



dryxPyramid Documentation

Release v0.4.2

Dave Young

2021

TABLE OF CONTENTS

1	Features	3
1.1	Installation	3
1.1.1	Development	3
1.2	Initialisation	4
1.2.1	Modifying the Settings	4
1.2.2	Basic Python Setup	4
1.3	Todo List	4
1.4	Release Notes	5
2	API Reference	7
2.1	Modules	7
2.1.1	models (<i>module</i>)	8
2.1.2	renderers (<i>module</i>)	8
2.1.3	templates (<i>module</i>)	8
2.1.4	responses (<i>module</i>)	9
2.1.5	views (<i>module</i>)	9
2.1.6	models_base (<i>module</i>)	9
2.1.7	models_login_post (<i>module</i>)	10
2.1.8	security (<i>module</i>)	11
2.1.9	views_base (<i>module</i>)	11
2.1.10	views_base_element (<i>module</i>)	12
2.1.11	views_download (<i>module</i>)	14
2.1.12	views_login (<i>module</i>)	15
2.1.13	views_logout (<i>module</i>)	16
2.2	Classes	17
2.2.1	base_model (<i>class</i>)	18
2.2.2	RootFactory (<i>class</i>)	18
2.2.3	renderer_csv (<i>class</i>)	18
2.2.4	renderer_json (<i>class</i>)	19
2.2.5	renderer_plain_table (<i>class</i>)	19
2.2.6	renderer_plain_text (<i>class</i>)	20
2.2.7	templates_download (<i>class</i>)	20
2.2.8	templates_login (<i>class</i>)	20
2.2.9	base_view (<i>class</i>)	21
2.2.10	base_element_view (<i>class</i>)	22
2.2.11	download_view (<i>class</i>)	22
2.2.12	login_view (<i>class</i>)	23
2.3	Functions	23
2.3.1	get_users_and_groups (<i>function</i>)	23
2.3.2	groupfinder (<i>function</i>)	23

2.3.3	forbidden (<i>function</i>)	24
2.3.4	logout (<i>function</i>)	24
2.4	A-Z Index	24
3	Release Notes	27
	Python Module Index	29
	Index	31

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](#).

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/master/docs/source/_templates/line 1.`)

1.4 Release Notes

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.*

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

<code>models_base</code>	<i>The base model for other model modules to build on top of</i>
<code>models_login_post</code>	

2.1.2 renderers (*module*)

Classes

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

Sub-modules

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

2.1.3 templates (*module*)

Sub-modules

<code>responses</code>	
------------------------	--

2.1.4 responses (*module*)

Classes

<code>templates_download(log, request[, elementId])</code>	<i>The worker class for the templates_download module</i>
<code>templates_login(log, request[, ...])</code>	<i>The worker class for the templates_login module</i>

Sub-modules

<code>templates_download(log, request[, elementId])</code>	<i>The worker class for the templates_download module</i>
<code>templates_login(log, request[, ...])</code>	<i>The worker class for the templates_login module</i>

2.1.5 views (*module*)

Sub-modules

<code>views_base</code>
<code>views_base_element</code>
<code>views_download</code>
<code>views_login</code>
<code>views_logout</code>

2.1.6 models_base (*module*)

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

Author David Young

Classes

<code>base_model(log, request[, elementId, search])</code>	A superclass model for pyramid apps
<code>object()</code>	The most base type

Sub-modules

<code>base_model(log, request[, elementId, search])</code>	A superclass model for pyramid apps
<code>object()</code>	The most base type
<code>os</code>	OS routines for NT or Posix depending on what system we're on.
<code>sys</code>	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

<code>RootFactory(request)</code>	
<code>object()</code>	The most base type

Sub-modules

<code>Allow</code>	<code>str(object='') -> str str(bytes_or_buffer[, encoding[, errors]]) -> str</code>
<code>Everyone</code>	<code>str(object='') -> str str(bytes_or_buffer[, encoding[, errors]]) -> str</code>
<code>RootFactory(request)</code>	
<code>object()</code>	The most base type

class `RootFactory` (*request*)

Bases: `object`

2.1.8 security (module)

Classes

<code>zip</code>	<code>zip(*iterables)</code> -> zip object
------------------	--

Functions

<code>get_users_and_groups(request)</code>	<i>Get the users and groups from the database</i>
<code>groupfinder(userid, request)</code>	<i>*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).</i>

Sub-modules

<code>get_users_and_groups(request)</code>	<i>Get the users and groups from the database</i>
<code>groupfinder(userid, request)</code>	<i>*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).</i>
<code>zip</code>	<code>zip(*iterables)</code> -> zip object

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

<code>Response([body, status, headerlist, ...])</code>	
<code>base_view(request)</code>	<i>The base view that can be used for any API resource</i>
<code>object()</code>	The most base type
<code>view_config(**settings)</code>	A function, class or method decorator which allows a developer to create view registrations nearer to a <code>:term:view callable</code> definition than use <code>:term:imperative</code> configuration to do the same.

Functions

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

Sub-modules

<code>HTTPFound([location, detail, headers, ...])</code>	subclass of <code>_HTTPMove</code>
<code>Response([body, status, headerlist, ...])</code>	
<code>base_view(request)</code>	<i>The base view that can be used for any API resource</i>
<code>logging</code>	Logging package for Python.
<code>object()</code>	The most base type
<code>view_config(**settings)</code>	A function, class or method decorator which allows a developer to create view registrations nearer to a <code>:term:view callable</code> definition than use <code>:term:imperative configuration</code> to do the same.
<code>view_defaults(**settings)</code>	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

<code>Response([body, status, headerlist, ...])</code>
<code>base_element_view(request)</code>

continues on next page

Table 19 – continued from previous page

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

Functions

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

Sub-modules

<code>HTTPFound([location, detail, headers, ...])</code>	subclass of <code>_HTTPMove</code>
<code>Response([body, status, headerlist, ...])</code>	
<code>base_element_view(request)</code>	
<code>logging</code>	Logging package for Python.
<code>object()</code>	The most base type
<code>view_config(**settings)</code>	A function, class or method decorator which allows a developer to create view registrations nearer to a <code>:term:view callable definition</code> than use <code>:term:imperative configuration</code> to do the same.
<code>view_defaults(**settings)</code>	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

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

Functions

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

Sub-modules

<code>HTTPFound([location, detail, headers, ...])</code>	subclass of <code>_HTTPMove</code>
<code>Response([body, status, headerlist, ...])</code>	
<code>download_view(request)</code>	
<code>logging</code>	Logging package for Python.
<code>object()</code>	The most base type
<code>templates_download(log, request[, elementId])</code>	<i>The worker class for the templates_download module</i>
<code>view_config(**settings)</code>	A function, class or method decorator which allows a developer to create view registrations nearer to a <code>:term:view callable</code> definition than use <code>:term:imperative</code> configuration to do the same.
<code>view_defaults(**settings)</code>	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

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

Functions

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

Sub-modules

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

continues on next page

Table 27 – continued from previous page

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

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

Functions

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

Sub-modules

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

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

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

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

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

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

Classes

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

continues on next page

Table 46 – continued from previous page

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

Functions

<code>dryxPyramid.security. get_users_and_groups</code>	<i>Get the users and groups from the database</i>
<code>dryxPyramid.security.groupfinder</code>	<i>*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).</i>
<code>dryxPyramid.views.views_login. forbidden</code>	
<code>dryxPyramid.views.views_logout.logout</code>	

RELEASE NOTES

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

`dryxPyramid.models`, [8](#)
`dryxPyramid.models.models_base`, [9](#)
`dryxPyramid.models.models_login_post`,
[10](#)

r

`dryxPyramid.renderers`, [8](#)

s

`dryxPyramid.security`, [11](#)

t

`dryxPyramid.templates`, [8](#)
`dryxPyramid.templates.responses`, [9](#)

v

`dryxPyramid.views`, [9](#)
`dryxPyramid.views.views_base`, [11](#)
`dryxPyramid.views.views_base_element`,
[12](#)
`dryxPyramid.views.views_download`, [14](#)
`dryxPyramid.views.views_login`, [15](#)
`dryxPyramid.views.views_logout`, [16](#)

A

`add_adapter()` (*renderer_json method*), 19

B

`base_element_view` (class in `dryxPyramid.views.views_base_element`), 13, 22

`base_model` (class in `dryxPyramid.models.models_base`), 10, 18

`base_view` (class in `dryxPyramid.views.views_base`), 12, 21

C

`close()` (*base_model method*), 10

D

`delete()` (*base_element_view method*), 13

`delete()` (*base_view method*), 12

`delete()` (*download_view method*), 14

`download_view` (class in `dryxPyramid.views.views_download`), 14, 22

`dryxPyramid.models`
module, 8

`dryxPyramid.models.models_base`
module, 9

`dryxPyramid.models.models_login_post`
module, 10

`dryxPyramid.renderers`
module, 8

`dryxPyramid.security`
module, 11

`dryxPyramid.templates`
module, 8

`dryxPyramid.templates.responses`
module, 9

`dryxPyramid.views`
module, 9

`dryxPyramid.views.views_base`
module, 11

`dryxPyramid.views.views_base_element`
module, 12

`dryxPyramid.views.views_download`
module, 14

`dryxPyramid.views.views_login`
module, 15

`dryxPyramid.views.views_logout`
module, 16

F

`forbidden()` (in module `dryxPyramid.views.views_login`), 16, 24

G

`get()` (*base_element_view method*), 13

`get()` (*base_view method*), 12

`get()` (*download_view method*), 14

`get()` (*templates_download method*), 20

`get()` (*templates_login method*), 21

`get_csv()` (*base_element_view method*), 13

`get_csv()` (*base_view method*), 12

`get_html()` (*base_element_view method*), 13

`get_html()` (*base_view method*), 12

`get_json()` (*base_element_view method*), 13

`get_json()` (*base_view method*), 12

`get_plain_table()` (*base_element_view method*), 13

`get_plain_table()` (*base_view method*), 12

`get_users_and_groups()` (in module `dryxPyramid.security`), 11, 23

`groupfinder()` (in module `dryxPyramid.security`), 11, 23

L

`login()` (*login_view method*), 16

`login_view` (class in `dryxPyramid.views.views_login`), 16, 23

`logout()` (in module `dryxPyramid.views.views_logout`), 17, 24

M

module

`dryxPyramid.models`, 8

`dryxPyramid.models.models_base`, 9

`dryxPyramid.models.models_login_post`, 10

- `dryxPyramid.renderers`, 8
- `dryxPyramid.security`, 11
- `dryxPyramid.templates`, 8
- `dryxPyramid.templates.responses`, 9
- `dryxPyramid.views`, 9
- `dryxPyramid.views.views_base`, 11
- `dryxPyramid.views.views_base_element`, 12
- `dryxPyramid.views.views_download`, 14
- `dryxPyramid.views.views_login`, 15
- `dryxPyramid.views.views_logout`, 16

P

- `post()` (*base_element_view method*), 13
- `post()` (*base_view method*), 12
- `post()` (*download_view method*), 14
- `put()` (*base_element_view method*), 13
- `put()` (*base_view method*), 12
- `put()` (*download_view method*), 14

R

- `renderer_csv` (*class in dryxPyramid.renderers*), 18
- `renderer_json` (*class in dryxPyramid.renderers*), 19
- `renderer_plain_table` (*class in dryxPyramid.renderers*), 19
- `renderer_plain_text` (*class in dryxPyramid.renderers*), 20
- `RootFactory` (*class in dryxPyramid.models.models_login_post*), 10, 18

T

- `templates_download` (*class in dryxPyramid.templates.responses*), 20
- `templates_login` (*class in dryxPyramid.templates.responses*), 20