

web2py T2/T3 Cheatsheet - <http://web2py.appspot.com/t3>

Install T2 as 'plugin_t2' application via web2py gui already included in T3

In model:

```
from applications.plugin_t2.modules.t2 import T2
t2=T2(request,response,session,cache,T,db,all_in_db=False)
- these fields in user-defined tables are automatically stamped (not
created) by CRUD: created_by_ip , created_on , created_by ,
created_signature , modified_by_ip , modified_on , modified_by ,
modified_signature
```

state variables:

t2.person_id id of the person logged in, or None
t2.person_name name/alias of the person logged in, or None
t2.person_email email address of the person logged in, or None
t2.error_action action to call in case of error, defaults to 'error'
t2.now current date-time
t2.request web2py request object
t2.response web2py response object
t2.session web2py session object
t2.cache web2py cache object
t2.T web2py translation object
t2.db database to be used by T2
t2.all_in_db True is binary data and sessions go in DB
t2.is_gae True is running on GAE
t2.id value of request.args[-1] if request.args else 0
t2.logged_in True is the person is logged in else False
t2.my_groups list of group_ids the logged in person is member of

auth methods:

t2.register(verification=False,sender="",next='login',onaccept=None)
t2.login(next='index',onlogin=None)
t2.logout(next='index')
t2.verify(next='login') - registration verification link controller action
t2.requires_login(next='login') - controller method decorator

```
def register(): return dict(form=t2.register())
def login(): return dict(form=t2.login())
def logout(): t2.logout()
def verify(): t2.verify()
@t2.requires_login()
def private_page(): return dict()
```

utility methods:

t2.action(f=None,args=[],vars={}) - makes a url
t2.redirect(f=None,args=[],vars={},flash=None)

```
t2.action('index') == URL(r=request,f='index')
t2.redirect('index') == redirect(URL(r=request,f='index'))
```

view helpers:

t2.include() - includes required CSS and JS files, use in <head>
t2.menu(menu,style='h') - makes menu

```
<html><head>{{=t2.include()}}</head>
<body>{{=t2.menu(response.menu)}} content</body> </html>
```

download control:

t2.download() - serve/stream files (filename in request.args[0]), works even if the data is in db.

```
def download(): return t2.download()
```

CRUD controls:

t2.create(table,next=None,vars={},onaccept=None) - form to create a db record
t2.update(table,query=None,next=None, deletable=True, vars={},onaccept=None, ondelete=None) - form to update a db record, prevents concurrent updates
t2.delete(table,query=None,next=None) - form to delete a db record
t2.read(table,query=None, limitby=None,orderby=None)
t2.display(table,query=None) - form to display a db record
- table is the table to work on
- query is the subset of the table to act on. If None uses t2.id
- next is the action to go to after finishing. If None the same action is rendered again
- vars is a dictionary of optional variables inserted into the form.vars before accepting
- onaccept /ondelete are optional callback functions, only argument is form object itself

```
def create_form():
    return dict(form=t2.create(db.mytable))
or directly in the view
{{=t2.create(db.mytable)}}
similar usage for update and display controls
```

list and search:

t2.itemize(*tables,**opts)
t2.search(*tables,**opts) - search widget, calls itemize internally. can optionally specify a query as additional required condition
- arguments are: table objects, optional named arguments (query, orderby, limitby)

```
{{=t2.search(db.mytable)}}
{{=t2.itemize(db.mytable,query=db.mytable.id>0,orderby=db.mytable.id)}} - displays all records that match the query. If no limitby is specified, this control paginates the items, 25 per page.
If more than one table are specified, they perform an INNER JOIN.
The items are represented, by default, by the first field in the table.
This can be changed by
db.mytable.represent=lambda row: '%s %s' % (row.myfield,row.id)
```

additional controls:

t2.email(sender,to,subject='test',message='test') - send emails. Works on GAE too.
t2.attachments(table,readable=True,writable=True,deletable=False)
t2.comments(table,moderated=False,readable=True,writable=True,deletable=False,hide=True)
t2.reviews(table,status='approved',readable=True,writable=True,deletable=False,subtitle="You Review")

membership methods:

people have membership in groups
t2.add_membership(person_id,group_id,membership_type='default') - adds person_id to group_id
t2.del_membership(person_id,group_id,membership_type='default') - deletes person_id from group_id
t2.have_membership(group_id,membership_type='default') - logged in person is in group_id?
t2.my_memberships() - returns list of groups of logged in user

access methods: groups have access to resources (tables or table records)

t2.add_access(table,record_id=0,access_type='default',group_id=None) - give access to table/record to group_id
t2.del_access(table,record_id=0,access_type='default',group_id=None) - remove access to table/record from group_id
t2.have_access(table,record_id=0,access_type='default') - logged in user has access to table/record?

widgets: override the way a field is rendered in a create/update form.

t2.rating_widget(value,callback=None) - used by t2.ratings
t2.tag_widget(value,tags=[]) - allows multiple values to be stored in a field

```
db.mytable.myfield=T2.tag_widget(['red','green','blue'])
Values are stored as "[red][green]".
```

special table and field attributes:

db.mytable.myfield.label is the label to be used in forms.
db.mytable.myfield.comment is the comment to be shown in the third column of create/update forms.
db.mytable.exposes is a list of fields that should appear in create/update forms (None means all fields).
db.mytable.displays is a list of fields that should appear in display forms (None, means all fields).
db.mytable.represent is a lambda function that takes a row for the table and return a representation for that row.

Google Checkout (experimental) :

t2.clear_cart() - clears the cart (session.t2.cart).
t2.add_to_cart(name,price,quantity=1,description="",weight=0,height=0,length=0,depth=0,currency='USD',weight_unit='LB')
t2.checkout_cart(merchant_id,action_url,button_url,continue_url,attributes={}) - generates a form with hidden buttons and a "Google checkout" button. On pressing the button, the control goes to Google Checkout and the visitor can pay using the credit card. For details look at the Google Checkout docs.

Available T2 components in T3

self.action(), self.add_access(), self.add_membership(), self.add_to_cart(), self.barchart(), self.checkout_cart(), self.clear_cart(), self.comments(), self.coords_by_address(), self.create(), self.del_access(), self.delete(), self.del_membership(), self.display(), self.email(), self._error(), self.have_access(), self.have_membership(), self.itemize(), self.login(), self.logout(), self.my_memberships(), self.profile(), self.rating_widget(), self.read(), self.redirect(), self.requires_login(), self.reset_password(), self.search(), self.tag_widget(), self.update(), self.urlopen(), self.verify(), self.wiki(), self.wiki_menu(), self.wiki_nav()

Global T3 objects:

All Python commands (excepting print, looping, conditional expressions)
All Python modules but need to be imported
response
session
cache (example: cache.ram('key',function,t))
db (example: db.define_table(...))
all web2py helpers