dreamtools

Version 1.0.3

dreamgeeker

févr. 04, 2021

Chapitre 1

Dreamtools

Dreamtools est un outils d'aide au développement contenant une liste de fonction d'utilisation basique

1.1 Installation

```
$ pip install dreamtools
$ tools-installer
**********************************************************************
** Création architecture
** ------------------------------------------------------------------
**  Répertoire logs
**      >> Répertoire créé :  C:\Users\Ohanna\Geekspace\testpack\logs
**  Répertoire configuration
**      >> Répertoire créé :  C:\Users\Ohanna\Geekspace\testpack\cfg
**==================================================================-
```

Le répertoire de configuration "cfg" sera créé à la racine du projet.

Avertissement Initiliser les données de l'application

1.2 Configuration

```
import dreamtools

app_name = "AMON_APP"    #nom de votre application
dreamtools.config(app_name, mode='DEBUG')  # par défaut mode ='PROD'
```

Avertissement Le paquet comprend un module de cryptage non supporté par Winddows

1.3 Crédits

Conçut par Dreamgeerker

Chapitre 2

Librairie de fonctions

2.1 Module de fonctions basiques

Liste de fonctions utiles

pathfile : dreamtools/tools

2.1.1 Constantes globales

Note

- RGX_ACCENTS = "àâäãéèêëîïôöõòùüûÿñç"
- RGX_EMAIL = Expression reguliere email
- RGX_PUNCT = Caractere speciaux autorisé pour mot de passe
- RGX_PWD = Expression régulière pour un mot de passe de 8 à 12 avec un car.Special/une Majuscule/Une minuscule
- RGX_PHONE = Expression réguliere rematives à un numéro de téléphon
- RGX_URL = expression reguliere pour URL
- *PROJECT_DIR* : Repertoire du projet
- *APP_NAME* : Nom de l'application
- *APP_DIR* : PROJECT_DIR/APP_NAME
- *TMP_DIR* : PROJECT_DIR/tmp

Avertissement Il faut configurer l'application afin d'avoir accès au variable PROJECT_DIR, APP_-NAME, TMP_DIR

```
>>> from dreamtools import config
>>> from dreamtools import tools
>>> config.CConfig('monapp')
>>> print (tools.APP_DIR)
'../PROJECT/mon_app'
```

add_list (*v*, *ll*) Ajout d'un item dans une liste avec gestion des doublons

Paramètres
- **v** (*str*) – valeur à ajouter
- **ll** (*list*) – liste

addhex (*h, v*) Additionne une valeur hexadécimal

Paramètres
- **h** (*str*) – valeur hexadécimal
- **v** (*int*) – valeur entière à ajouter

Renvoie valeur additionné en hexedécimal

Example
```
>>> hx = '0x129'
>>> addhex(hx, 2)
0x12b
```

aleatoire (*end, s=1*) Génération d'un nombre aléatoire entre [1-end] => end caractère

Paramètres
- **end** (*int*) – valeur maximal (paut indiquer la taille si s=1)
- **s** – valeur de départ, default to 1

Renvoie Un chiffre aléatoire

Exemple
```
>>> aleatoire (5)
1 : Renvoie un chiffre entre 1 et 5
>>> aleatoire (5,3)
1 : Renvoie un chiffre entre 3 et 5
>>> 4
```

check_password (*s*) Vérifie que la syntaxe d'une chaine répond au critère d'un mot de passe

Conditions
- Une majuscule
- Une minuscule
- Un chiffre
- Un carectère spécial (@#!?$&-_ autorisé)

Paramètres **s** (*str*) – chaine à vérifier

Return bool True si la chaine est valide

clean_allspace (*ch, very_all=True*) Nettoyage de tous les espace et carateres vides

Paramètres
- **ch** (*str*) – Chaine à nettoyer
- **very_all** (*bool*) – caractère vide aussi, True (False = Espaces uniquement)

Exemple
```
>>> chaine = 'Se  réveiller au matin
     de sa destiné !'
>>> clean_allspace (chaine)
'Seréveilleraumatindesadestiné!'
```

clean_coma (*ch, w_punk=False*) Supprime les accents/caractères spéciaux du texte source en respectant la casse

Paramètres
- **ch** – Chaine de caractere à « nettoyer »
- **w_punk** – indique si la punctuation est à nettoyer ou pas (suppression)

Exemple
```
>>> s = 'Se  réveiller au matin    de sa
destiné !!'
>>> clean_coma (s)
'Se seveiller au matin (ou pas) de sa
```

```
destine !!''
>>> clean_coma (s, True)
'Se reveiller au matin ou pas de sa destine'
```

clean_master (*ch*) Supprime les accents, caractères spéciaux et espace du texte source

Paramètres **ch** (*str*) – Chaine de caractere à « nettoyer »

Return str chaine sans accents:car. spéciaux ni espace en minuscule

Exemple

```
>>> s = 'Se  réveiller au matin  (ou pas) de sa
destiné !'
>>> clean_master (s)
'sereveilleraumatinoupasdesadestine
```

clean_space (*ch*) Nettoyage des espaces « superflus »

- Espaces à gouche et à droite supprimés
- Répétition d'espace réduit

Exemple

```
>>> chaine = 'Se  réveiller au matin        de sa
destiné    ! !           '
>>> clean_space (chaine)
'Se réveiller au matin  de sa destiné ! !'
```

code_maker (*i_size=4*) Génération d'une chaine aléatoire composé de lettre et de chiffres

Paramètres **i_size** (*int*) – taille du code
Rtype str

comphex (*hx_a, hx_b*) Compare deux valeurs hexadécimales

Paramètres

- **hx_a** (*str*) –
- **hx_b** (*str*) –

Return int

- 0 : hx_a == hx_b
- 1 : hx_a > hx_b
- -1 : hx_a < hx_b

dicFindKey (*value, dic*) Recherche une clé d'un dictionnaire à partir de sa valeur

dictlist (*k, v, d*) Ajout d'un valeur dans une liste d'un dictionnaire

Paramètres

- **k** (*str*) – clé dictionnaire
- **v** – valeur à ajouter
- **list[]] d** (*dict[str,*) – dictionnaire

Exemple

```
>>> dictionnaire= {}
>>> dictlist('printemps', 'mar', dictionnaire)
dictionnaire{'printemps', ['mars']}
>>> dictlist('printemps', 'avril', dictionnaire)
dictionnaire{'printemps', ['mars', ''avril']}
>>> dictlist('printemps', 'mars', dictionnaire)
dictionnaire{'printemps', ['mars', ''avril']}
```

inttohex (*v*) Conversion d'une valeur en hexadécimal

Paramètres **v** (*int*) – nombre à convertir

Renvoie valeur en hexadécimal

Type renvoyé str

plain_hex (*hx, s=3*) Complète un chiffre hexadecimal en préfixant une valeur de zéro

Paramètres
- **hx** (*str*) – valeur hexadécimal
- **s** (*int*) – longeur chaine attendu

Type renvoyé str:

Examples

```
>>> hx = '0x129'
>>> plain_hex(hx, 5)
0x00129
```

plain_zero (*v, s*) Complete une valeur chaine de zéro

Paramètres
- **v** – valeur à completer
- **s** – taille chaine attendu préfixé de zerom

Exemple

```
>>> d = 5
>>> plain_zero(d,3)
'005'
```

pop_dic (*l_ids, dic*) Suppression d'une liste d'éléments d'un dictionnaire
:param list[str] l_ids : liste de clé à supprimer :param dict[str:object] dic: dictionnaire à nettoyer

print_err (**args, **kwargs*) Ecriture sur le flux erreur de la console

Paramètres
- **args** – arguments 1
- **kwargs** – arguemnts2

pwd_maker (*i_size=8*) Génération d'un password respectant les regles de password

Conditions
- Une majuscule
- Une minuscule
- Un chiffre
- Un carectère spécial (@#!?$&-_ autorisé)

Paramètres **i_size** (*int*) – Nombre de caracteres de la chaine

Renvoie Mot de passe

str_dic (*chaine*) Convertion d'une chaine en dictionnaire

Paramètres **chaine** (*str*) –

Type renvoyé dic

Exemple

```
>>> s_dic = "{'key':value}"
>>> str_dic(s_dic)
{'key': 'value'}
```

string_me (*v*) Convertion d'une valeur en chaine

Paramètres **v** – valeur à convertir

Type renvoyé str, None en cas d'erreur

2.2 Module complémentaire

pathfile : dreamtools/features.py

test_http_link (*url*) Vérifie une url et renvoie l'url valide

Paramètres **url** – url à évaluer
Rtype str

url_join (*domaine*, *page*) Generation d'une url

Traitement sur dates

3.1 Module de Gestion des date

Liste de fonction pour utilisation des dates pathfile : dreamtools/tools

3.1.1 Constantes globales

Liste des jours de la semaine I_MON, I_TUES, I_WED,I_THU, I_FRI, I_SAT, I_SUN = 1, 2, 3, 4,5,6,0

3.1.2 Fonctions

date_add_workday (*dte*, *nb*) Ajoute un nombre de jours ouvrés donnés à une date

Paramètres
- **dte** (*datetime*) – date de référence
- **nb** (*int*) – nombre de jour à additionner (valeur négative/positive)

Renvoie date de depat + nombre de jours

date_dayed (*dte=None*, *b=True*) Positionne la date indiquée à minuit au matin ou au soir

Paramètres
- **dte** (*datetime*) – Date
- **b** (*bool*) – Date debut de jour (00:00:00.000) ou date de fin de journee date du jour + 1 (minuit) soit lendemin à 00

Example

```
>>> date_dayed()
datetime.datetime(2020, 12, 19, 0,
0,datetime.datetime(2020, 12, 19, 0, 0)
```

date_rss (*dte=None*) Dtate au format RSS

dateadd (*dte*, *nb*, *fm='d'*) Ajoute un nombre de jours données à une date

Paramètres

Type renvoyé datetime

datepaques (*y*) Dates Pâques d'une année donnée

- Lundi de paque : lundi suivant le dimanche de paque (La Pâque)
- Jeudi de l'ascension : 3 jour après paques

- pentecote : 49 jours après le lundi de paques

Paramètres **y** (*int*) – année de référence

Type renvoyé list[date]

datestr (*dte=None, fm='%Y-%m-%dT%H:%M:%S'*) Convertit une date en chaine selon un format donnée

Paramètres
- **dte** (*datetime*) – date à convertir date du jour par défaut
- **fm** (*str*) – format désirée, defaults to "%d/%m/%Y"

Renvoie Renvoie un chaine correspondant au format date passé en parametre

Type renvoyé str

Exemple

```
>>> d = maintenant ()
>>> datestr (d,
'%d.%m.%Y')
02.06.2019
```

datetime_from_local_to_utc (*utc_datetime*) Convertie une date et heure local en heure utc

Paramètres **utc_datetime** (*datetime*) – datetime local

Renvoie datatime utc

datetime_from_utc_to_local (*utc_datetime*) Convertie la date et heure donné (utc) en date local

Paramètres **utc_datetime** – datetime utc

Renvoie date locale

day_in_hour (*dy*) Convertion d'un nombre de jours en heure

Paramètres **dy** (*int*) – nombre de jours

Type renvoyé int

day_in_sec (*dy, ml=False*) Convertion d'un nombre de jours en secondes ou milisecondes

Paramètres
- **dy** (*int*) – nombre de jours
- **ml** (*bool*) – en millisecondes si True sinon en secondes, dafault False

Renvoie (milli) secondes

dtecompare (*dtea, dteb*) Compare si dateb es supérieur à la datea

Paramètres
- **dtea** (*datetime*) – date à comparer
- **dteb** (*datetime*) – date à comparer

Retur 0 si les dates sont égale, -1 si la dateb est antérieur à datea 1 le c&s inverse

Type renvoyé int

dtediff (*dtea*, *dteb*) Calcul du nombre de jours entre deux dates

Paramètres
- **dtea** (*datetime*) – date à comparer
- **dteb** (*datetime*) – date à comparer

Type renvoyé int

dtets (*dte=None*) Conversion date - timestamp

Paramètres **dte** (*date*) – date à convertir

Return int date en miliseconde (sans les ms)

fullmonth (*dte*) Renvoie la date du jour au format MOIS YYYY

Paramètres **dte** (*datetime*) –

Type renvoyé str

get_date (*p_year*, *p_month*, *p_day*) Generationd'une date a partir des valeur numerique

Paramètres
- **p_year** (*int*) – année
- **p_month** (*int*) – mois
- **p_day** (*int*) – date jour

Type renvoyé datetime

get_time (*dte*) Renvoie d'une date au format time :param datetime dte: :rtype: time

get_weeks_num (*dte=None*) Renvoie le numéro de la date indiqué (now par deafut)

is_workday (*dte*) Determine si la date est un jour ouvré ou vaqué (week-end / fériés)

Paramètres **dte** – date à évaluer

Renvoie renvoie le statut jour ouvré (true=ouvré)

Rtype bool

isotodate (*dte*) Conversion str_iso - date
Format ISO : YYYY-MM-DDTHH:MN :param dte:
Renvoie

jours_feries (*y=None*) Jour fériés pour une date donnée

Paramètres **y** (*int*) – Année de référence (optionnel), default : année en cours

Renvoie un tableau de date de jours fériés

Exemple

```
>>> jours_feries ()              #Jours fériés
année en cours
>>> jours_feries (2018) # jours fériés année 2018
```

maintenant (*utc=False*, *fm=None*, *tz=None*) Date et heure de l'instant (Now)

Paramètres
- **utc** (*bool*) – Si True renvoie de l'heure UTC (GMT) ou l'heure local
- **fm** (*str*) – format date : fm == FRM_ISO (iso) | FRM_TIMESTAMP (ts)

- **tz** (*timezone*) – Timezone, None by def

Type renvoyé datetime | string

Exemple

```
>>> maintenant ()
datetime.datetime (2019, 06, 02,
 17, 30, 43, 248622)
>>> maintenant (True)
'2019-06-02T17:30:43.248622'
```

set_timezone (*dt, tz=<UTC>*) Applique la timezone indiquée à la date passée en parametre

Paramètres
- **dt** (*datetime*) – date
- **tz** (*timezone*) – timezone

strdate (*dte, fm='%d-%m-%Y %H:%M:%S'*) Conversion string - date

Paramètres
- **dte** (*str*) – date
- **fm** (*str*) – formayt, default "%d-%m-%Y %H:%M:%S": patterne, optional

Renvoie Renvoie la date convertit ou None en cas d'invalidité (date non conform)

Type renvoyé datetime

Exemple

```
>>> s = '24-02-1976 16:45'
>>> strdate (s, '%d-%m-%Y
%H:%m')
datetime.datetime(1976,02,24,16,45)
```

timeadd (*dte, nb*) Ajoute un nombre d'heure données à une date

Paramètres
- **dte** (*date*) – date de départ
- **nb** (*int*) – nombre de jour à additionner (valeur négative/positive)

Renvoie date de depat + nombre de jours

today (*fm='%d/%m/%Y'*) Renvoie la date du jour

Paramètres **fm** (*str*) – Format de la date attendu

Type renvoyé str

Exemple

```
>>> today ()
'02/06/2019'
>>> today('%d.%m.%Y')
'02.06.2019'
```

tsdate (*ts*) Conversion timestamp - date

Paramètres **ts** – temps en milliseconde depuis 1970

Renvoie date

tstring (*ts, fm='%Y.%m.%d-%H:%M (%a)'*) Conversion timestamp - chaine(str)

Paramètres
- **ts** (*int*) – timestamp

- **fm** (*str*) – format attendu

Type renvoyé str

utcnow_iso ( ) Date et heure actuelle utc au format iso :return: date utc

utcnow_ts ( ) Date et heure actuelle utc au format timestamp :return: timestamp

Traitement sur repertoires et sur fichier

clean_directory (*directory*, *pattern='*'*) Supprimes tous les élements d'un repertoire

Paramètres
- **directory** (*str*) – chemin du repertoire
- **pattern** (*string*) – patter des fichier à supprimer (filtre)

Return int nombre de fichier supprimer

dirparent (*path*) Renvoie du repertoire parent

Paramètres **path** (*str*) – repertoire

Type renvoyé str

dirparser (*directory*, *pattern='*'*) Récupération des fichiers d'un répertoire

Paramètres
- **directory** (*str*) – repertoire
- **pattern** (*str*) – "*" pour tous type de fichier par défaut

Exemple

```
>>> directory =
'C:\Users\public\Documents'
>>> pattern='*.txt'
>>> for filename, path_file in
dirparser(directory, pattern):
...     print(path_file)
'C:\Users\public\Documents\fichier.txt'
'C:\Users\public\Documents\autre_fichier.txt'
```

dirproject (*source=None*) Répertoire pour le fichier en cours

Type renvoyé str

file_exists (*fp*) Vérifie l'existance d'un fichier

Paramètres **fp** (*str*) – filepath
Rtype bool

file_ext (*ps_file*) Retrourne l'extension d'un fichier

Paramètres **ps_file** –

Renvoie Extension de fichier

file_ext_less (*ps_file*) Retrourne le fichier sans extension

Paramètres **ps_file** –

Renvoie Extension de fichier

```
>>> f = 'filename.ext'
>>> file_ext_less(f)
filename
```

fileloader (*fp, b=False*) Chargement d'un fichier :param fp: filepath données à enregistrer :param bool b: données en byte ?, optional :return:

filerecorder (*doc, fp, b=False*) Enregistrement d'un fichier
:param dict|list d:données à enregistrer :param str fp: filepath :param boolean b: données en byte ?, optional :return:

makedirs (*path*) Création du répertoire données

Paramètres **path** – chemin du répertoire à créer
Rtype bool

path_build (*directory, ps_complement*) Construction d'un pathfile

Paramètres

- **directory** (*str*) – repertoire
- **ps_complement** (*str*) – complement permettant de generer le chemin

Type renvoyé str

Exemple

```
>>> path =
'c:\Users\public\directory'
>>> path_build(path, '..\other_dir')
'c:\Users\public\other_dir'
```

remove_file (*p*) Suppression d'un fichier si existant

Paramètres **p** (*str*) – chemin complet du fichier à supprimer

Chapitre 5

Module de validation

Validation de formulaire.

Note Les shémas de validation sont sauvegarder dans le dossier de configuration validators Les schéma de normalization dans normalizator

Gestion de mailing

Chapitre 6

Traitement d'image

TYPE_IMG_GIF = 'GIF' Class permettant le traitement d'une image png convertit en jpg avec prise en charge de la transparence (fond blanc)
pathfile : dreamtools/pyimaging

class CImagine (*src*, *dest*, *with_ext=False*)

classmethod imgrecorder (*fs*, *fp*) Enregistremetn d'une image uploaded (byte)

Paramètres

- **fs** (*file*) – filestream from flask request
- **fp** (*str*) – filepath d'enregistrement

Renvoie

protected (*artist*, *description*) Ajoute un nom d'artist et le copyright d'une image

resize (*s=None*, *mn=None*, *mx=None*) Redimensionnement de l'image au format jpg

Paramètres

- **int) s** (*tuple(int,*) – si indiqué, (w,h) de redimensionnem
- **mn, optional** (*int*) –

taille maximum (carré), default 250

- **mx, optional** (*int*) – taille maximum de l'image, default 250

Renvoie

thumbed (*s=None*) Thumb Image

Paramètres **int] s** (*tuple[int,*) – taille image, defaul (200, 200à

classmethod treat_dir (*s*) Redimensionne toutes les images contenu dans un répertoire donné + thumb :param s:

classmethod treat_img (*s,f*) Enregistre une image, la reformat + thumb :param s: :param f:

property white_background Ajout d'un fond transparent :return:

Module de cryptage

pathfile : dreamtools/tools

Important Module utilisable uniquement sous Linux. Pas de prise en charge du module crypt par Windows

Note Une clé privé et un grain de sel doivent être défini dans le fichier « de parapetre d'application »

- **PROJECT_DIR/cfg/.app.yml**
- SECRET_KEY
- SALT_EXT

Méthode : sha-512`*

Exemple

```
>>> from krypt import CKrypting
>>> pwd = CKrypting.encrypt("Mon mon de passe en clair")
>>> if CKrypting.compare(pwd, 'Mon mot de passe en claire'):
>>>     print ('Saisie invalide')
>>> else:
>>>     print ('Bienvenue')
```

Gestion configuration

Gestion fichiers de configurations (YAML)
pathfile : dreamtools/cfgmng.py

9.1 Repertoires par défaut

Note

- PROJECT_DIR/cfg/PROJECT_DIR/cfg/.log.yml : Fichier de configuration des logs
- PROJECT_DIR/cfg/.app.yml : Fichier de configuration de l'application
- PROJECT_DIR/cfg/categorie.yml : Fichier de liste définie par un code et un libelle
- PROJECT_DIR/cfg/mailing.yml : Fichier de mails préparés
- PROJECT_DIR/cfg/validators.yml : Fichier de validation(cf CERBERUS)
- PROJECT_DIR/cfg/normalizor.yml : Fichier de normalization(cf CERBERUS)

9.2 Class CFBases

class `CFGBases` Cette class permet de gere des fichiers de configuration disponibles dans le repertoire <PROJET_DIR>/cfg

static `app_cfg` (*code=None*) Parametres application

Paramètres `code` (*str*) – clé a retourner (filtre)

Renvoie Configuration

static `categorie_lib` (*code=None*) Liste de definition

Paramètres `code` (*str*) – référence du de la liste

Renvoie liste(s) de categories

Type renvoyé dict

static `loadingbyref` (*filename*, **args*, ***kwargs*) Récupération des parametres de configuration du fichier <filepath> section <r>

Paramètres

- `filepath` (*str*)

– Fichier de configuration

- **code** (*str*) – référence parametres à récupérer, optionnel
- **mode** (*str*) – bytes par defaut

Renvoie configuration | None

static logs_cfg ( ) Configuration des logs

Exemple

```
>>> import import logging.config as
log_config
>>> import logging
>>>
log_config.dictConfig(CFGBases.logs_cfg())
>>> tracker =
logging.getLogger('PROD|TEST')
>>> tracker.info("Exemple dun message
d'information")
```

static mailing_lib (*code*) Mail préparé

Paramètres code (*str*) – référence du mail à envoyer

Renvoie mail

static normalizor ( ) Parametres de normalisation de formulaire :return: parametres de normaisation :rtype: dict

static validator ( ) Parametres de validation de formulaire

class CFGEngine cfg engine

static initial (*baz_dir='cfg'*) **Paramètres baz_dir** (*str*) – **Renvoie**

static loading (*p*, *ref=None*, *m='r'*) Récupération des parametres de configuration du fichier <p> section <r>

Paramètres

- **p** (*str*) – Fichier de configuration
- **ref** (*str*) – référence parametres à récupérer, optionnel
- **m** (*str*) – bytes par defaut

Renvoie configuration | None

static save_cfg (*d*, *f*, *m='w'*) **Paramètres**

- **list(str)) d** (*dict(str,*) – données à enregistrer
- **f** (*str*) – nom du fichier
- **m** (*str*) – default (write): mode « w | a », optional

Renvoie

Gestions des logs

- Configuration des logs
- Traitement des erreurs et des exceptions

pathfile : dreamtools/logmng.py

exception `CError` (*message*, *status*, *title='ERRCustom'*) Gestion des erreurs et traitement des exceptions personnalisées

10.1 Class CReponder

class `CReponder` (*status=200*, *data=None*, *msg=None*)

property `ok` Renvoie le status de la réponse

Type renvoyé bool

property `response` **Renvoie** {« message »: self.message, « data »: self.-data, "status_-code": self._sta-tus}

10.2 Class CTracker

Module de gestion des logs :

- Récupération des logs, traitement
- Execution sécurisé

class `CTracker`

static `alert_tracking` (*msg*, *title*, *code=0*) Message d'alerte (WARNING)

Paramètres

- `msg` (*str*) – message à ecrire dans logs
- `title` (*str*) – Titre ou référence associé au message

- **code** (*int*) – Code numérique

static config (*mode='PROD'*) Initialisation du gestionnaire de log à partir de la configuration enregistré

> **Avertissement** La configuration doit être configurer dans le fichier <PROJECT_DIR>/cdg/.log.yml

Exemple

```
>>> CTracker.config()
Configuration mode PRODUCTION
>>> CTracker.config("DEBUG")
Configuration mode debug
```

static critical_tracking (*msg*, *title*, *code=0*) Message dcritique (CRITIQUE)

Paramètres
- **msg** (*str*) – message à ecrire dans logs
- **title** (*str*) – Titre ou référence associé au message
- **code** (*int*) – Code numérique

static error_tracking (*msg*, *title*, *code=500*) Message d'error (ERROR)

Paramètres
- **msg** (*str*) – message à ecrire dans logs
- **title** (*str*) – Titre ou référence associé au message
- **code** (*int*) – Code numérique

static exception_tracking (*ex*, *title*) Récupération et traitement des exceptions

Paramètres
- **ex** – Exception
- **title** (*str*) – Information

static flag (*trace*) Permet de pointer la dernier action

Paramètres **trace** (*str*) – Action à enregistrer

static fntracker (*fn*, *action*, **args*, ***kwargs*) Execution « securisé » d'une fonction avec gestions des erreurs

Paramètres
- **fn** – fonction a executer
- **action** – Titre de l'execution pour

tracabilité

- **args** – argument de la fonction
- **kwargs** – parametres supplementaire (status par defaut en cas de reussite)

Type renvoyé *CReponder*

Exemple

```
>>> from
dreamtools.logmng
import CTracker
>>> def fn(arg)::
>>>     return
int(arg)
>>> r =
CTracker.fntracker(fn,
 'Test de
convertion int',
'j')
>>> r.response
{'message':
"invalid literal
for int() with
base 10: 'j'",
'data': None,
'status_code':
500}
>>> r =
CTracker.fntracker(fn,
 'Test de
convertion int',
'589321')
>>> r.response
{'message': None,
'data': 589321,
'status_code':
200}
```

static info_tracking (*msg, title, code=0*) Message d'info (INFO)

Paramètres

- **msg** (*str*) – message à ecrire dans logs
- **title** (*str*) – Titre ou référence associé au message
- **code** – Code numérique

static msg_tracking (*msg, title, log_level=20, code=0*) Tracking message

Paramètres

- **msg** (*str*) – message à ecrire dans logs
- **title** (*str*) –

Titre ou référence associé au message

- **log_level** (*int*) – LOG LEVEL Niveau de l'alert (DEBUG | INFO | WARN |)
- **code** (*int*) – Code numérique

- *Index*
- *Index du module*
- *Page de recherche*

d

A

C

D

U

V

W