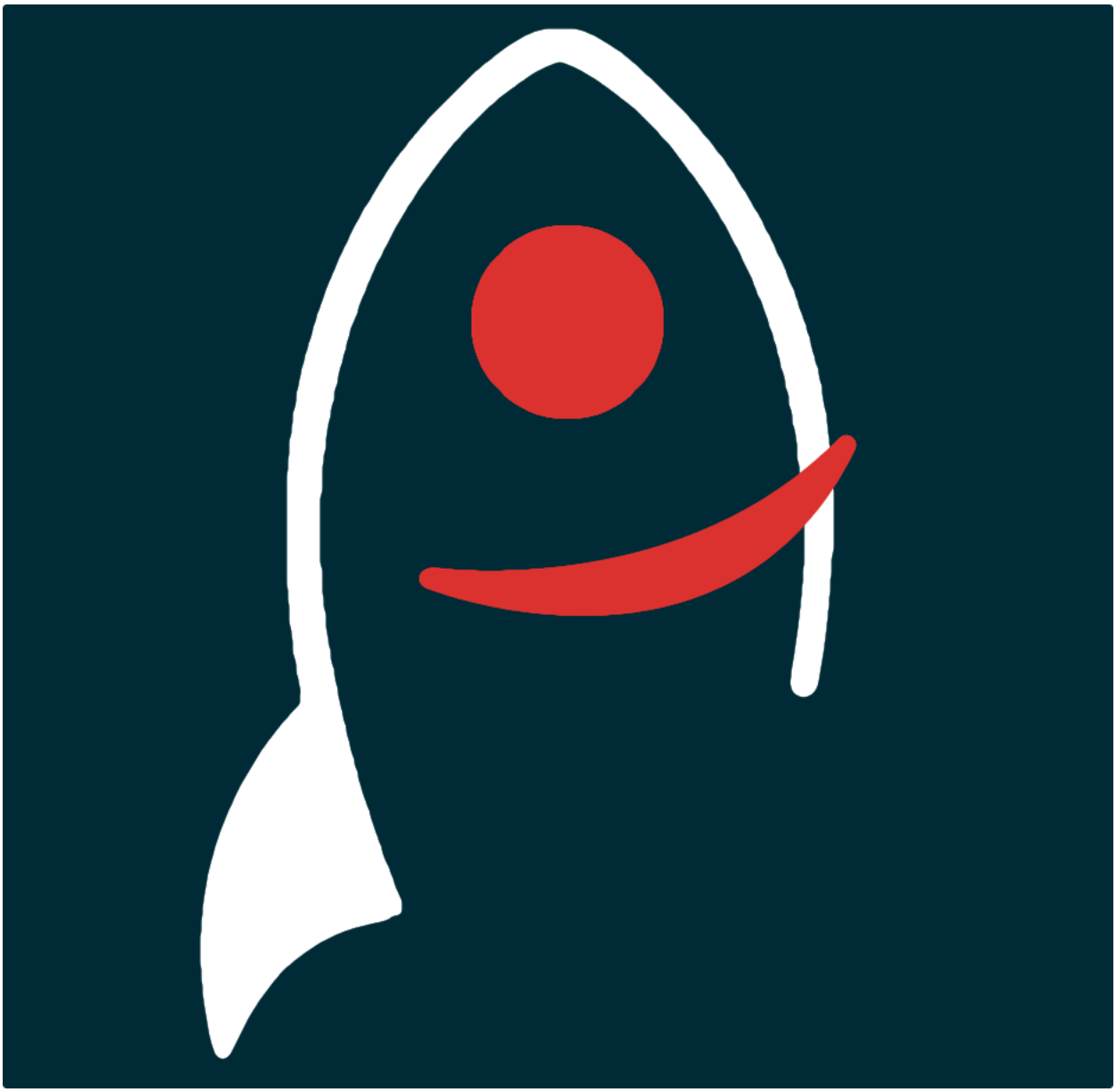

dryxPyramid Documentation

Release v0.4.5

Dave Young

2023

TABLE OF CONTENTS

basic, reusable code for pyramid webapps.

Documentation for dryxPyramid is hosted by Read the Docs (development version and master version). The code lives on github. Please report any issues you find here.

CHAPTER

ONE

FEATURES

-

1.1 Installation

The easiest way to install dryxPyramid is to use `pip` (here we show the install inside of a conda environment):

```
conda create -n dryxPyramid python=3.7 pip
conda activate dryxPyramid
pip install dryxPyramid
```

Or you can clone the github repo and install from a local version of the code:

```
git clone git@github.com:thespacedoctor/dryxPyramid.git
cd dryxPyramid
python setup.py install
```

To upgrade to the latest version of dryxPyramid use the command:

```
pip install dryxPyramid --upgrade
```

To check installation was successful run `dryxPyramid -v`. This should return the version number of the install.

1.1.1 Development

If you want to tinker with the code, then install in development mode. This means you can modify the code from your cloned repo:

```
git clone git@github.com:thespacedoctor/dryxPyramid.git
cd dryxPyramid
python setup.py develop
```

Pull requests are welcomed!

1.2 Initialisation

Before using dryxPyramid you need to use the `init` command to generate a user settings file. Running the following creates a yaml settings file in your home folder under `~/.config/dryxPyramid/dryxPyramid.yaml`:

```
dryxPyramid init
```

The file is initially populated with dryxPyramid's default settings which can be adjusted to your preference.

If at any point the user settings file becomes corrupted or you just want to start afresh, simply trash the `dryxPyramid.yaml` file and rerun `dryxPyramid init`.

1.2.1 Modifying the Settings

Once created, open the settings file in any text editor and make any modifications needed.

1.2.2 Basic Python Setup

If you plan to use `dryxPyramid` in your own scripts you will first need to parse your settings file and set up logging etc. One quick way to do this is to use the `fundamentals` package to give you a logger, a settings dictionary and a database connection (if connection details given in settings file):

```
## SOME BASIC SETUP FOR LOGGING, SETTINGS ETC
from fundamentals import tools
from os.path import expanduser
home = expanduser("~")
settingsFile  = home + "/.config/dryxPyramid/dryxPyramid.yaml"
su = tools(
    arguments={"settingsFile": settingsFile},
    docString=__doc__,
)
arguments, settings, log, dbConn = su.setup()
```

1.3 Todo List

Todo:

- nice!

(The *original entry* is located in /home/docs/checkouts/readthedocs.org/user_builds/dryxpyramid/checkouts/develop/docs/source/_templ
line 1.)

1.4 Release Notes

v0.4.5 - June 15, 2022

- **FIXED** - removing tests from release

v0.4.3 - May 10, 2022

- **FIXED** - fixing docs

v0.4.2 - December 11, 2020

- **FIXED** - doc dependencies

v0.4.1 - June 24, 2020

- **enhancements** - added some tweaks for the python 3 version of webapps

v0.4.0 - May 25, 2020

- Now compatible with Python 3.*

CHAPTER

TWO

API REFERENCE

2.1 Modules

dryxPyramid.models	
dryxPyramid.renderers	
dryxPyramid.templates	
dryxPyramid.templates.responses	
dryxPyramid.views	
dryxPyramid.models.models_base	*The base model for other model modules to build on top of*
dryxPyramid.models.models_login_post	
dryxPyramid.security	
dryxPyramid.views.views_base	
dryxPyramid.views.views_base_element	
dryxPyramid.views.views_download	
dryxPyramid.views.views_login	
dryxPyramid.views.views_logout	

2.1.1 models *(module)*

Sub-modules

models_base	*The base model for other model modules to build on top of*
models_login_post	

2.1.2 renderers *(module)*

Classes

renderer_csv(info)	*The CSV renderer - can return plain text in browser or a file to download*
renderer_json(info)	*The json renderer - can return content to browser or a file to download*
renderer_plain_table(info)	*The plain_table renderer - can return content to browser or a file to download*
renderer_plain_text(info)	*The plain_text renderer - can return content to browser or a file to download*

Sub-modules

renderer_csv(info)	*The CSV renderer - can return plain text in browser or a file to download*
renderer_json(info)	*The json renderer - can return content to browser or a file to download*
renderer_plain_table(info)	*The plain_table renderer - can return content to browser or a file to download*
renderer_plain_text(info)	*The plain_text renderer - can return content to browser or a file to download*

2.1.3 templates *(module)*

Sub-modules

responses	

2.1.4 responses *(module)*

Classes

templates_download(log, request[, elementId])	*The worker class for the templates_download module*
templates_login(log, request[, ...])	*The worker class for the templates_login module*

Sub-modules

templates_download(log, request[, elementId])	*The worker class for the templates_download module*
templates_login(log, request[, ...])	*The worker class for the templates_login module*

2.1.5 views *(module)*

Sub-modules

views_base
views_base_element
views_download
views_login
views_logout

2.1.6 models_base *(module)*

The base model for other model modules to build on top of

Author David Young

Classes

base_model(log, request[, elementId, search])	A superclass model for pyramid apps
`object`()	The base class of the class hierarchy.

Sub-modules

base_model(log, request[, elementId, search])	A superclass model for pyramid apps
object()	The base class of the class hierarchy.
os	OS routines for NT or Posix depending on what system we're on.
sys	This module provides access to some objects used or maintained by the interpreter and to functions that interact strongly with the interpreter.

class base_model(*log*, *request*, *elementId=False*, *search=False*)

Bases: object

A superclass model for pyramid apps

Key Arguments:

- log – logger
- request – the pyramid request
- elementId – the specific element id requests (or False)
- search – is the result given from a search query

close()

2.1.7 models_login_post *(module)*

Classes

RootFactory(request)	

object()	The base class of the class hierarchy.

Sub-modules

Allow	str(object='') -> str str(bytes_or_buffer[, encoding[, errors]]) -> str
Everyone	str(object='') -> str str(bytes_or_buffer[, encoding[, errors]]) -> str
RootFactory(request)	
object()	The base class of the class hierarchy.

class RootFactory(*request*)

Bases: object

2.1.8 security *(module)*

Classes

`zip`	zip(*iterables) –> A zip object yielding tuples until an input is exhausted.

Functions

`get_users_and_groups`(request)	*Get the users and groups from the database*
`groupfinder`(userid, request)	*If the userid exists in the system, it will return a sequence of group identifiers (or an empty sequence if the user isn't a member of any groups).

Sub-modules

`get_users_and_groups`(request)	*Get the users and groups from the database*
`groupfinder`(userid, request)	*If the userid exists in the system, it will return a sequence of group identifiers (or an empty sequence if the user isn't a member of any groups).
`zip`	zip(*iterables) –> A zip object yielding tuples until an input is exhausted.

get_users_and_groups (*request*)
Get the users and groups from the database

groupfinder (*userid*, *request*)
If the userid exists in the system, it will return a sequence of group identifiers (or an empty sequence if the user isn't a member of any groups). If the userid does not exist in the system, it will return ``None``.

2.1.9 views_base *(module)*

Classes

`Response`([body, status, headerlist, ...])

`base_view`(request)	*The base view that can be used for any API resource*
`object`()	The base class of the class hierarchy.
`view_config`(**settings)	A function, class or method decorator which allows a developer to create view registrations nearer to a :term:`view callable` definition than use :term:`imperative configuration` to do the same.

Functions

view_defaults(**settings)	A class decorator which, when applied to a class, will provide defaults for all view configurations that use the class.

Sub-modules

HTTPFound([location, detail, headers, ...])	subclass of _HTTPMove
Response([body, status, headerlist, ...])	

base_view(request)	*The base view that can be used for any API resource*
logging	Logging package for Python.
object()	The base class of the class hierarchy.
view_config(**settings)	A function, class or method decorator which allows a developer to create view registrations nearer to a :term:view callable definition than use :term:imperative configuration to do the same.
view_defaults(**settings)	A class decorator which, when applied to a class, will provide defaults for all view configurations that use the class.

class base_view(*request*)

Bases: object

The base view that can be used for any API resource

delete()

get()

get_csv()

get_html()

get_json()

get_plain_table()

post()

put()

2.1.10 views_base_element *(module)*

Classes

Response([body, status, headerlist, ...])	

base_element_view(request)	

continues on next page

Table 19 – continued from previous page

object()	The base class of the class hierarchy.
view_config(**settings)	A function, class or method decorator which allows a developer to create view registrations nearer to a :term:view callable definition than use :term:imperative configuration to do the same.

Functions

view_defaults(**settings)	A class decorator which, when applied to a class, will provide defaults for all view configurations that use the class.

Sub-modules

HTTPFound([location, detail, headers, ...])	subclass of _HTTPMove
Response([body, status, headerlist, ...])	
base_element_view(request)	
logging	Logging package for Python.
object()	The base class of the class hierarchy.
view_config(**settings)	A function, class or method decorator which allows a developer to create view registrations nearer to a :term:view callable definition than use :term:imperative configuration to do the same.
view_defaults(**settings)	A class decorator which, when applied to a class, will provide defaults for all view configurations that use the class.

class base_element_view(*request*)

Bases: object

delete()

get()

get_csv()

get_html()

get_json()

get_plain_table()

post()

put()

2.1.11 views_download *(module)*

Classes

Response([body, status, headerlist, ...])	
download_view(request)	
object()	The base class of the class hierarchy.
templates_download(log, request[, elementId])	*The worker class for the templates_download module*
view_config(**settings)	A function, class or method decorator which allows a developer to create view registrations nearer to a :term:view callable definition than use :term:imperative configuration to do the same.

Functions

view_defaults(**settings)	A class decorator which, when applied to a class, will provide defaults for all view configurations that use the class.

Sub-modules

HTTPFound([location, detail, headers, ...])	subclass of _HTTPMove
Response([body, status, headerlist, ...])	
download_view(request)	
logging	Logging package for Python.
object()	The base class of the class hierarchy.
templates_download(log, request[, elementId])	*The worker class for the templates_download module*
view_config(**settings)	A function, class or method decorator which allows a developer to create view registrations nearer to a :term:view callable definition than use :term:imperative configuration to do the same.
view_defaults(**settings)	A class decorator which, when applied to a class, will provide defaults for all view configurations that use the class.

class download_view(*request*)

Bases: object

delete()

get()

post()

put()

2.1.12 views_login *(module)*

Classes

Response([body, status, headerlist, ...])	
forbidden_view_config(**settings)	New in version 1.3.
login_view(request)	
object()	The base class of the class hierarchy.
sha256_crypt([implicit_rounds])	This class implements the SHA256-Crypt password hash, and follows the password-hash-api.
templates_login(log, request[, ...])	*The worker class for the templates_login module*
view_config(**settings)	A function, class or method decorator which allows a developer to create view registrations nearer to a :term:view callable definition than use :term:imperative configuration to do the same.

Functions

forbidden(request)	
forget(request, **kw)	Return a sequence of header tuples (e.g.
get_users_and_groups(request)	*Get the users and groups from the database*
remember(request, userid, **kw)	Returns a sequence of header tuples (e.g.
view_defaults(**settings)	A class decorator which, when applied to a class, will provide defaults for all view configurations that use the class.

Sub-modules

HTTPFound([location, detail, headers, ...])	subclass of _HTTPMove
Response([body, status, headerlist, ...])	
forbidden(request)	
forbidden_view_config(**settings)	New in version 1.3.
forget(request, **kw)	Return a sequence of header tuples (e.g.
get_users_and_groups(request)	*Get the users and groups from the database*
logging	Logging package for Python.
login_view(request)	
object()	The base class of the class hierarchy.

continues on next page

Table 27 – continued from previous page

remember(request, userid, **kw)	Returns a sequence of header tuples (e.g.
sha256_crypt([implicit_rounds])	This class implements the SHA256-Crypt password hash, and follows the password-hash-api.
templates_login(log, request[, ...])	*The worker class for the templates_login module*
view_config(**settings)	A function, class or method decorator which allows a developer to create view registrations nearer to a :term:view callable definition than use :term:imperative configuration to do the same.
view_defaults(**settings)	A class decorator which, when applied to a class, will provide defaults for all view configurations that use the class.

class login_view(*request*)

Bases: object

login()

forbidden(*request*)

2.1.13 views_logout *(module)*

Classes

Response([body, status, headerlist, ...])	
forbidden_view_config(**settings)	New in version 1.3.
view_config(**settings)	A function, class or method decorator which allows a developer to create view registrations nearer to a :term:view callable definition than use :term:imperative configuration to do the same.

Functions

forget(request, **kw)	Return a sequence of header tuples (e.g.
logout(request)	
remember(request, userid, **kw)	Returns a sequence of header tuples (e.g.
view_defaults(**settings)	A class decorator which, when applied to a class, will provide defaults for all view configurations that use the class.

Sub-modules

HTTPFound([location, detail, headers, ...])	subclass of _HTTPMove
Response([body, status, headerlist, ...])	
forbidden_view_config(**settings)	New in version 1.3.
forget(request, **kw)	Return a sequence of header tuples (e.g.
logging	Logging package for Python.
logout(request)	
remember(request, userid, **kw)	Returns a sequence of header tuples (e.g.
view_config(**settings)	A function, class or method decorator which allows a developer to create view registrations nearer to a :term:view callable definition than use :term:imperative configuration to do the same.
view_defaults(**settings)	A class decorator which, when applied to a class, will provide defaults for all view configurations that use the class.

logout(*request*)

2.2 Classes

dryxPyramid.models.models_base.base_model	A superclass model for pyramid apps
dryxPyramid.models.models_login_post.RootFactory	
dryxPyramid.renderers.renderer_csv	*The CSV renderer - can return plain text in browser or a file to download*
dryxPyramid.renderers.renderer_json	*The json renderer - can return content to browser or a file to download*
dryxPyramid.renderers.renderer_plain_table	*The plain_table renderer - can return content to browser or a file to download*
dryxPyramid.renderers.renderer_plain_text	*The plain_text renderer - can return content to browser or a file to download*
dryxPyramid.templates.responses.templates_download	*The worker class for the templates_download module*
dryxPyramid.templates.responses.templates_login	*The worker class for the templates_login module*
dryxPyramid.views.views_base.base_view	*The base view that can be used for any API resource*
dryxPyramid.views.views_base_element.base_element_view	
dryxPyramid.views.views_download.download_view	

continues on next page

Table 31 – continued from previous page

dryxPyramid.views.views_login.login_view

2.2.1 base_model *(class)*

class base_model(*log*, *request*, *elementId=False*, *search=False*)

Bases: object

A superclass model for pyramid apps

Key Arguments:

- log – logger
- request – the pyramid request
- elementId – the specific element id requests (or False)
- search – is the result given from a search query

Methods

close()

2.2.2 RootFactory *(class)*

class RootFactory(*request*)

Bases: object

Methods

2.2.3 renderer_csv *(class)*

class renderer_csv(*info*)

Bases: object

The CSV renderer - can return plain text in browser or a file to download

Methods

—

2.2.4 renderer_json *(class)*

class renderer_json(*info*)

Bases: `pyramid.renderers.JSON`

The json renderer - can return content to browser or a file to download

Methods

`add_adapter`(type_or_iface, adapter)	When an object of the type (or interface) `type_or_iface` fails to automatically encode using the serializer, the renderer will use the adapter `adapter` to convert it into a JSON-serializable object.
`datetime_adapter`(obj, request)	
`decimal_adapter`(obj, request)	

add_adapter(*type_or_iface*, *adapter*)

When an object of the type (or interface) `type_or_iface` fails to automatically encode using the serializer, the renderer will use the adapter `adapter` to convert it into a JSON-serializable object. The adapter must accept two arguments: the object and the currently active request.

```
class Foo:
    x = 5

def foo_adapter(obj, request):
    return obj.x

renderer = JSON(indent=4)
renderer.add_adapter(Foo, foo_adapter)
```

When you've done this, the JSON renderer will be able to serialize instances of the `Foo` class when they're encountered in your view results.

2.2.5 renderer_plain_table *(class)*

class renderer_plain_table(*info*)

Bases: `object`

The plain_table renderer - can return content to browser or a file to download

Methods

2.2.6 renderer_plain_text *(class)*

class renderer_plain_text(*info*)
Bases: object

The plain_text renderer - can return content to browser or a file to download

Methods

2.2.7 templates_download *(class)*

class templates_download(*log*, *request*, *elementId=False*)
Bases: object

The worker class for the templates_download module

Key Arguments

- log – logger
- request – the pyramid request
- elementId – the specific element requested (or False)

Methods

close()	
get()	*get the templates_download object*

get()
get the templates_download object

Return

- responseContent – the response

2.2.8 templates_login *(class)*

class templates_login(*log*, *request*, *mainCssFilePath='main.css'*, *jsFilePath='main-ck.js'*, *pageTitle='Login'*, *iconPath=''*, *came_from='/'*, *message=''*)
Bases: object

The worker class for the templates_login module

Key Arguments

- log – logger

- `request` – the pyramid request
- `mainCssFilePath` – the filename of the main css file
- `jsFilePath` – the filename of the main js file
- `pageTitle` – pageTitle
- `icon` – webapp icon
- `came_from` – the url this login page was triggered from
- `message` – message to display as notification

Methods

`close`()	
`get`()	*get the templates_login object*

get()
get the templates_login object

Return

- `loginPage` – the login page

2.2.9 base_view *(class)*

class base_view(*request*)
Bases: `object`

The base view that can be used for any API resource

Methods

delete()	
get()	
get_csv()	
get_html()	
get_json()	
get_plain_table()	
post()	
put()	

2.2.10 base_element_view *(class)*

class base_element_view(*request*)
Bases: object

Methods

delete()
get()
get_csv()
get_html()
get_json()
get_plain_table()
post()
put()

2.2.11 download_view *(class)*

class download_view(*request*)
Bases: object

Methods

delete()
get()
post()
put()

2.2.12 login_view *(class)*

class login_view(*request*)
Bases: object

Methods

login()	

2.3 Functions

dryxPyramid.security.get_users_and_groups	*Get the users and groups from the database*
dryxPyramid.security.groupfinder	*If the userid exists in the system, it will return a sequence of group identifiers (or an empty sequence if the user isn't a member of any groups).
dryxPyramid.views.views_login.forbidden	
dryxPyramid.views.views_logout.logout	

2.3.1 get_users_and_groups (*function*)

get_users_and_groups(*request*)
Get the users and groups from the database

2.3.2 groupfinder (*function*)

groupfinder(*userid*, *request*)
If the userid exists in the system, it will return a sequence of group identifiers (or an empty sequence if the user isn't a member of any groups). If the userid does not exist in the system, it will return ``None``.

2.3.3 forbidden (*function*)

forbidden(*request*)

2.3.4 logout (*function*)

logout(*request*)

2.4 A-Z Index

Modules

dryxPyramid.models	
dryxPyramid.renderers	
dryxPyramid.templates	
dryxPyramid.templates.responses	
dryxPyramid.views	
dryxPyramid.models.models_base	*The base model for other model modules to build on top of*
dryxPyramid.models.models_login_post	
dryxPyramid.security	
dryxPyramid.views.views_base	
dryxPyramid.views.views_base_element	
dryxPyramid.views.views_download	
dryxPyramid.views.views_login	
dryxPyramid.views.views_logout	

Classes

dryxPyramid.models.models_base.base_model	A superclass model for pyramid apps
dryxPyramid.models.models_login_post.RootFactory	
dryxPyramid.renderers.renderer_csv	*The CSV renderer - can return plain text in browser or a file to download*
dryxPyramid.renderers.renderer_json	*The json renderer - can return content to browser or a file to download*

continues on next page

Table 46 – continued from previous page

dryxPyramid.renderers.renderer_plain_table	*The plain_table renderer - can return content to browser or a file to download*
dryxPyramid.renderers.renderer_plain_text	*The plain_text renderer - can return content to browser or a file to download*
dryxPyramid.templates.responses.templates_download	*The worker class for the templates_download module*
dryxPyramid.templates.responses.templates_login	*The worker class for the templates_login module*
dryxPyramid.views.views_base.base_view	*The base view that can be used for any API resource*
dryxPyramid.views.views_base_element.base_element_view	
dryxPyramid.views.views_download.download_view	
dryxPyramid.views.views_login.login_view	

Functions

dryxPyramid.security.get_users_and_groups	*Get the users and groups from the database*
dryxPyramid.security.groupfinder	*If the userid exists in the system, it will return a sequence of group identifiers (or an empty sequence if the user isn't a member of any groups).
dryxPyramid.views.views_login.forbidden	
dryxPyramid.views.views_logout.logout	

CHAPTER

THREE

RELEASE NOTES

v0.4.5 - June 15, 2022

- **FIXED** - removing tests from release

v0.4.3 - May 10, 2022

- **FIXED** - fixing docs

v0.4.2 - December 11, 2020

- **FIXED** - doc dependencies

v0.4.1 - June 24, 2020

- **enhancements** - added some tweaks for the python 3 version of webapps

v0.4.0 - May 25, 2020

- Now compatible with Python 3.*

PYTHON MODULE INDEX

m

r

s

t

v

INDEX

P

R

T