WEB2PY 2.0 Cheat Sheet

`http://web2py.com`

URL Parsing

`http://`*host:port*`/admin` (admin interface)
`http://`*host:port*`/`*app*`/static/`*file* (app static file)
`http://`*host:port*`/`*app*`/appadmin` (database interface)
`http://`*host:port*`/`*app*`/`*c*`/`*f(.e)*`/!`*args*`?`*vars*

host → `request.http_host`
port → `request.http_port`
app → `request.application`
c → `request.controller`
f → `request.function`
e → `request.extension`
args → `request.args` (list)
vars → `request.vars` (dict)
'c/f.e' → `response.view`

Global Objects

`request.`*obj*

`application`, `controller`, `function`, `now`, `client`, `is_local`, `is_https`, `ajax`, `args`, `vars`, `get_vars`, `post_vars`, `env.request_method`, `env.path_info`, `env.query_string`, `env.http_*`, `env.wsgi_*`

`response.`*obj*

`status=200`, `view='filename.html'`, `flash='flash me'`, `js = 'alert("run me")'`, `download(request,db)`, `stream(file)`, `render(template,**vars)`

`session.`*obj*

`connect(request,response,db,separate=False)`, `flash`, `secure()`, `forget()`, `_unlock(response)`

`cache`

```
@cache('key',3600,cache.ram)
@cache('key',3600,cache.disk)
cache.ram.clear(regex='k.*')
```

`T` (internationalization)

`T('hello %(key)s',dict(key='thing'))`
`T.current_languages = ['en']` (no translate)
`T.force('en')` (use languages/en.py)

`URL`, `redirect`, and `HTTP`

`URL('function')`
`URL('controller','function')`
`URL('app','controller','function')`
`URL('function',args=[...],vars={...})`
`URL('function',scheme=True)` (full url)
`URL('function',user_signature=True)`
(then use `@auth.requires_signature()`)
`redirect(URL('index'))`
`raise HTTP(500,'message')`

Database Abstraction Layer

```
db = DAL('sqlite://storage.sqlite',pool_size=1)
db.define_table('thing', Field('name','string'))
id = db.thing.insert(name='max')
query = db.thing.name.contains('m')&(db.thing.id==1)
db(query).update(name='max')
db(query).delete()
things = db(query).select(db.thing.ALL,
    orderby=~db.thing.name, groupby=db.thing.id
    dictinct=True, cache=(cache.ram,60))
thing = db.thing(id) or redirect(URL('error'))
thing.update_record(name='max')
things.export_to_csv_file(open(filename,'wb'))
db.thing.import_from_csv_file(open(filename,'rb'))
```

Field Types

`string`, `text`, `boolean`, `integer`, `double`, `decimal(n,m)`, `date`, `time`, `datetime`, `password`, `upload`, `blob`, `list:string`, `list:integer`, `reference` *table*, `list:reference` *table*

Field Attributes

```
Field(fieldname, type='string', length=None,
   default=None, required=False, requires=None,
   ondelete='CASCADE', notnull=False, unique=False,
   uploadfield=True, widget=None, label=None,
   comment=None, writable=True, readable=True,
   update=None, authorize=None, autodelete=False,
   represent=None, uploadfolder=None,
   uploadseparate=False, compute=None, ...)
```

Validators

`CLEANUP`, `CRYPT`, `IS_ALPHANUMERIC`, `IS_DATE`, `IS_DATETIME`, `IS_DATETIME_IN_RANGE`, `IS_DATE_IN_RANGE`, `IS_DECIMAL_IN_RANGE`, `IS_EMAIL`, `IS_EMPTY_OR`, `IS_EQUAL_TO`, `IS_EXPR`, `IS_FLOAT_IN_RANGE`, `IS_GENERIC_URL`, `IS_HTTP_URL`, `IS_IMAGE`, `IS_INT_IN_RANGE`, `IS_IN_DB`, `IS_IN_SET`, `IS_IN_SUBSET`, `IS_IPV4`, `IS_LENGTH`, `IS_LIST_OF`, `IS_LOWER`, `IS_MATCH`, `IS_NOT_EMPTY`, `IS_NOT_IN_DB`, `IS_NULL_OR`, `IS_SLUG`, `IS_STRONG`, `IS_TIME`, `IS_UPLOAD_FILENAME`, `IS_UPPER`, `IS_URL`

Helpers

`A`, `B`, `BEAUTIFY`, `BODY`, `BR`, `CAT`, `CENTER`, `CODE`, `COL`, `COLGROUP`, `DIV`, `EM`, `EMBED`, `FIELDSET`, `FORM`, `H1`, `H2`, `H3`, `H4`, `H5`, `H6`, `HEAD`, `HR`, `HTML`, `I`, `IFRAME`, `IMG`, `INPUT`, `LABEL`, `LEGEND`, `LI`, `LINK`, `MARKMIN`, `MENU`, `META`, `OBJECT`, `ON`, `OL`, `OPTGROUP`, `OPTION`, `P`, `PRE`, `SCRIPT`, `SELECT`, `SPAN`, `STYLE`, `TABLE`, `TAG`, `TBODY`, `TD`, `TEXTAREA`, `TFOOT`, `TH`, `THEAD`, `TITLE`, `TR`, `TT`, `UL`, `XHTML`, `XML`

```
DIV(SPAN('hello'),_id='myid',_class='myclass')
A('link',_href=URL(...))
SPAN(A('link',callback=URL(...),delete='span'))
TABLE(*[TR(TD(item)) for item in [...]])
div = DIV(SPAN('hello',_id='x'))
div.element('span#x').append("world")
div.element('span#x')['_class'] = 'myclass'
DIV('1<2').xml()==DIV(XML('1&lt;2',sanitize=True)).xml()
div = TAG.DIV(TAG.SPAN('hello',_id='x'))
div = TAG('<div><span id="hello">hello</span></div>')
```

Forms

`form = SQLFORM(db.thing,record=None)`
`form = SQLFORM.factory(Field('name'))` (no db)
`form = SQLFORM.dictform(d)` (for `d={...}`)

```
form = SQLFORM(db.thing).process()
if form.accepted: ...
elif form.errors: ...
```

Grids

```
grid = SQLFORM.grid(query)
grid = SQLFORM.smartgrid(table, linked_tables=[])
```

```
SQLFORM.grid(
   query, fields=None, field_id=None, left=None,
   headers={}, orderby=None, searchable=True,
   sortable=True, paginate=20, deletable=True,
   editable=True, details=True, selectable=None,
   create=True, csv=True, links=None, ...)
```

Auth

`@auth.requires_login()`
`@auth.requires_membership('groupname')`
`@auth.requires_premission('edit','tablename',id)`
`@auth.requires(`*condition*`)`
`auth.(has|add|del)_membership(...)`
`auth.(has|add|del)_permission(...)`

Full Example

models/db.py

```
from gluon.tools import *
db = DAL('sqlite://storage.sqlite')
auth = Auth(db)
auth.define_tables()
db.define_table('thing',
    Field('name',requires=IS_NOT_EMPTY()), auth.signature)
auth.enable_record_versioning(db) # for full db auditing
```

controllers/default.py

```
def index(): return auth.wiki() # embed a wiki
def download(): return response.download(request,db)
def user(): return dict(form=auth) # login/etc.

@auth requires_login()
def manage_things(): # access you data
    grid = SQLFORM.grid(db.thing.created_by==auth.user.id)
    return locals()
```

views/default/manage_things.html

```
{{extend 'layout.html'}}
<h1>Your things</h1>
{{=grid}}
{{# any python between double braces}}
```

Generic views

```
generic.html
generic.rss
generic.ics
generic.map # google map
generic.pdf # html -> pdf
generic.json
generic.jsonp
```

Web services

```
from gluon.tools import Service
service = service()
def call(): return service()
@service.rss
@service.xml
@service.json
@service.xmlrpc
@service.jsonrpc
@service.amfrpc3('domain')
@service.soap('name',args={'x':int},returns={'y':int})
@service.run
```

REST

```
@request.restful()
def index():
    def GET(a,b,c): return dict()
    def PUT(a,b,c): return dict()
    def POST(a,b,c): return dict()
    def DELETE(a,b,c): return dict()
    return locals()
```

MARKMIN

```
text = """
# section
## subsection
**bold** ''italic'' ``code``, ``what``:up
-----------------------------------------------
image  | http://example.com/image.jpg
audio  | http://example.com/audio.mp3
video  | http://example.com/video.mp4
iframe | embed:http://example.com/page.html
-------------------------------------:css_class
@{variable} and @{controller/function/args}"""
{{=MARKMIN(text,
    url=True,environment=dict(variable='x'),
    extra=dict(up=lambda t:cgi.escape(t.upper()))))}}
```

Login Methods

```
from gluon.contrib.login_methods.basic_auth import *
auth.settings.login_methods.append(
    basic_auth('http://server'))
```

```
from ....ldap_auth import *
auth.settings.login_methods.append(ldap_auth(
    mode='ad', server='my.domain.controller',
    base_dn='ou=Users,dc=domain,dc=com'))
```

```
from ....pam_auth import *
auth.settings.login_methods.append(pam_auth())
```

```
from ....openid_auth import *
auth.settings.login_form = OpenIDAuth(auth)
```

```
from ....email_auth import *
auth.settings.login_methods.append(
   email_auth("smtp.gmail.com:587","@gmail.com"))
```

```
from ....browserid_account import *
auth.settings.login_form = BrowserID(request,
   audience = "http://127.0.0.1:8000"
   assertion_post_url = 'http://...//user/login')
```

```
from ....dropbox_account import *
auth.settings.login_form = DropboxAccount(request,
    key="...",secret="...",access_type="...",
    url = "http://.../user/login')
```

```
from ....rpx_account import *
auth.settings.login_form = RPXAccount(request,
    api_key="...",domain="...",
    url='http://.../user/login'
```

```
from ....x509_auth import *
auth.settings.login_form = X509Account()
```

Payment Systems

Google wallet button

```
from gluon.contrib.google_wallet import button
{{=button(merchant_id="123456789012345",
   products=[dict(name="shoes",
      quantity=1, price=23.5, currency='USD',
      description="running shoes black")])}}
```

Stripe

```
from gluon.contrib.stripe import Stripe
Stripe(key).charge(amount=100,currency='usd',
    card_number='4242424242424242',
    card_exp_month='5',card_exp_year='2012',
    card_cvc_check='123', description='test charge')
Stripe(key).check(d['id'])
Stripe(key).refund(d['id'])
```

Authorize.Net

```
from gluon.contrib.AuthorizeNet import process
process(card_number,expiration,total,cvv=None,
   tax=None,invoice=None, login='cnpdev4289',
   transkey='SR2P8g4jdEn7vFLQ',testmode=True)
```

Deployment

```
web2py.py -i ip -p port -a password
web2py.py -S app -M -N -R script.py (run script)
web2py.py -S app -M -N (shell)
web2py.py -K app (task queue worker)
anyserver.py -s server (third party server)
```

servers: bjoern, cgi, cherrypy, diesel, eventlet, fapws, flup, gevent, gnuicorn, mongrel2, paste, rocket, tornado, twisted, wsgiref

Apache + mod_proxy

```
sudo aptitude install libapache2-mod-proxy-html
sudo a2enmod proxy
cd /etc/apache2
sudo ln -s mods-available/proxy_http.load \
           mods-enabled/proxy_http.load
```

In VirutualHost:

```
ProxyRequests off
ProxyPass /myapp http://127.0.0.1:8000/myapp
ProxyHTMLURLMap http://127.0.0.1:8000/myapp /myapp
```

Apache + mod_wsgi

```
sudo apt-get install libapache2-mod-wsgi
sudo a2enmod wsgi
```

In VirutualHost:

```
DocumentRoot /path/web2py/
WSGIScriptAlias / /path/web2py/wsgihandler.py
WSGIDaemonProcess web2py user=apache group=web2py \
   home=/path/web2py/ processes=5
<LocationMatch "(/[\w_]*/static/.*)">
  Order Allow,Deny
  Allow from all
</LocationMatch>
<Location ""/">
  Order deny,allow
  Allow from all
  WSGIProcessGroup web2py
</Location>
```

uWSGI

```
hg clone http://projects.unbit.it/hg/uwsgi
cd uwsgi; make -f Makefile.Py27
uwsgi/uwsgi --pythonpath /path/web2py --async 24 -t 20 \
  --ugreen --module wsgihandler -s /tmp/we2py.sock
```