two use cases are particularly common. First, a developer may wish to use an XmlModel just like an XmlObject, but with the added semantics of being eXist-backed:

```
class StoredWidget(XmlModel):
    name = StringField("name")
    quantity = IntegerField("quantity")
    top_customers = StringListField("(order[@status='active']/customer)[position()
↪<5]/name")
    objects = Manager("//widget")
```

Second, if an XmlObject is defined elsewhere, an application developer might simply expose *ExistDB* backed objects:

```
class StoredThingie(XmlModel, Thingie):
    objects = Manager("/thingie")
```

Of course, some applications ask for mixing these two cases, extending an existing XmlObject while adding application-specific fields:

```
class CustomThingie(XmlModel, Thingie):
    best_foobar = StringField("qux/fnord[@application='myapp']/name")
    custom_detail = IntegerField("detail/@level")
    objects = Manager("/thingie")
```

In addition to the fields inherited from XmlObject, XmlModel objects have one extra field:

_managers

A dictionary mapping manager names to *Manager* objects. This dictionary includes all of the managers defined on the model itself, though it does not currently include managers inherited from the model's parents.

Custom Template Tags

Custom template filter for converting eXist highlight match tags to HTML.

To use, add `{% load existdb %}` to your template and then use the exist_matches filter when you output data, e.g.:

```
{{ poem.title|exist_matches }}
```

You should add CSS for span.exist-match to style it for search-term highlighting.

The *exist_matches()* template tag expects to be given an instance of an XmlObject (either a top-level object or a sub-object mapped via NodeField or NodeListField).

pyexistdb.templatetags.existdb.**exist_matches**(*value*, *autoescape=None*)

Custom django template filter to convert structured fields in xml returned by the eXist database to HTML. XmlObject values are recursively processed, escaping text nodes and converting <exist:match> tags to tags. Other values are simply converted to unicode and escaped.

Parameters **value** – XmlObject instance

Currently performs the following conversions:

- `<exist:match>` is converted to `<span class="exist-match">`
- other elements are stripped
- text nodes are HTML escaped where the template context calls for it

`pyexistdb` Management commands

The following management command will be available when you include *pyexistdb* in your django `INSTALLED_APPS` and rely on the existdb settings described above.

For more details on these commands, use `manage.py <command> help`

- **existdb** - update, remove, and show information about the index configuration for a collection index; reindex the configured collection based on that index configuration

`testutil` Unit Test utilities

pyexistdb.testutil provides utilities for writing and running tests against code that makes use of *pyexistdb*. This module includes a customization of `django.test.TestCase` with eXist-db fixture handling, and custom test suite runners with Fedora environment setup / teardown for all tests.

To use, configure as test runner in your Django settings:

```
TEST_RUNNER = 'pyexistdb.testutil.ExistDBTextTestSuiteRunner'
```

When `xmlrunner` is available, xmlrunner variants are also available. To use this test runner, configure your test runner as follows:

```
TEST_RUNNER = 'pyexistdb.testutil.ExistDBXmlTestSuiteRunner'
```

The xml variant honors the same django settings that the xmlrunner django testrunner does (TEST_OUTPUT_DIR, TEST_OUTPUT_VERBOSE, and TEST_OUTPUT_DESCRIPTIONS).

Any *ExistDB* instances created after the test suite starts will automatically connect to the test collection.

If you are using `nose` or `django-nose`, you should use the *ExistDBSetUp* plugin to set up the test eXist database. With `django-nose`, you should add `pyexistdb.testutil.ExistDBSetUp` to **NOSE_PLUGINS** and `--with-existdbsetup` to **NOSE_ARGS** to ensure the plugin is automatically enabled.

---

class pyexistdb.testutil.**ExistDBSetUp**

help()
Return help for this plugin. This will be output as the help section of the –with-$name option that enables the plugin.

class pyexistdb.testutil.**ExistDBTestWrapper**
A context manager that replaces the Django eXist-db configuration with a newly-created temporary test configuration inside the block, returning to the original configuration and deleting the test one when the block exits.

class pyexistdb.testutil.**TestCase**(*methodName='runTest'*)
Customization of `django.test.TestCase`

If TestCase instance has an attribute named `exist_fixtures`, the specified fixtures will be loaded to eXist before the tests run.

The `exist_fixtures` attribute should be a dictionary with information about fixtures to load to eXist. Currently supported options:

- *index* - path to an eXist index configuration file; will be loaded before any other fixture files, and removed in fixture teardown
- *directory* - path to a fixture directory; all .xml files in the directory will be loaded to eXist
- *files* - list of files to be loaded (filenames should include path)

assertPattern(*regex*, *text*, *msg_prefix=''*)
Assert that a string matches a regular expression (regex compiled as multiline). Allows for more flexible matching than the django assertContains.

pyexistdb.testutil.**alternate_test_existdb**
alias of *pyexistdb.testutil.ExistDBTestWrapper*

debug_panel Debug Toolbar Panel

1.2 Change & Version Information

The following is a summary of changes and improvements to *pyexistdb*. New features in each version should be listed, with any necessary information about installation or upgrade notes.

1.2.1 1.0.1

- Circumvent a small bug in eXist-db, where XML-RPC requests do not specify the charset in the content-type header, causing UTF-8 encoded documents to be decoded improperly.
- Added the *db.retrieve_text* method that always returns a string. *db.retrieve* may return a Binary object if eXist-db decides to use base64 for encoding some fields.

1.2.2 1.0.0

Took `eulexistdb` and made it compatible with Python 3. This is the first version that's called *pyexistdb*.

1.2.3 0.21.1

- Clean up reference to old overwrite parameter no longer used in *db.load* method; explicitly test load method #12
- Code cleanup based on landscape.io scan.

1.2.4 0.21

- Removed unused kwargs from db.ExistDB init method #1
- db.ExistDB `create_group` and `create_account` methods now re-raise unexpected errors #2
- Improved timeout handling; fixes timeouts on REST API requests #3
- bugfix: make Django settings actually optional
- Require eulxml 1.1.2 to handle duplicate `xml:id` attributes included in a single exist result. (Duplicate id test case contributed by @lddubeau in PR #5)

- Add opt-in patch for extended type handling in xmlrpc. Contributed by @lddubeau in PR #6, resolves #4
- Removed `overwrite` option from *eulexistdb.ExistDB.load* (no longer applicable under the REST API, and misleading) #9
- Improved django-debug-toolbar integration. #7, #8
- Updated *Existdb.DB* initialization parameters to restore support for xmlrpc-style urls with username and password used in previous versions of eulexistdb. #10
- Bugfix: xquery xpath prep now handles nested function calls
- Updated unit tests so they can be run with and without django, in order to test that eulexistdb works properly without django.
- Configured unit tests on travis-ci to test with and without django.

1.2.5 0.20

- **NOTE:** `Existdb.DB` initialization parameters has changed; server url is no longer expected to include full xmlrpc path.
- Updated and tested for compatibility with eXist-db 2.2
- Improved `eulexistdb.query.QuerySet` efficiency when retrieving results (now retrieves chunked results using eXist REST API, making fewer requests to the server)
- Simple xml-based query syntax now supported via `eulexistdb.query.XmlQuery`
- Updated for compatibility with current versions of Django
- Now uses *requests <http://docs.python-requests.org/>* for REST API access and as XML-RPC transport for improved handling and connection pooling.
- New custom django-debug-toolbar panel to view existdb xqueries used to generate a django page.

1.2.6 0.19.2

- Unittest2 and Django test runner are now optional when using testutils.

1.2.7 0.19.1

- Basic support for preceding/following/preceding-sibling/following-sibling queries when returning additional fields from a query via XmlModel.
- Bugfix: support xml returns for xpaths ending with node() or *

1.2.8 0.19

- New method for sorting a `eulexistdb.query.QuerySet` by a raw XPath, for those cases when the desired sort xpath cannot be specified as an `xmlmap` field: `eulexistdb.query.QuerySet.order_by_raw()`
- The Django manage.py script for managing eXist-DB index configuration files now takes optional username and password credentials, for use with sites that run in guest mode or with limited access.
- bugfix: `QuerySet` greater than and less than filters no longer assume numeric values should be treated as numbers, to allow comparison of string values of numbers.

- bugfix: `Xquery` now correctly generates xqueries with more than one where statement.

1.2.9 0.18

- New filters and operators supported on `eulexistdb.query.QuerySet`: * `exists` - filter on the presence of absence of a node * comparison operators `gt`, `gte`, `lt`, `lte`
- Support for excluding documents using all existing filters with new method `eulexistdb.query.QuerySet.exclude()`.

1.2.10 0.17

- Support for restricting xqueries to a single document in `eulexistdb.query.QuerySet` with `document_path` filter.

1.2.11 0.16

- Development requirements can now be installed as an optional requirement of the eulexistdb package (`pip install "eulexistdb[dev]"`).
- Unit tests have been updated to use `nose`
- Provides a nose plugin to set up and tear down an eXist database collection for tests, as an alternative to the custom test runners.

1.2.12 0.15.2

- Update to latest released version of `eulxml` (0.18.0) with backwards-incompatible DateField/DateTimeField change.

1.2.13 0.15.1 - Bugfix Release

- Support Python 2.7.
- Rearrange test code to support easier recombination.

1.2.14 0.15.0 - Initial Release

- Split out existdb-specific components from `eulcore`; now depends on `eulxml`.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

pyexistdb.db, 3

m

pyexistdb.manager, 14
pyexistdb.models, 14

p

pyexistdb, 3

q

pyexistdb.query, 11

t

pyexistdb.templatetags.existdb, 15
pyexistdb.testutil, 16

Index

Symbols

_managers (pyexistdb.models.XmlModel attribute), 15

A

all() (pyexistdb.query.QuerySet method), 11
also() (pyexistdb.query.QuerySet method), 11
also_raw() (pyexistdb.query.QuerySet method), 11
alternate_test_existdb (in module pyexistdb.testutil), 17
assertPattern() (pyexistdb.testutil.TestCase method), 17

C

count (pyexistdb.db.QueryResult attribute), 10
count() (pyexistdb.query.QuerySet method), 12
create_account() (pyexistdb.db.ExistDB method), 7
create_group() (pyexistdb.db.ExistDB method), 7
createCollection() (pyexistdb.db.ExistDB method), 5, 6

D

describeDocument() (pyexistdb.db.ExistDB method), 7
distinct() (pyexistdb.query.QuerySet method), 12

E

exclude() (pyexistdb.query.QuerySet method), 12
executeQuery() (pyexistdb.db.ExistDB method), 6, 7
exist_matches() (in module pyexistdb.templatetags.existdb), 15
ExistDB (class in pyexistdb.db), 4
ExistDBException, 10
ExistDBSetUp (class in pyexistdb.testutil), 16
ExistDBTestWrapper (class in pyexistdb.testutil), 16

F

filter() (pyexistdb.query.QuerySet method), 12

G

get() (pyexistdb.query.QuerySet method), 12
get_query_set() (pyexistdb.manager.Manager method), 14
getCollectionDescription() (pyexistdb.db.ExistDB method), 7
getDoc() (pyexistdb.db.ExistDB method), 7
getDocument() (pyexistdb.db.ExistDB method), 5, 7
getDocument() (pyexistdb.query.QuerySet method), 13
getHits() (pyexistdb.db.ExistDB method), 6, 7
getPermissions() (pyexistdb.db.ExistDB method), 7

H

hasCollection() (pyexistdb.db.ExistDB method), 5, 7
hasCollectionIndex() (pyexistdb.db.ExistDB method), 7
hasDocument() (pyexistdb.db.ExistDB method), 8
help() (pyexistdb.testutil.ExistDBSetUp method), 16
hits (pyexistdb.db.QueryResult attribute), 10

L

load() (pyexistdb.db.ExistDB method), 5, 8
loadCollectionIndex() (pyexistdb.db.ExistDB method), 8

M

Manager (class in pyexistdb.manager), 14
moveDocument() (pyexistdb.db.ExistDB method), 8

O

only() (pyexistdb.query.QuerySet method), 13
only_raw() (pyexistdb.query.QuerySet method), 13
or_filter() (pyexistdb.query.QuerySet method), 13
order_by() (pyexistdb.query.QuerySet method), 13
order_by_raw() (pyexistdb.query.QuerySet method), 13

P

pyexistdb (module), 3
pyexistdb.db (module), 3
pyexistdb.manager (module), 14
pyexistdb.models (module), 14
pyexistdb.query (module), 11
pyexistdb.templatetags.existdb (module), 15
pyexistdb.testutil (module), 16

Q

query() (pyexistdb.db.ExistDB method), 5, 8
query_result_type (pyexistdb.query.QuerySet attribute), 14
QueryResult (class in pyexistdb.db), 10
QuerySet (class in pyexistdb.query), 11
querySummary() (pyexistdb.db.ExistDB method), 6, 9

R

reindexCollection() (pyexistdb.db.ExistDB method), 9
releaseQueryResult() (pyexistdb.db.ExistDB method), 6, 9
removeCollection() (pyexistdb.db.ExistDB method), 5, 9
removeCollectionIndex() (pyexistdb.db.ExistDB method), 9
removeDocument() (pyexistdb.db.ExistDB method), 9
reset() (pyexistdb.query.QuerySet method), 14
result_id (pyexistdb.query.QuerySet attribute), 14
results (pyexistdb.db.QueryResult attribute), 10
retrieve() (pyexistdb.db.ExistDB method), 6, 10
retrieve_text() (pyexistdb.db.ExistDB method), 10
return_type (pyexistdb.query.QuerySet attribute), 14

S

setPermissions() (pyexistdb.db.ExistDB method), 10

T

TestCase (class in pyexistdb.testutil), 16

U

using() (pyexistdb.query.QuerySet method), 14

X

XmlModel (class in pyexistdb.models), 14
XmlQuery (class in pyexistdb.query), 14