

web2py



Massimo Di Pierro
Associate Professor
School of Computing
DePaul University, Chicago



Contributors



1ks!

Daniel.Lin
Mark.Moore
Sharriff.Aina
Hans.Donner
Nathan.Freeze
Fred.Yanowski
Marcel.Hellkamp
Alvaro.JustenRochaes
Bruno.Rochaes
Chris.May
Eric.Vicenti
Ryan.Seto
Kyle.Smith
Michael.WillisBoris.Manojlovic
Vinicius.Assaf
Olaf.Ferger
Craig.Younkins
Robin.Bhattacharyya
Josh.Jaques
Martin.Hufsky
Andrew.Willimott
Yarko.Tymciurak
Ovidio.Marinho.Falcao.Neto
Francisco.Gama
Brian.Meredyk
Mariano.Reingart
Ovidio.Marinho.Falcao.Neto
Sriram.Durbha
Ondrej.Such
Markus.Gritsch
Francisco.Gama
Paolo.Caruccio
Keith.Yang
Lucas.D'Avila
Tim.Michelsen
Alan.Etkin
Josh.Goldfoot
Anthony.Bastardi
Telman.Yusupov
David.Wagner
Jonathan.Benn
Ruijun.Luo
Zahariash
Martin.Mulone
Chris.Steeli
Gyuris.Szabolcs
Carsten.Haese
Hans.C..v..Stockhausen
Dave.Stoll
Fran.Boon
Jose.Jachhuf
Mark.Larsen
Anders.Roos
Patrick.Breitenbach
Niall.Sweeny
Stuart.Rackham
Simone.Bizzotto
Phyo.Arkar.Lwin
Thadeus.Burgess
Alexey.Nezhdanov
CJ.Lazell
Angelo.Comitini
Graham.Dumpleton
Michele.Comitini
Mateusz.Banach
Arun.K.Rajeevan
Vidul.Nikolaev.Petrov
Yannis.Aribaud
Pierre.Thibault
Marcel.Leuthi
Falko.Krause
Bill.Ferrett
Omi.Chiba
Nicolas.Bruixer
Jose.Vicente.de.Sousa
Denes.Lengyel
Robert.Valentak
Dene.Niccolo.Polo
Gilson.Filho
Jose.Redrejo.Rochaes
Daniel.Lin
ans.Murx

Resources

The screenshot shows the official Web2py website. A yellow circle highlights three mobile application screenshots: a smartphone displaying a weather app and a tablet displaying a news app. A yellow arrow points from this circle to a yellow box labeled "books". Another yellow arrow points from the same circle to a yellow box labeled "videos". The page also features a banner for "InfoWorld's 2012 Technology of the Year Award Winner" and sections for "WEB2PY™ WEB FRAMEWORK", "BATTERIES INCLUDED", "WEB-BASED IDE", and "EXTENSIVE DOCS".

books

videos

web2py.com

The screenshot shows the "examples" section of the Web2py website, featuring a grid of 60+ example applications. Each example includes a thumbnail, a title, and download/browse links. Examples shown include FacebookClone, FacebookConnectExample, FileTreeBrowser, FindRideSharing, GeolocateByIp, ImageGallery, NeedARide, GeoLiteCountry, HotelManagementExample, and KPAX.

60+ example apps

The screenshot shows the Subism website, which is a collection of plugins for Web2py. It features a background image of clouds and network nodes. A yellow box labeled "plugins" is overlaid on the left side. The page includes sections for "Form Widgets" (Horizontal Radio Widget, Multiple Select Widget, Suggest Widget) and a link to "dev.s-cubism.com/web2py_plugins".

plugins

web2py-plugins

A collection of plugins of Web2py, an opensource Python web framework. Here we love to share useful code parts produced by our development with the framework. The code parts are organized in a web2py's plugin system, and easily available.

Form Widgets

Horizontal Radio Widget
A radio widget

Multiple Select Widget
A user-friendly multiple options

Suggest Widget
A refined autocomplete

dev.s-cubism.com/web2py_plugins

The screenshot shows the web2pyslices.com website, which is a social network built with Web2py. It features a large image of a smartphone with a spoon and fork icon, a cartoon character holding a tray, and a login form. A yellow box labeled "recipes" is overlaid on the right side. The page includes a "FEATURED SLICES" section and a "matplotlib howto" post.

recipes

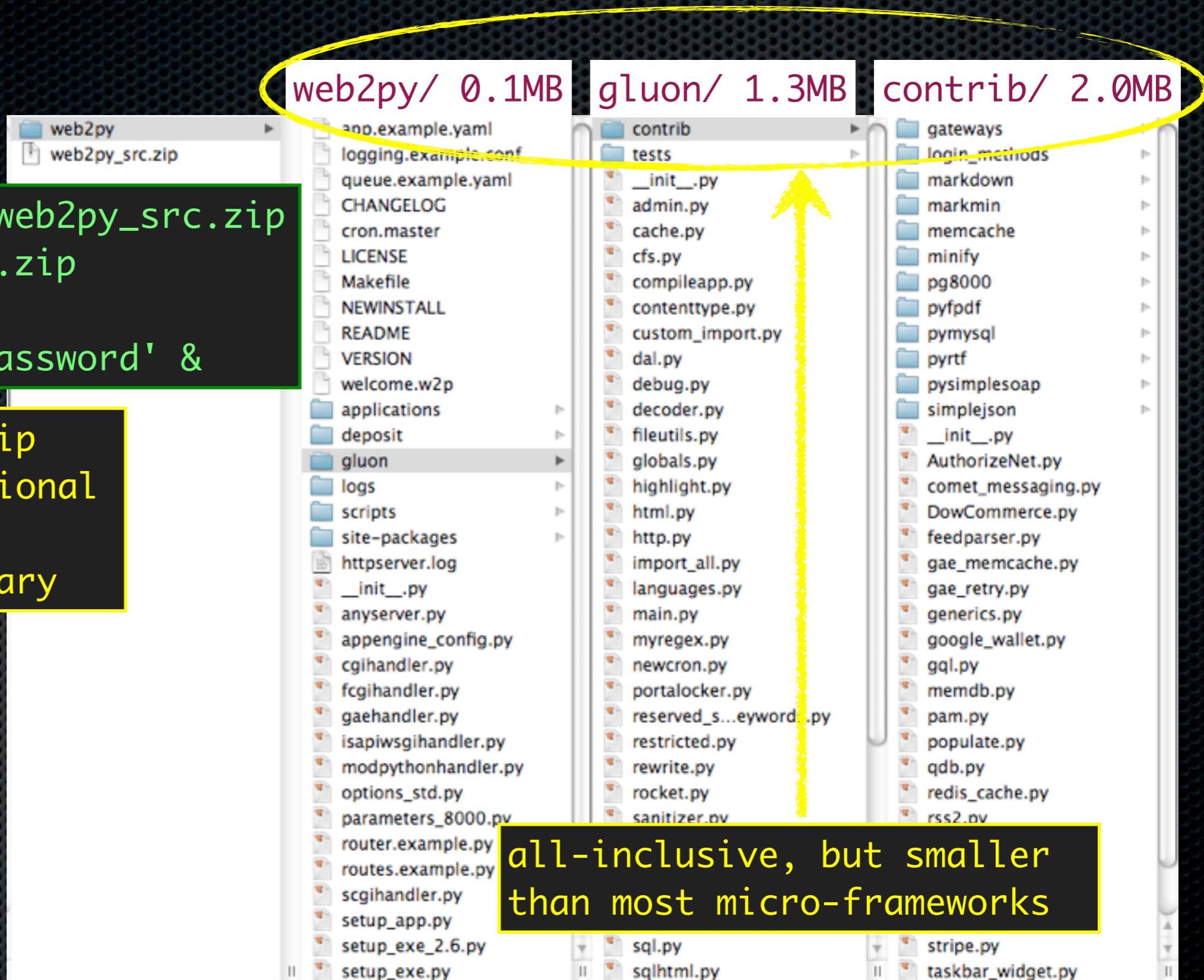
<http://www.web2pyslices.com>
(built with web2py-based social network software: <http://movu.ca/>)

web2py Goals

- Make web development easy and secure
- ... by providing a simple consistent syntax
- ... everything should have a default
- ... all tools in one box
- ... including a web based IDE
- ... takes care of all management issues (no shell)

```
wget http://.../web2py_src.zip  
unzip web2py_src.zip  
cd web2py  
web2py.py -a 'apassword' &
```

download and unzip
installation optional
no configuration
no coding necessary

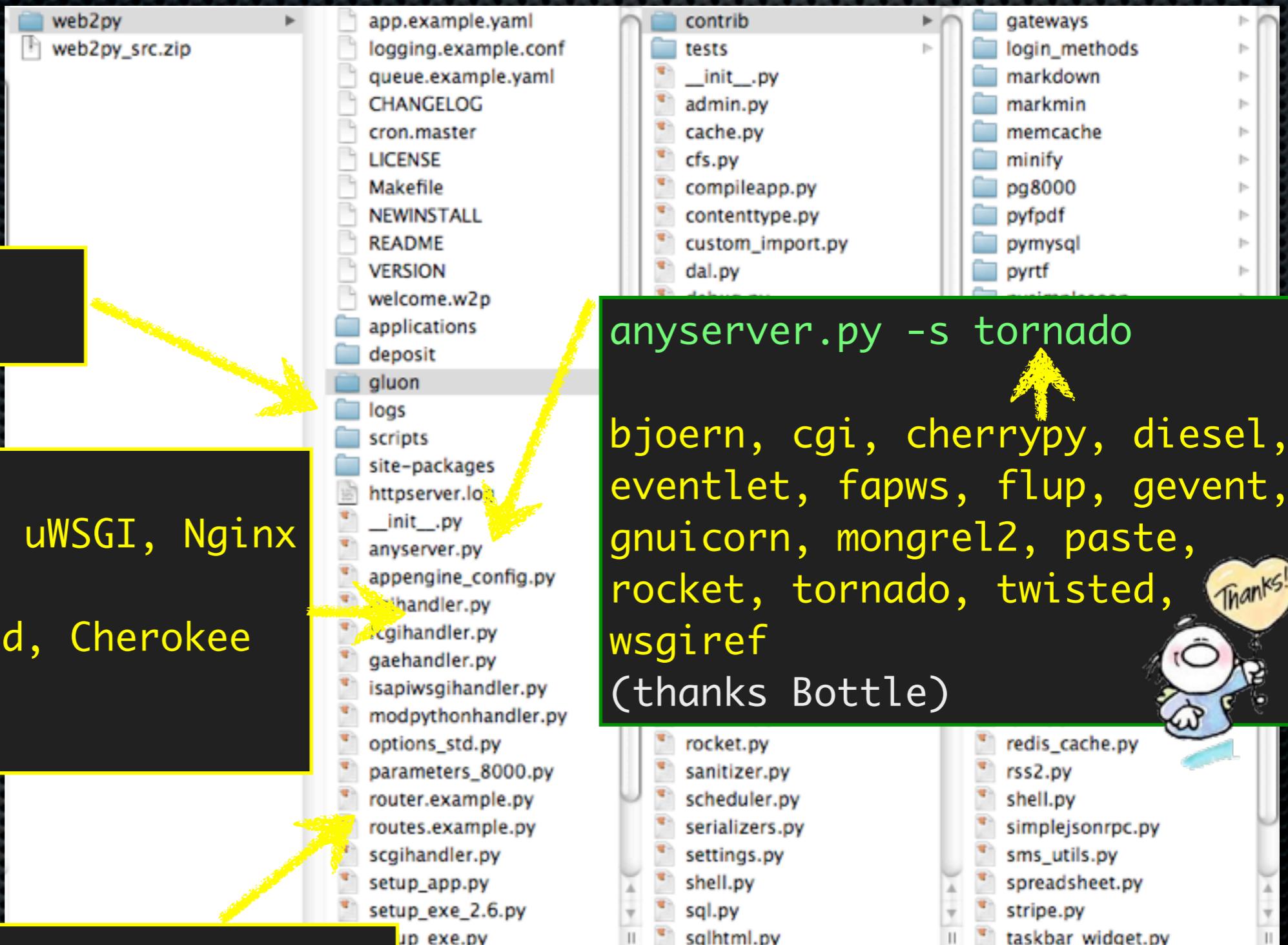


`*.log:`
`log files`

`adapters:`

- WSGI: Apache, uWSGI, Nginx
- CGI
- FCGI: Lighttpd, Cherokee
- GAE
- ISAPI: IIS

`routes.py:`
like `urls.py` but optional



`anyserver.py -s tornado`

bjoern, cgi, cherrypy, diesel,
eventlet, fapws, flup, gevent,
gnunicorn, mongrel2, paste,
rocket, tornado, twisted,
wsgiref
(thanks Bottle)



`dal.py`: Database Abstraction Layer
single file / no dependencies / framework agnostic

supports: sqlite, postgresql, mysql, db2, firebird, mssql,
google datastore, interbase, ingres, imap, ~sapdb,
~cubrid, ~teradata, ~mongodb

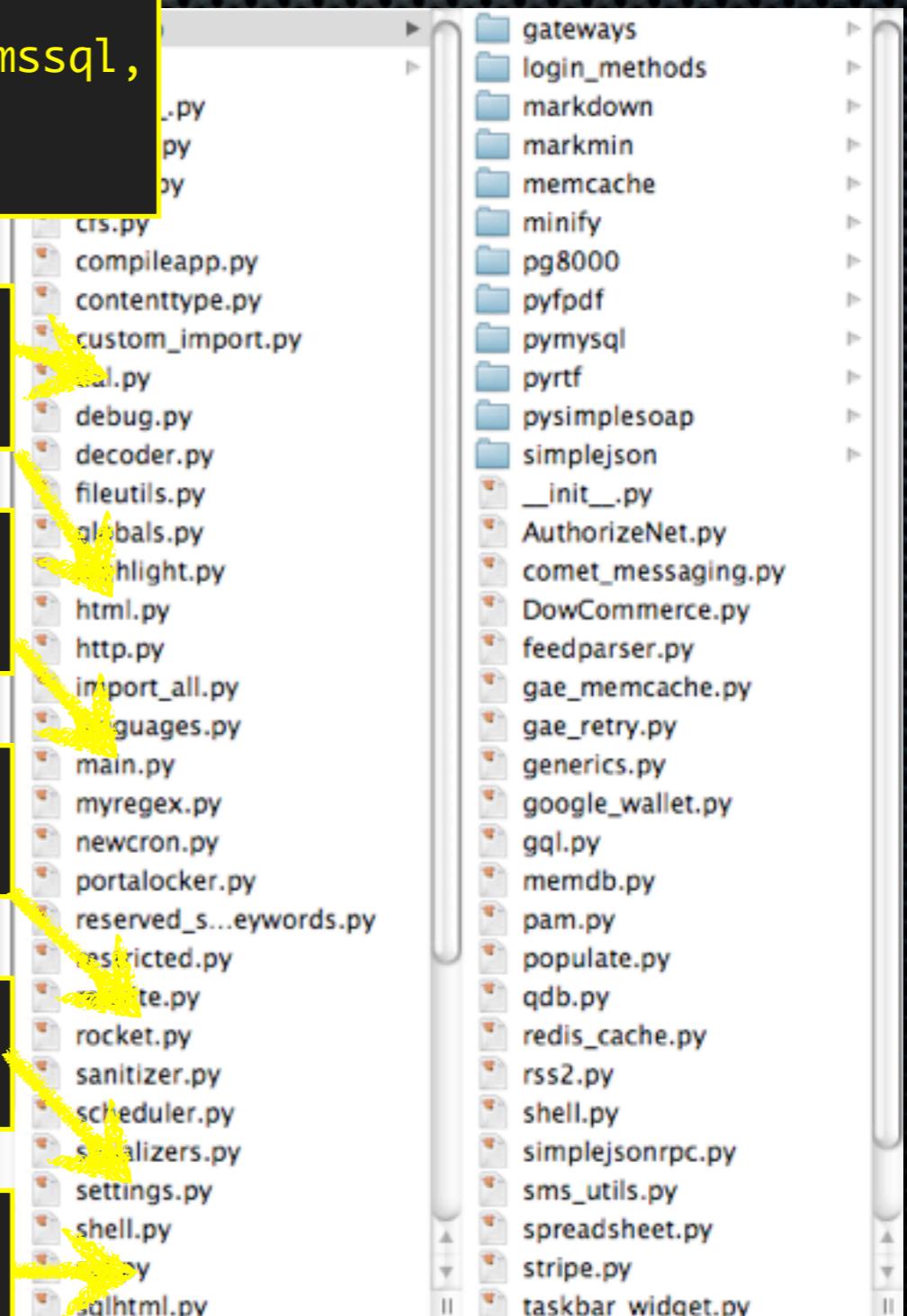
`html.py` & `sqlhtml.py`:
helpers, html parser, form generation

`main.py`:
contains "def wsgibase", the WSGI app

`rocket.py`: ssl-enabled threaded web server
single file / no dependencies / framework agnostic

`template.py`: pure python template language
single file / no dependencies / framework agnostic

`cron.py`: pure python cron
`scheduler.py`: master/worker task scheduler



Document processing:

pyfpdf (html to pdf), pyrtf (generates Rich Text Format)
markdown, MARKMIN (like markdown but better ;-)

Protocols:

rss, feedparser, json, PySimpleSOAP, ...

Caching Adapters:

Redis, Memcache, GAE Memcache

Payment processing:

Authorize.net, Stripe.com, DowCommerce.com

Login Methods:

Oauth 1.0, OAuth 2.0, OpenID, CAS (consumer and provider),
PAM, LDAP, Janrain, x509, LinkedIn, Loginza, Janrain,
gmail, Dropbox, motp

Drivers:

pyMySQL, pg8000 (for postgresql)

other stuff...

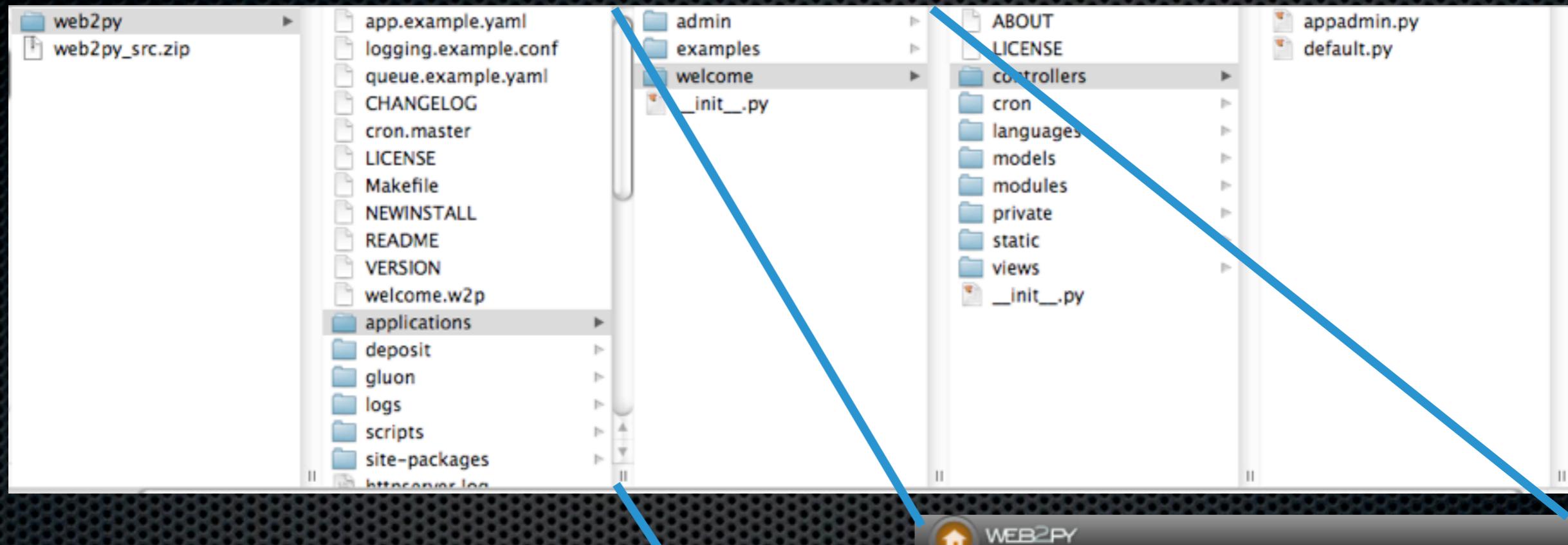


contrib/ 2.0MB

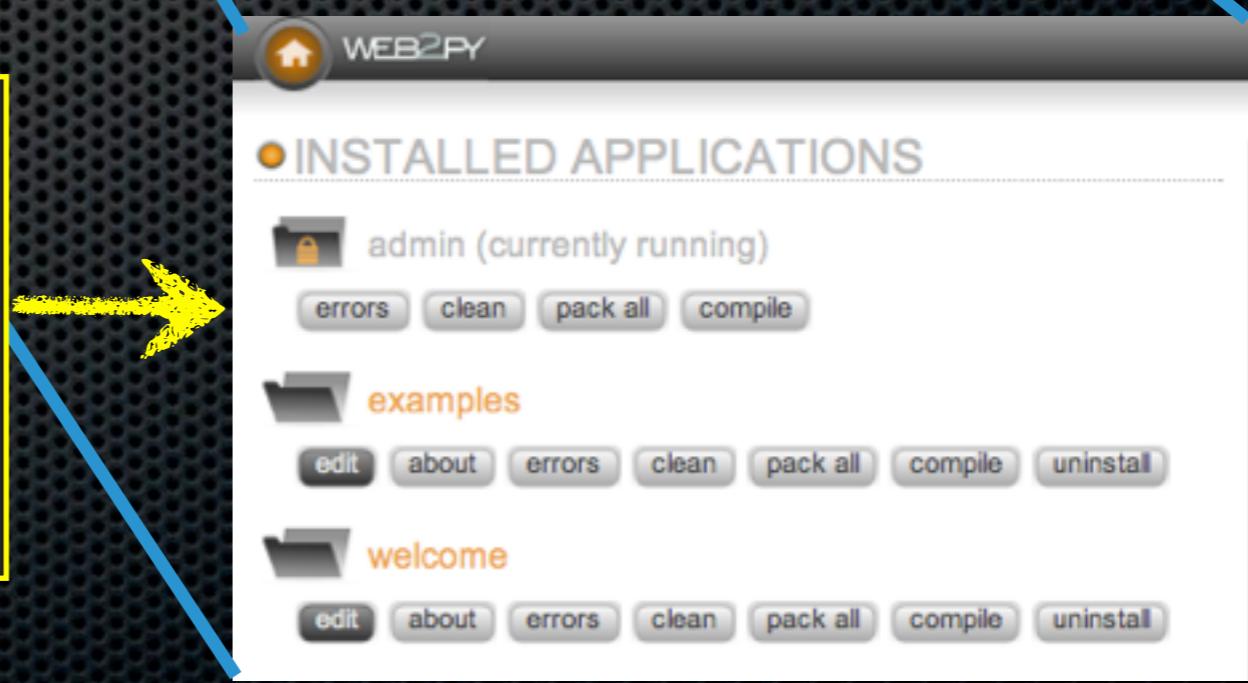
gateways
login_methods
markdown
markmin
memcache
minify
pg8000
pyfpdf
pymysql
pyrtf
pysimplesoap
simplejson
__init__.py
AuthorizeNet.py
comet.messaging.py
DowCommerce.py
feedparser.py
gae_memcache.py
gae_retry.py
generics.py
google_wallet.py
gql.py
memdb.py
pam.py
populate.py
qdb.py
redis_cache.py
rss2.py
shell.py
simplejsonrpc.py
sms_utils.py
spreadsheet.py
stripe.py
taskbar_widget.py

scgihandler.py
setup_app.py
setup_exe_2.6.py
setup_exe.py
settings.py
shell.py
sql.py
sqlhtml.py

Multi-Project



admin: the web based IDE
examples: clone of the web site
welcome: the scaffolding app
...
one web2py can run
many projects
without conflicts





Web Based IDE

The screenshot shows the Web2Py Admin interface at `127.0.0.1:8000/admin/default/site`. On the left, there's a sidebar titled "INSTALLED APPLICATIONS" listing four applications: "admin (currently running)", "examples", "friends", and "welcome". Each application has buttons for "Edit", "About", "Errors", "Clean", "Pack all", "Compile", "Uninstall", and "Disable". On the right, there's a main panel with the following sections:

- "Change admin password" button
- "Version 1.99.7 (2012-03-07 15:25:49) dev" status with "Unable to check for upgrades"
- "Running on Rocket 1.2.4"
- "Reload routes" button
- "New application wizard" section with "Start wizard (requires internet access)" button
- "New simple application" section with "Application name:" input field and "Create" button
- "Upload and install packed application" section with fields for "Application name:", "Upload a package:", "Get from URL:", and "Install" button. A blue arrow points to this section from the "remote install and deploy" callout.
- "Deploy on Google App Engine" section with "Deploy" button. A blue arrow points to this section from the "remote install and deploy" callout.

Annotations on the left side highlight specific features:

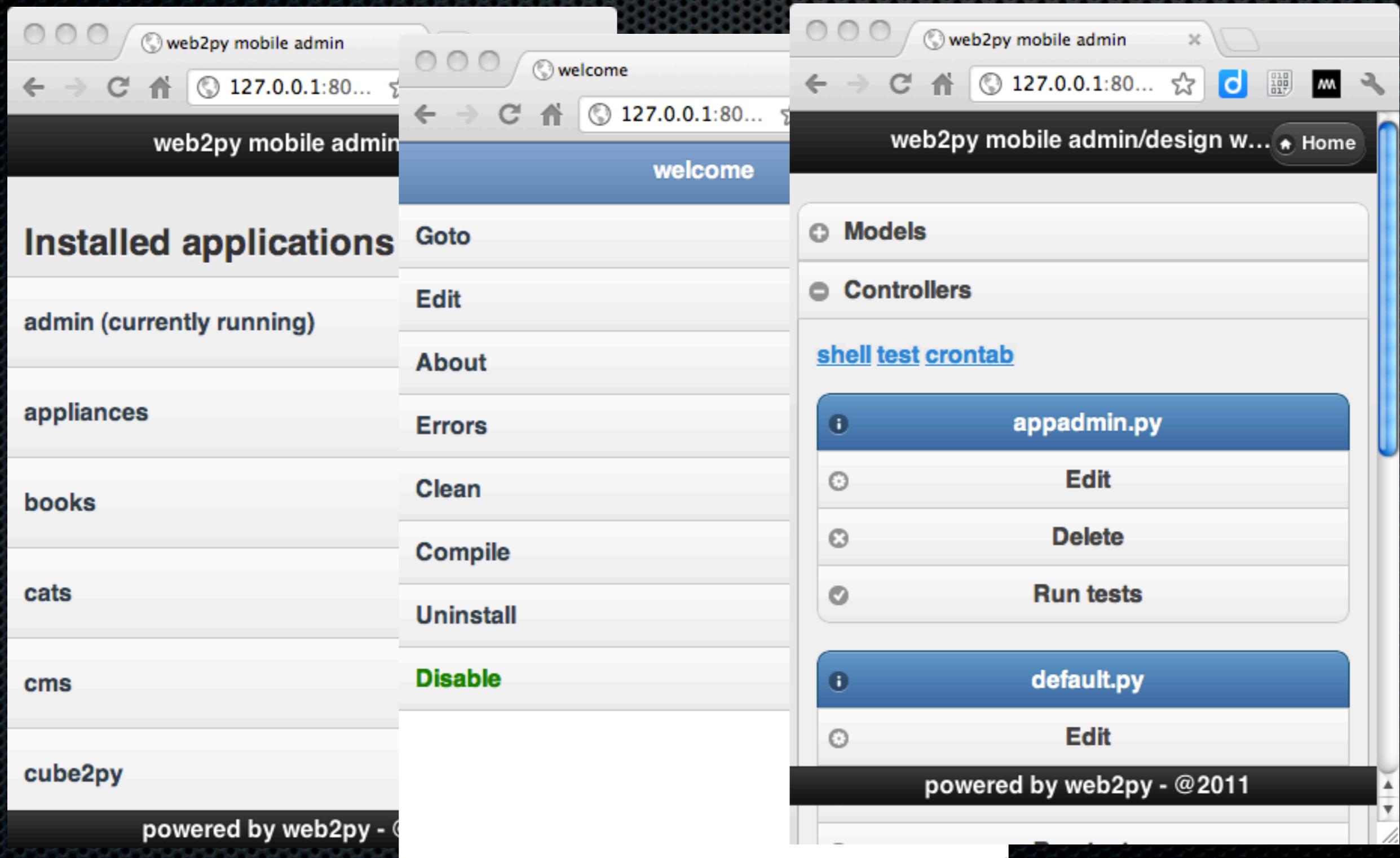
- A yellow box labeled "manage/edit" covers the sidebar area.
- A yellow box labeled "remote install and deploy" covers the bottom right area.

Annotations on the right side highlight specific features:

- A yellow box labeled "wizard" covers the "New application wizard" section.
- A yellow box labeled "create" covers the "New simple application" section.

Web Based IDE

mobile version (based on jQuery mobile)





Web Translation

WEB2PY

- EDITING LANGUAGE FILE "WELCOME/LANGUAGES/
- ORIGINAL/TRANSLATION

"update" is an optional expression like "field1='newvalue'". You cannot update or delete the results of a
"update" è un'espressione opzionale come "campo1='nuovo valore'".
Non si può fare "update" o "delete" dei risultati di un JOIN

[delete](#)

%s rows deleted

%s righe ("record") cancellate [delete](#)

%s rows updated

%s righe ("record") modificate [delete](#)

%Y-%m-%d

%d/%m/%Y [delete](#)

%Y-%m-%d %H:%M:%S

%d/%m/%Y %H:%M:%S [delete](#)

Administrative interface

Interfaccia amministrativa [delete](#)

Tickets

no distinction debug vs production



logs all errors and issues tickets to users by default

The screenshot shows a web interface for managing errors. At the top, there's a navigation bar with a home icon and the text "WEB2PY". Below it, a secondary header says "ERROR TICKET FOR 'A1'". On the left, a sidebar lists "ERROR LOGS FOR 'A1'" with buttons for "check all", "unchecked all", and "delete all checked". A note says "Click row to expand traceback". There's also a "lists by ticket" button. The main content area has a table with columns "Delete", "Count", "File", and "Error". Two rows are shown:

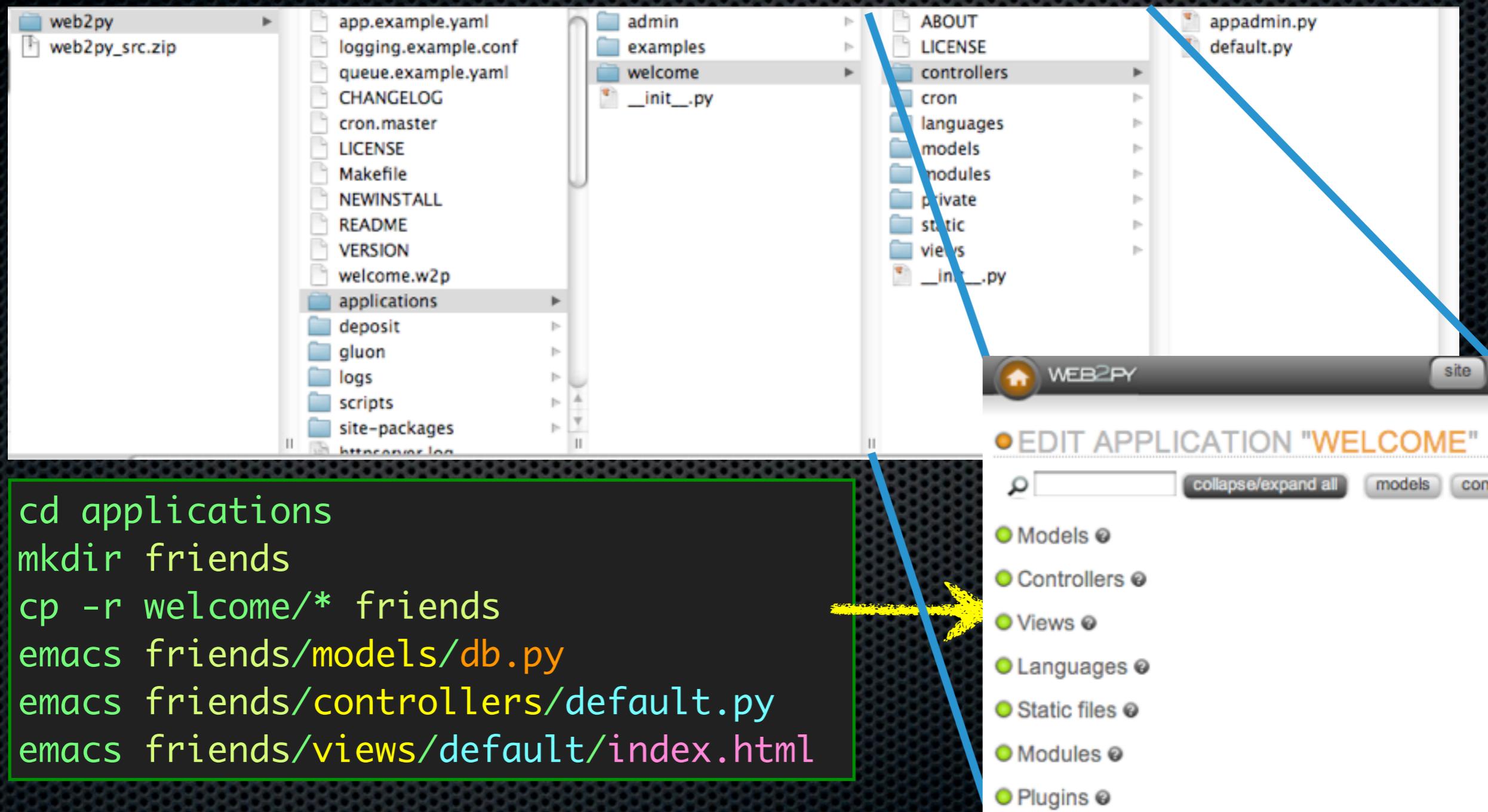
Delete	Count	File	Error
<input type="checkbox"/>	3	default.py	ZeroDivisionError: integer division or modulo by zero + details
<input type="checkbox"/>	1	default.py	NameError: name 'asasdf' is not defined + details

A callout box highlights the text "groups common errors". In the bottom right corner, a separate callout box highlights the word "traceback".

Below the table, a detailed traceback is shown:

```
1. Traceback (most recent call last):
2.   File "/Users/mdipierro/web2py/gluon/restricted.py", line 192, in restricted
3.     exec code in environment
4.   File "/Users/mdipierro/web2py/applications/a1/controllers/default.py", line 12, in <module>
5.     1/0
6. ZeroDivisionError: integer division or modulo by zero
7.
```

Model-View-Controller



shell access or web access or both (no metadata, no restart)

Overall Design



Most Python Frameworks

runme.py

your
program

import

```
import bottle  
  
@bottle.route('/index')  
def index():  
    return 'hello world'  
  
bottle.run()
```

framework
libraries

web2py

web2py.py

cron

scheduler

```
from gluon.rocket import Rocket  
from gluon.main import wsgibase  
  
Rocket(app_info =  
    {'wsgi_app':wsgibase}).start()  
  
+ routes + session + cache + ...
```

execute

your
program

```
def index():  
    return "hello world"
```

"do-not-repeat-yourself" > "explicit is better than implicit"

Conventions

web2py/
applications/
friends/
models/

controllers/main.py:

```
def index():
    return "hello world"
```

views/

http://.../**friends/main/index**

similar to Rails
but multi-app



Conventions

web2py/
applications/
friends/
models/

controllers/main.py:

```
def index():
    message = "hello world"
    return dict(message=message)
```

views/main/index.html:

```
{{extend 'layout.html'}}
<h1>Example</h1>
{{=message}}
```

http://.../**friends/main/index**

default layout.html is
mobile friendly, based
on flexible grid
(skeleton.css)

Conventions

web2py/

applications/

friends/

models/friends.py:

```
db.define_table('friend', Field('name'))
```

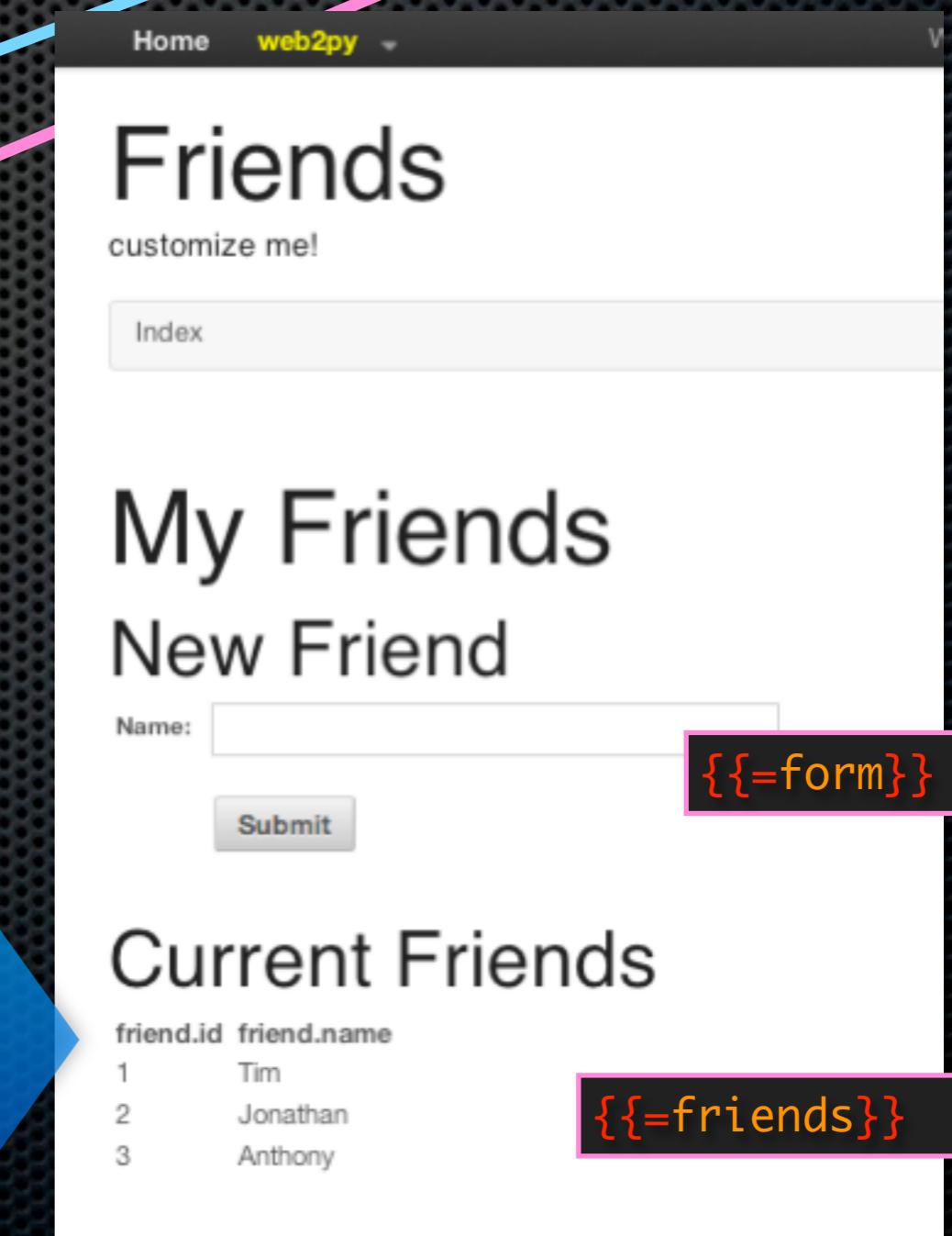
controllers/main.py:

```
def index():
    form = SQLFORM(db.friend).process()
    friends = db(db.friend).select()
    return locals()
```

views/main/index.html:

```
{{extend 'layout.html'}}
<h1>{{=T('My Friends')}}</h1>
<h2>New Friend</h2>
{{=form}}
<h2>Current Friends</h2>
{{=friends}}
```

http://.../friends/main/index



Controllers

Django (view in MTV)

```
def get_friend(request):
    friend_id = request.GET['id']
    friend = Friend.objects.get(pk=friend_id)
    output = friend.name
    return HttpResponseRedirect(output)
```



web2py (controller in MVC)

```
def get_friend():
    friend_id = request.get_vars['id']
    friend = Friend(friend_id)
    output = friend.name
    return dict(output=output)
```

Routes

Django (url.py)

```
urlpatterns = patterns('',
    (r'^friends/$', 'friends.views.index'),
    (r'^friends/(\d{4})/$', 'friends.views.read'),
)
```



web2py (routes.py)

```
routes_in = [
    (r'friends/', '/friends/default/index'),
    (r'friends/(\d{4})', '/friends/default/get_friend?id=\1') always optional
    (r'friends/$id', /friends/default/get_friend?id=\1'),
    (r'127.0.0.*:http://domain.com friends/(\d{4})', '/otherapp')
]
```

```
routes_out = [...]
```

```
routes_onerror = [
    (r'friends/400', r'/friends/default/login'),
    (r'/*/*', r'/friends/static/fail.html')]
```

reverse routes

routes on error

Models (DAL not ORM)

Django

```
class Friend(models.Model):
    name      = models.CharField(max_length=255, null=False)
    email     = models.EmailField()
    info      = models.TextField()
    image     = models.ImageField()
    birthdate= models.DateField()
```



django-admin.py syncdb

web2py

```
Friend = db.define_table('friend',
    Field('name', length=255, notnull=True),
    Field('email', requires=IS_EMAIL()),
    Field('info', 'text'),
    Field('image', 'upload', requires=IS_IMAGE()),
    Field('birthdate', 'date'))
```

automatic migrations
CREATE / ALTER tables

Meta Models (weird stuff)

web2py

```
db.define_table('meta_thing', Field('names', 'list:string'))  
  
meta_thing = db.meta_thing(session.id)  
  
Thing = db.define_table('thing_type_%i' % meta_thing.id,  
    *[Field(field.name) for field in meta_thing.fields])
```

Database queries

Django

```
q = Friend.objects.filter(name__startswith="T")
q = q.filter(birthdate__lte=datetime.now())
q = q.exclude(info__icontains="python")
print q
```

web2py

```
q = Friend.name.startswith("T")
q = q & (Friend.birthdate<datetime.now())
q = q & (!Friend.info.contains("python"))
print db(q).select()
```

overloaded operators

appadmin (database admin)

Django admin

The screenshot shows the Django admin interface. The top bar includes the title 'Site administration | Django site admin', the URL 'http://127.0.0.1:8000/admin/', and a Google search bar. The main header says 'Django administration' and 'Welcome, Jacob. Documentation / Change password / Log out'. Below this, the 'Site administration' heading is followed by a list of models: 'Auth' (Groups, Users), 'Sites' (Sites), and 'Ch6' (Authors, Books, Publishers). Each model entry has 'Add' and 'Change' buttons. To the right, a sidebar titled 'Recent Actions' shows 'None available'. A cartoon character in the bottom right corner holds a yellow heart-shaped balloon with the word 'Thanks!' written on it.

every app has its own
same look and feel as the app
designed for administrators not users
crud components can be embedded in apps
runs on GAE

web2py appadmin

design db state cache

Welcome Massimo Logout | Pr

Friends

customize me!

[Index](#)

Available databases and tables

[db.auth_user](#)
[insert new auth_user]

[db.auth_group](#)
[insert new auth_group]

[db.auth_membership](#)
[insert new membership]

database [db](#) table friend record id [2](#)

Edit current record

Id:

Name:

Check to delete:



CRUD, Grid, SmartGrid

web2py

```
def manage_friends():
    grid = SQLFORM.smartgrid(db.friend)
    return locals()
```

Manage Friends

Friends

Query Search Clear

Name New And Or

Id	Name	View	Edit	Delete
1	Tim	<input type="button" value="View"/>	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
2	Jonathan	<input type="button" value="View"/>	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
3	Anthony	<input type="button" value="View"/>	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>

Manage Friends

Friends

Back View

Id: 2
Name:

Check to delete:

Submit

breadcrumbs

query builder

pagination

reference links

Role Based Access Control

web2py

```
@auth.requires_login()  
@auth.requires_membership(role='friend manager')  
@auth.requires_permission('delete', 'friend')  
def manage_friends():  
    ...
```

The image displays five distinct web forms arranged in a grid-like layout:

- Register:** A form for creating a new account. It includes fields for First name, Last name, E-mail, and Password.
- Profile:** A form for managing user profile information. It includes fields for First name, Last name, E-mail, and Password. The E-mail field contains the value "mdipierro@cs.depaul.edu".
- Change password:** A form for changing a user's password. It includes fields for Old password, New password, and Verify Password, along with a "Change password" button.
- Login:** A form for logging in. It includes fields for E-mail and Password.
- Request reset password:** A form for requesting a password reset. It includes a single E-mail field and a "Request reset password" button.

Federated Authentication

(with built-in CAS consumer and provider in every app)

Provider app



any app can delegate authentication to another local or remote web2py app (CAS 2.0)



Consumer apps

```
auth = Auth(db, cas_provider='http://.../friends/default/user/cas')
```

Record Versioning (all tables)

web2py

```
auth.enable_record_versioning(db)
```

Will record all changes to all records

Templates

Mako

```
<%inherit file="base.html"/>

<%def name="show(friend)">
  <tr>
    <td>${friend.name}</td>
    <td>${friend.email}</td>
  </tr>
</%def>

<table>
  % for friend in friends:
    ${show(friend)}  
  % endfor
</table>
```



web2py

```
<{{extend "base.html"}}

{{ def show(friend): }}
```

return closes def

```
<tr>
  <td>{{=friend.name}}</td>
  <td>{{=friend.email}}</td>
</tr>
{{ return }}
```

pass closes for

```
<table>
  {{ for friend in friends: }}
```

{{ show(friend) }}

{{ pass }}

```
</table>
```

everything escaped by default except XML('...'), sanitize=True)

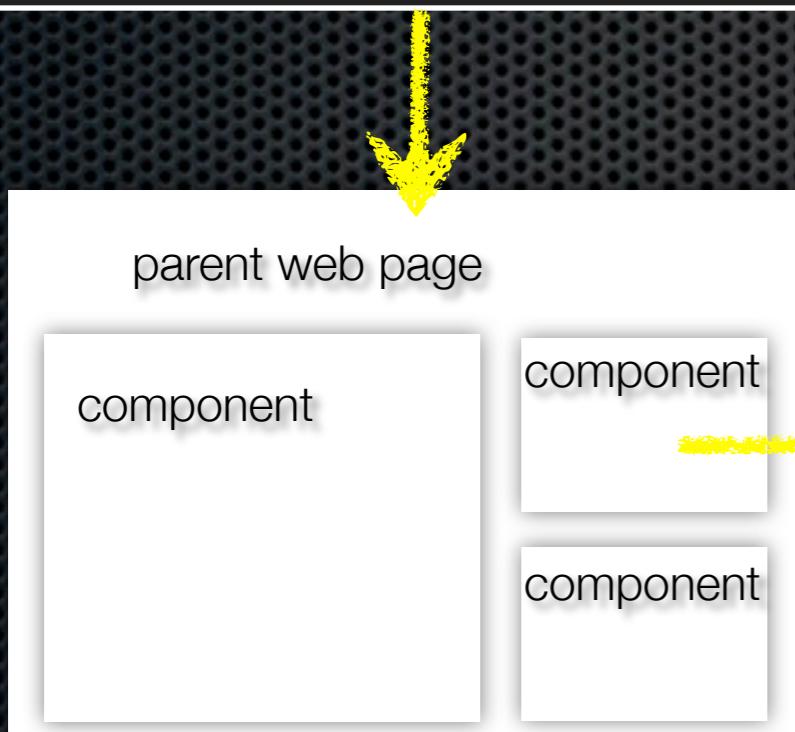
Beyond MVC

(ajax components with signed URL)



web2py (parent template)

```
<div>
{{=LOAD('plugin','component1',user_signature=True,ajax=True)}}
{{=LOAD('plugin','component2',user_signature=True,ajax=True)}}
{{=LOAD('plugin','component3',user_signature=True,ajax=True)}}
</div>
```



```
@auth.requires_signature()
def component1():
    form = SQLFORM(db.friend)
    if form.process().accepted:
        response.flash = 'done!'
    return locals()
```

Modules and thread locals

Flask (proxies to objects that are local to a specific context)

```
from flask import request  
  
with app.request_context(environ):  
    assert request.method == 'POST'
```



web2py (thread-locals)

```
from gluon import current  
  
assert current.request.env.http_method == 'POST'
```

thread local

request obj

wsgi environ



No Library Conflicts



web2py (each app can ship with its own version of libraries)

```
# app 1
import mylib      # from applications/app1/modules/
# app 2
import mylib      # from applications/app2/modules/
```

<https://github.com/mitsuhiko/multiversion>

```
import multiversion
multiversion.require_version('mylib', '1.0')
import mylib
```

Web Services

```
@service.json  
@service.xml  
@service.jsonrpc  
@service.xmlrpc  
@service.soap  
@service.amfrpc3('domain')  
def all(key):  
    return db(db.friend.name.contains(key)).select()
```

```
http://.../friends/default/call/json/add?key=tim  
http://.../friends/default/call/xml/add?key=tim  
...
```

```
from xmlrpclib import ServerProxy  
server = ServerProxy('http://.../friends/default/call/xmlrpc')  
print server.key('tim')
```

RESTful Web Services

```
@request.restful()
def myAPI():
    def POST(**vars):
        return db.friend.insert(**vars)
    def GET(*args,**vars):
        patterns = [
            "/friends[friend]",
            "/{friend.name.startswith}",
            "/{friend.name}/:field",
        ]
        return db.parse_as_rest(patterns,args,vars).response
    def DELETE(id):
        return db(db.friend.id==id).delete()
    def PUT(id,**vars):
        return db(db.friend.id==id).update(**vars)
    return locals()
```



<http://.../friends/default/myAPI/John/email>

Scheduler



web2py (in a model file for example)

```
from gluon.scheduler import Scheduler

def email_friend(friend_id,subject,message):
    email = db.friend(friend_id).email
    mailer.send(to=email, subject=subject, message=message)
    return 'done!'

scheduler = Scheduler(db,dict(mytask=email_friend))
```

Start workers, local or remote

```
web2py.py -K friends &
web2py.py -K friends &
web2py.py -K friends &
```

```
db.scheduler_task.insert(task='mytask',args=[2,'hi','...'])
```

Manage tasks from appadmin interface

Multi-tenancy



web2py

```
db._common_fields.append(  
    Field('request_tenant', default=request.env.host_name))
```

`http://domain1/friends`

`http://domain2/friends`

`http://domain3/friends`

share database but see different data
records filtered by domain (or other condition)

Let's build something

- URL shortening service
- Bookmarking service with tagging
- Click counter
- Url rating (WOT service)

models/myapp.py

```
Link = db.define_table('link',
    Field('url',unique=True),
    Field('visits','integer',default=0),
    Field('screenshot','upload',writable=False),
    format = '%(url)s')

Bookmark = db.define_table('bookmark',
    Field('link','reference link',writable=False),
    Field('category',
        requires=IS_IN_SET(['work','personal'])),
    Field('tags','list:string'),
    auth.signature)
```

controllers/default.py

```
def index():
    return dict()
```

models/myapp.py

```
def toCode(id):
    s,c = 'GKys67LJPDAFvcEp9rkwRd43fCjbxSXzMTQ28hgeUWuNYmqZt5VanBH', ''
    while id: c,id = c+s[id % 55], id//55
    return c

def toInt(code):
    s,id = 'GKys67LJPDAFvcEp9rkwRd43fCjbxSXzMTQ28hgeUWuNYmqZt5VanBH', 0
    for i in range(len(code)): id += s.find(code[i])*55**i
    return id

def shorten(id, row):
    s = URL('visit', args=toCode(id), scheme=True)
    return A(s,_href=s)

Bookmark.link.represent = shorten
Bookmark.id.readable = False
Bookmark.is_active.readable = False
Bookmark.is_active.writable = False
```

models/myapp_utils.py

```
from urllib import urlopen

def thumbalizr(url):
    """ free service to download web page thumbnail """
    return Bookmark.screenshot.store(
        urlopen('http://api1.thumbalizr.com/?url=%s&width=250' % url),
        filename = 'shot.png')

def wotrate(url):
    """ Web Of Trust service for web site rating """
    from gluon.tools import fetch
    from gluon.contrib.simplejson import loads
    wot = 'http://api.mywot.com/0.4/public_link_json?hosts=%s/'
    url = url.split('/')[2]
    return loads(urlopen(wot % url).read())[url]
```

models/menu.py

```
response.menu = [
    (TC('Home'), False, URL('default','index')),
    (TC('My Bookmarks'), False, URL('default','bookmarks')),
]
```

controllers/default.py

<http://127.0.0.1:8000/myapp/default/bookmark?url=http://www.google.com>

```
@auth.requires_login()
def bookmark():
    """ allows users to bookmark a page an get short url """
    url = request.vars.url
    link = Link(url=url) or Link.insert(url=url)
    link.update_record(screenshot=thumbalizr(url))
    bid = Bookmark(link=link) or Bookmark.insert(link=link)
    rating = cache.ram(link.url,lambda:wotrate(link.url),3600)
    form = SQLFORM(Bookmark,bid).process(next='bookmarks')
    return locals()

def visit():
    """ tracks visitors """
    link = Link(toInt(request.args(0)))
    link.update_record(visits=link.visits+1)
    redirect(link.url)
```

controllers/default.py

```
@auth.requires_login()
def bookmarks():
    """ allow users to manage their bookmarks """
    query = (Bookmark.created_by==auth.user.id)&\
            (Bookmark.link==Link.id)
    grid = SQLFORM.grid(query, create=False)
    return locals()

# stuff required for authentication
def user(): return dict(form=auth())
def download(): return response.download(request, db)
```

routes.py

localhost/K2 > localhost/myapp/default/visit/K2 > www.google.com

```
routes_in = [
    ('/(?P<code>\w+)', '/myapp/default/visit/\g<code>'),
]

routes_out = [
    ('/myapp/default/visit/$code', '/$code'),
]

routes_onerror=[

    ('myapp/*', 'myapp/default/error'),
]
```

views/default/bookmarks.py

```
{{extend 'layout.html'}}  
<h3>My Bookmarks</h3>  
{%grid%}
```

The screenshot shows a web browser window with the URL `127.0.0.1:8000/myapp/default/bookmarks`. The page title is "myapp". The main content area is titled "My Bookmarks". At the top, there is a search bar with "Query" and "Search" buttons, and an "Export" button. Below the search bar, a message says "2 records found". A table lists two bookmarks:

Link	Category	Tags	Id	Url	Visits	Screenshot	View	Edit	Delete
http://127.0.0.1:8000/K	None		1	http://www.google...	0		View	Edit	Delete
http://127.0.0.1:8000/y	None		2	https://www.googl...	0		View	Edit	Delete

views/default/bookmark.py

```
{{{extend 'layout.html'}}}

<h3>Bookmark for {{=url}}</h3>

{{{def show(rating,k):}}
    <strong>{{=rating.get(k,[0])[0]}}%</strong>{{return}}
<div>
Trustworthiness: {{show(rating,'1')}}
Reliability: {{show(rating,'2')}}
Privacy: {{show(rating,'3')}}
Child safety: {{show(rating,'4')}}
</div>

{{=form}}
```

views/default/bookmark.py

The screenshot shows a web browser window with the URL `127.0.0.1:8000/myapp/default/bookmark?url=https://www.google.com/` in the address bar. The page title is "myapp". The main content area displays a bookmark for `https://www.google.com/`. The bookmark details include:

- Link: <http://127.0.0.1:8000/y>
- Category: work
- Tags: (empty input field)

Below the form, there is a small thumbnail image of the Google homepage.

views/default/index.html

```
{{right_sidebar_enabled=True}}
{{extend 'layout.html'}}
```

Welcome to our bookmarking & url shortening service

```
{{block right_sidebar}}
Drag link to favorites:
<a style="padding:5px;color:white;background-color:black;" 
title="Bookmark"
href="javascript:window.location='{{=URL('bookmark',
scheme=True)}}?url='+window.location">Bookmark!</a>
{{end}}
```



Welcome to our URL bookmarking, shortening, and rating service

Drag link to favorites:

Bookmark!

models/services.py

<http://127.0.0.1:8000/myapp/services/call/xml/linksby?key=google>
<http://127.0.0.1:8000/myapp/services/api/google.json>

```
@service.json
@Service.xml
@Service.jsonrpc
@Service.xmlrpc
@Service.amfrpc3('domain')
@Service.soap()
def linksby(key=''):
    return db(Link.id==Bookmark.link) \
        (Bookmark.tags.contains(key)).select(Link.ALL,distinct=True)

@request.restful()
def api():
    def POST(**vars):
        raise HTTP(501)
    def GET(key=''):
        return dict(result=linksby(key).as_list())
    return locals()

def call(): return service()
```

models/mytasks.py

```
from gluon.scheduler import Scheduler

def emailme():
    me = 'mdipierro@cs.depaul.edu'
    message = '%s users' % db(db.auth_user).count()
    mailer.send(to=me, subject='stats', message=message)
    return 'done!'

scheduler = Scheduler(db,dict(emailme=emailme))
if db(db.scheduler_task).isempty():
    db.scheduler_task.insert(
        application_name=request.application,
        task_name='email me user count',
        function_name='emailme',args=[],vars={},
        period=3600*24,repeats=0)
```

```
web2py.py -K friends &
web2py.py -K friends &
web2py.py -K friends &
```

Give it a try!

