



# web2py

ideas we borrowed, ideas we had

Massimo Di Pierro  
Associate Professor  
School of Computing



DEPAUL UNIVERSITY

# Contributors

Daniel.Lin  
 Hans.Murx  
 Jose.L..Redrejo.Rodriguez  
 Gilson.Filho  
 Jose.Vicente.de.Sousa  
 Denes.Lengyel  
 Arun.Robert.Valentak  
 Nicolas.Bruixer  
 Omi.Chiba  
 Bill.Ferrett  
 Falko.Krause  
 Marcel.Leuthi  
 Pierre.Thibault  
 Yannis.Aribaud  
 Pai  
 Michele.Comitini  
 Mateusz.Banach  
 Arun.K..Rajeevan  
 Attila.Csipa  
 Vidul.Nikolaev.Petrov  
 C.J.Lazell  
 Graham.Dumpleton  
 Angelo.Compagnucci  
 Alexey.Nezhdanov  
 Thadeus.Burgess  
 Phylo.Arkar.Lwin  
 Simone.Bizzotto  
 Stuart.Rackham  
 Niall.Sweeny  
 Patrick.Breitenbach  
 Fran.Boon  
 Jose.Jachuff  
 Mark.Larsen  
 Anders.Roos  
 Hans.C..v..Stockhausen  
 Chris.Steel  
 Zahariash  
 Ruijun.Luo  
 Martin.Mulone  
 Kenji.Hosoda  
 Timothy.Farrell  
 Christian.Foster.Howes  
 Sergey.Podlesnyi  
 Ian.Reinhart.Geisser  
 Anthony.Bastardi  
 Telman.Yusupov  
 David.Wagner  
 Jonathan.Benn  
 Josh.Goldfoot  
 Lucas.D'Avila  
 Alan.Etkin  
 Tim.Michelsen  
 Craig.Younkins  
 Michael.WillisBoris.Manojloovic  
 Vinicius.Assef  
 Olaf.Ferger  
 Marin.Pranjic  
 Kyle.Smith  
 Ryan.Seto  
 Eric.Vicenti  
 Chris.May  
 Scott.Roberts  
 Carlos.Galindo  
 Limodou  
 Yair.Eshel  
 Chris.Clark  
 Ben.Goosman  
 Branko.Vukelic  
 Jan.Beilicke  
 Jonathan.Lundell  
 Mariano.Reingart  
 Brian.Meredyk  
 Sriram.Durbha  
 Ondrej.Such  
 Markus.Gritsch  
 Francisco.Gama  
 Paolo.Caruccio  
 Keith.Yang  
 Farshed.Ashouri  
 Ross.Peoples  
 Younghyun.Jo  
 Marcel.Hellkamp  
 Hamdy.Abdel-Badeea  
 Bruno.Rochaes  
 Sharriff.Aina  
 Hans.Donner  
 Nathan.Freeze  
 Fred.Yanowski  
 Mark.Moore

# Resources

The screenshot shows the Web2py homepage. At the top, there's a banner for 'InfoWorld's 2012 Technology of the Year Award Winner'. Below it, there are three mobile device screenshots showing different web applications built with Web2py. A yellow circle highlights these three devices. To the right of the devices, a yellow box contains the word 'videos'. Further down the page, there are sections for 'WEB2PY™ WEB FRAMEWORK', 'BATTERIES INCLUDED', 'WEB-BASED IDE', and 'EXTENSIVE DOCS'.

This screenshot shows a grid of 60+ example applications built with Web2py. Each example has a thumbnail, a title, and download/browse links. Examples include 'FacebookClone', 'FacebookConnectExample', 'FileTreeBrowser', 'FindRideSharing', 'ImageGallery', 'GeoLocateByIp', 'NeedARide', 'GeoLiteCountry', 'HotelManagementExample', and 'HotelManagement'. A yellow box on the right contains the text '60+ example apps'.

web2py.com



web2py.com/appliances

The screenshot shows the S-cubism website for web2py-plugins. It features a large background image of clouds with network graph icons. The page includes social sharing buttons (Twitter, Facebook, Google+) and a 'Contact us' form. Below the main image, there's a section for 'Form Widgets' with examples like 'Horizontal Radio Widget', 'Multiple Select Widget', and 'Suggest Widget'. A blue box at the bottom contains the URL 'dev.s-cubism.com/web2py\_plugins'.

3

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

# Ideas we borrowed

- Model View Controller on WSGI (like everybody else)
- w2p files (like Java's Web application ARchives)
- Optional Routing (like Rails)
- Routing mechanism (like Django)
- Pure Python Template Language (like Mako)
- Helpers (like Rails) but easier to remember: SPAN, A, ...
- `request['xxx'] == request.xxx` (like web.py)
- web based database interface (like Django admin)

# Ideas we had ...

- A tool for both technical and non-technical people
- Always backward compatible (since 2007, 2.5, 2.6, 2.7, pypy, jython)
- One click deploy (Windows and Mac binaries, USB drive)
- No configuration, no dependencies, and secure by default
- Everything has default (DRY)
- Multi project / multi db / share nothing by default
- Web based IDE (development, editor, deployment, management, translations, testing, debugger, version control) shell optional
- Automatic DB migrations (CREATE and ALTER table)
- Plugins / Components / Ajax with Digitally Signed URLs

# ... Ideas we had

- Role Based Access Control with pluggable authentication modules (openid, dlap, cas, oauth 1/2, pam, janrain, google, dropbox)
- Every app is a Central Authentication Service provider (and optionally consumer)
- Built-in portable cron and master/workers task scheduler
- multi tenant apps with common fields & common filters
- record versioning (without breaking refs)
- embeddable crud & grid controls
- Package lots of 3rd party modules: pyfpdf, database drivers, credit card payment APIs, ....

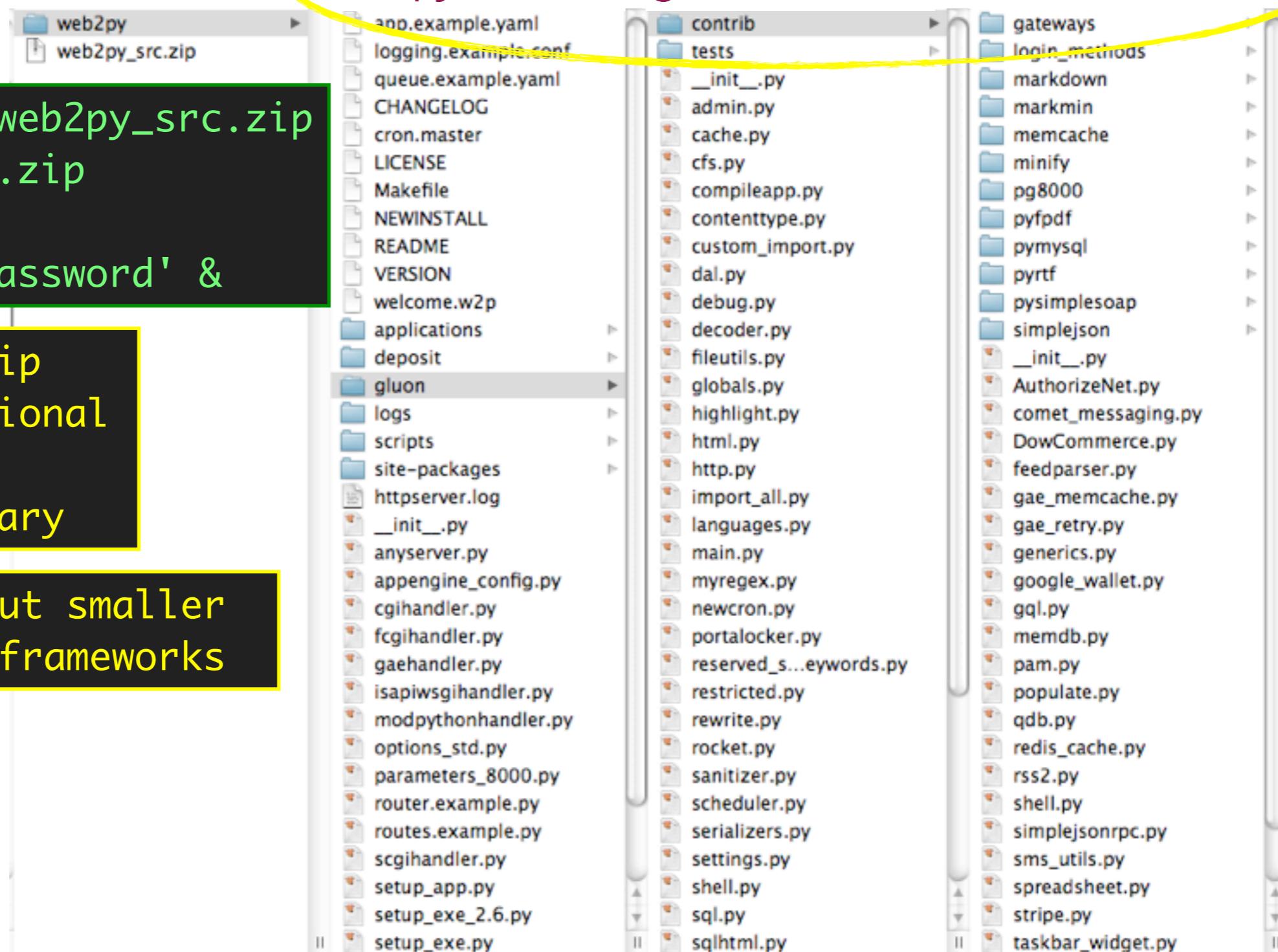


web2py/ 0.1MB gluon/ 1.3MB contrib/ 2.0MB

```
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

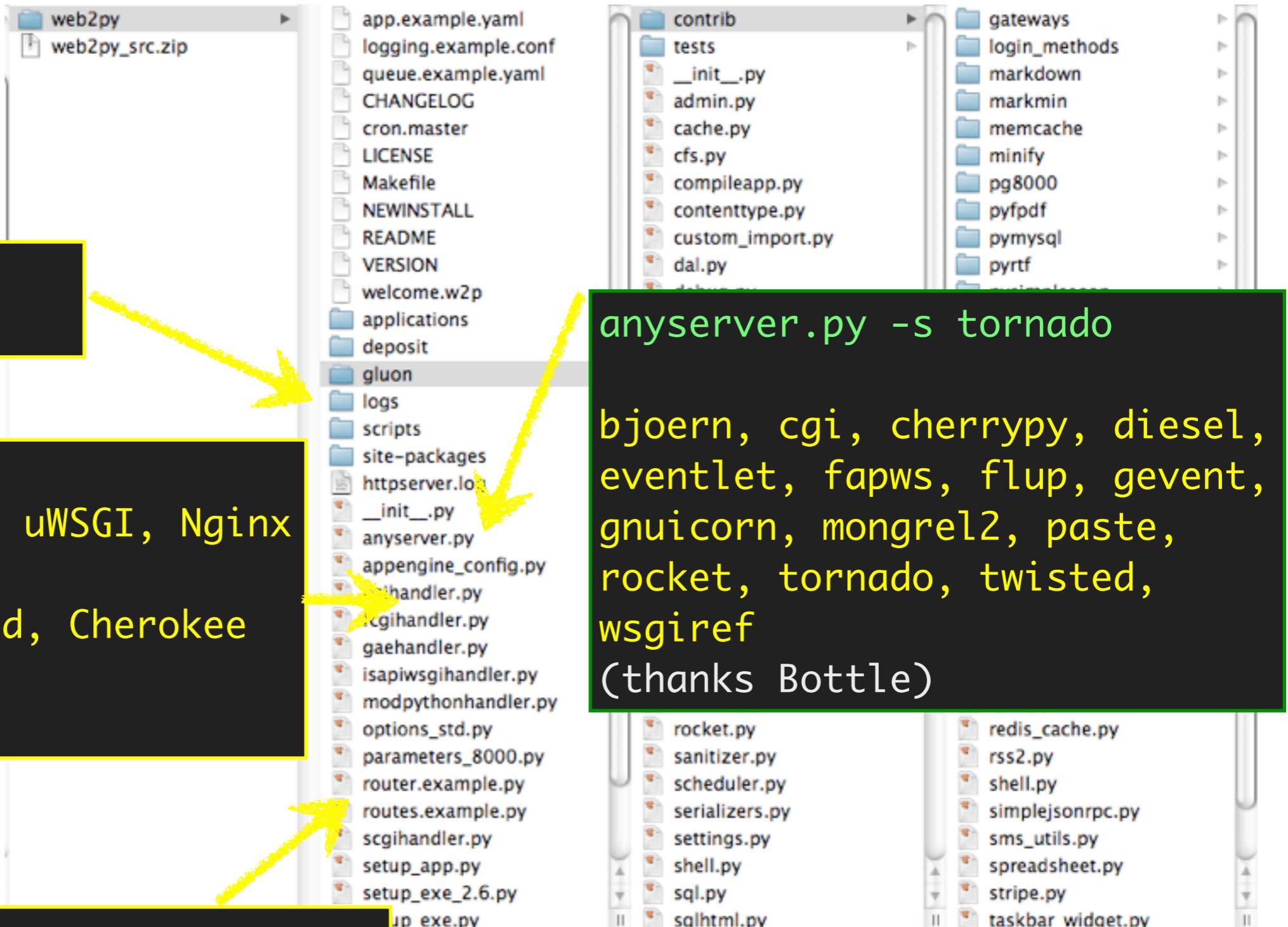
all-inclusive, but smaller  
than most micro-frameworks



`*.log:`  
log files

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

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



`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

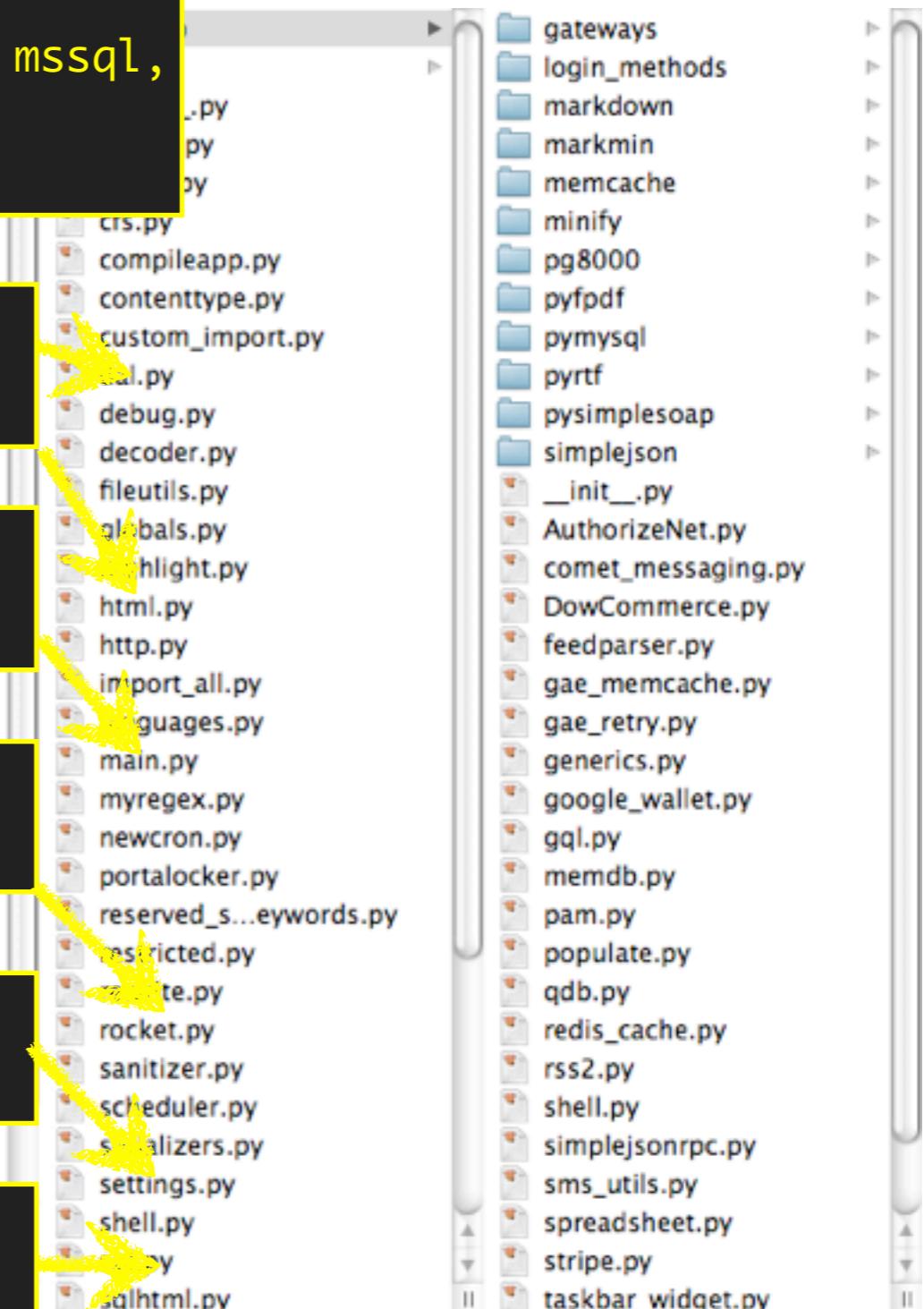
`html.py` & `sqlhtml.py`:  
helpers, html parser, form generation

`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:

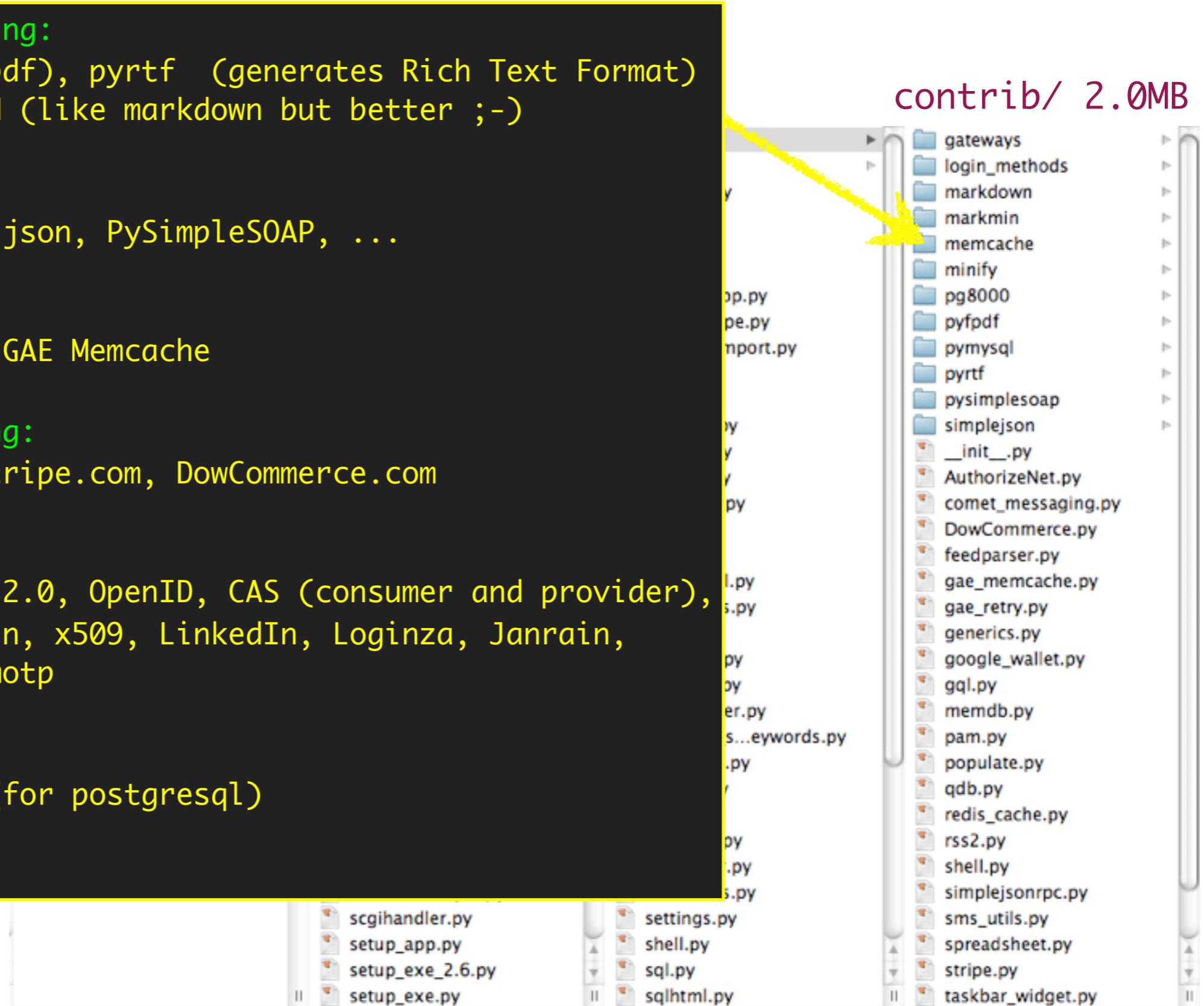
Oauth 1.0, OAuth 2.0, OpenID, CAS (consumer and provider),  
PAM, LDAP, Janrain, x509, LinkedIn, Loginza, Janrain,  
gmail, Dropbox, motp

Drivers:

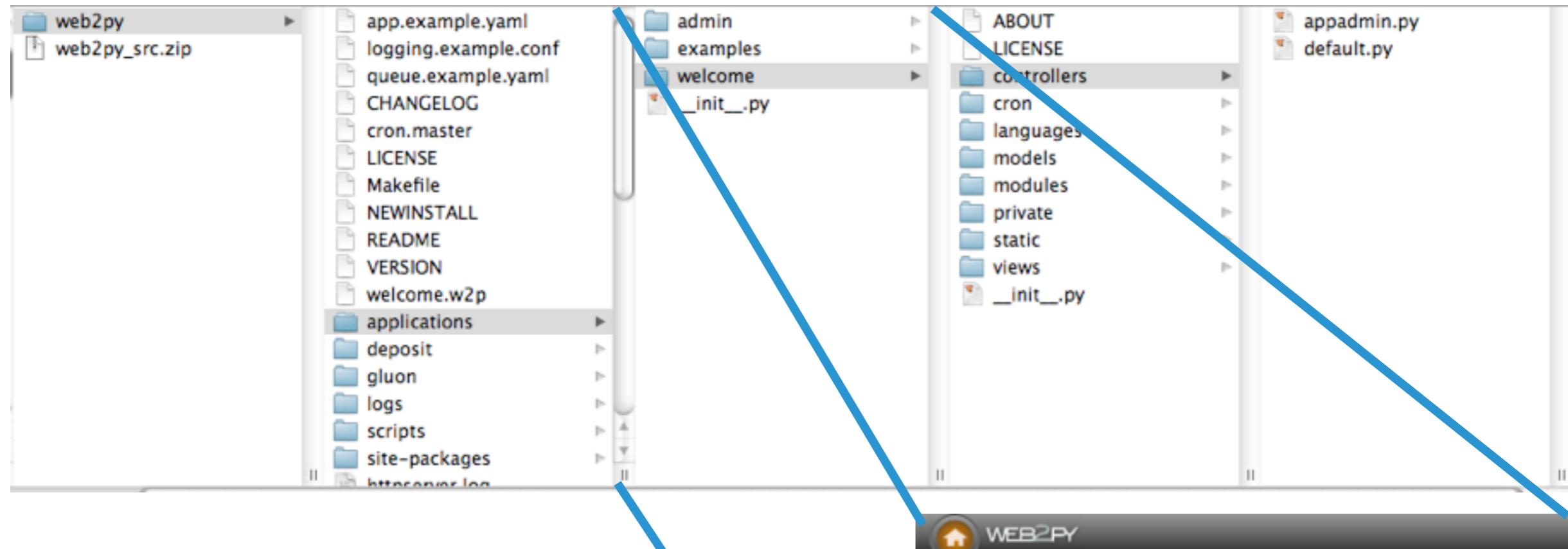
pyMySQL, pg8000 (for postgresql)

other stuff...

contrib/ 2.0MB



# 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 administration interface at `127.0.0.1:8000/admin/default/site`. On the left, there's a sidebar titled "INSTALLED APPLICATIONS" listing "admin (currently running)", "examples", "friends", and "welcome", each with edit, about, errors, clean, pack all, compile, uninstall, and disable buttons. A yellow box highlights the "manage/edit" link in the bottom-left corner. On the right, there's a "Change admin password" button and a status message: "Version 1.99.7 (2012-03-07 15:25:49) dev" followed by "Unable to check for upgrades". Below this, there are sections for "Running on Rocket 1.2.4", "Reload routes", "New application wizard" (with a "Start wizard" button), "New simple application" (with an "Application name:" input field and "Create" button), and "Upload and install packed application" (with fields for application name, upload package, get from URL, and an "Install" button). A blue arrow points from the "remote install and deploy" text below to the "Upload and install packed application" section. To the right of the "wizard" section is a "create" button, and below it is a cloud icon with a propeller, representing Google App Engine deployment.

site  
127.0.0.1:8000/admin/default/site  
WEB2PY Site Logout Debug Help

INSTALLED APPLICATIONS

- admin (currently running)  
Errors Clean Pack all Compile
- examples  
Edit About Errors Clean Pack all Compile Uninstall Disable
- friends  
Edit About Errors Clean Pack all Compile Uninstall Disable
- welcome  
Edit About Errors Clean Pack all Compile Uninstall Disable

Change admin password

Version 1.99.7 (2012-03-07 15:25:49) dev  
Unable to check for upgrades

Running on Rocket 1.2.4

Reload routes

New application wizard  
Start wizard (requires internet access)

New simple application  
Application name: Create

Upload and install packed application  
Application name:  
Upload a package: Choose File No file chosen  
Get from URL:  
Overwrite installed app  
Install

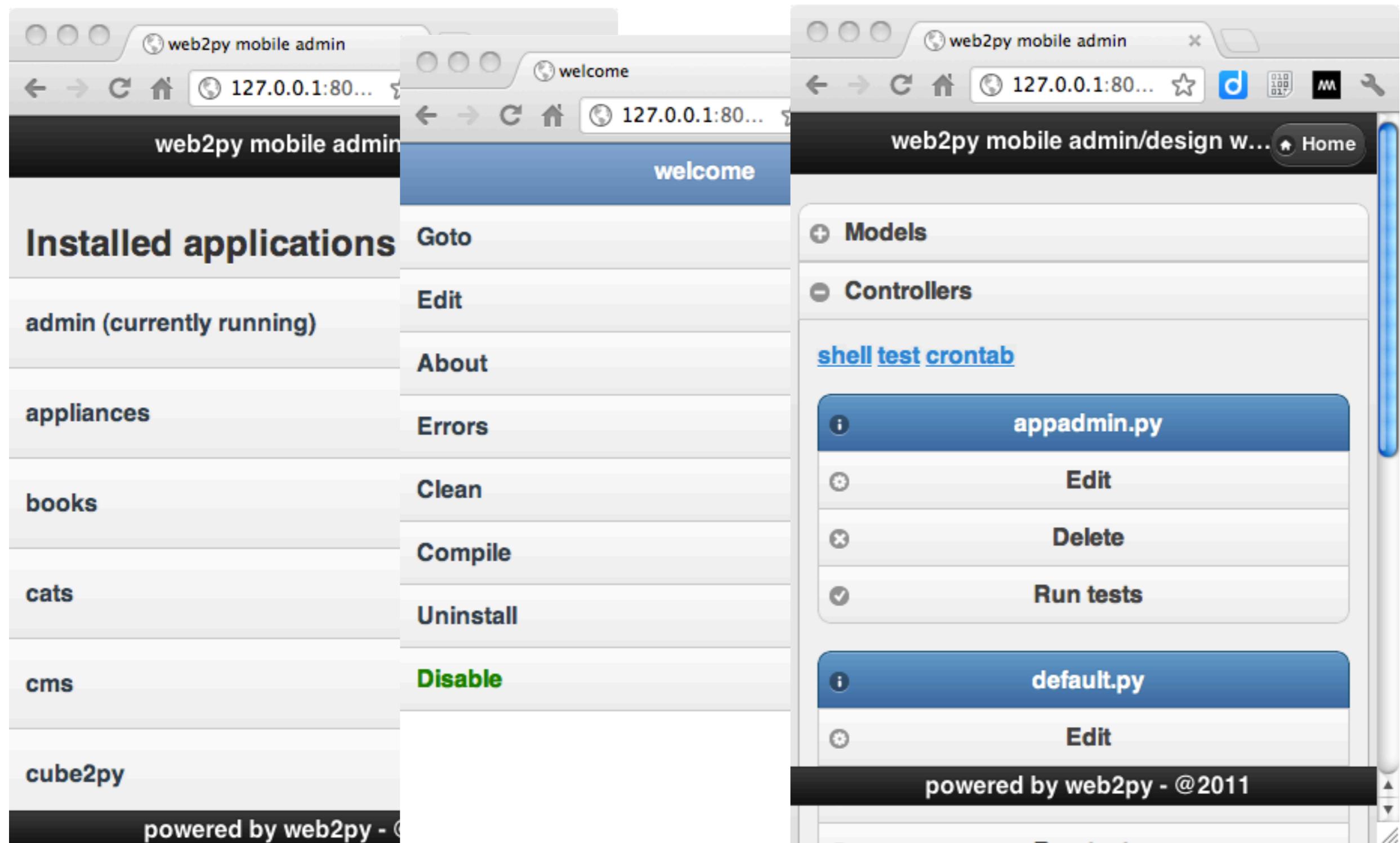
remote install and deploy

wizard

create

12

# Mobile IDE



# Web Translation

The screenshot shows the Web2py administrative interface for managing language files. At the top, there's a navigation bar with a home icon and the text "WEB2PY". Below it, two main sections are listed:

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

Under these sections, several language entries are displayed in a table-like format:

"update" is an optional expression like "field1='newvalue'". You cannot update or delete the results of a JOIN query.	<a href="#">delete</a>
"update" è un'espressione opzionale come "campo1='nuovo valore'". Non si può fare "update" o "delete" dei risultati di un JOIN	<a href="#">delete</a>
%s rows deleted	<a href="#">delete</a>
%s righe ("record") cancellate	<a href="#">delete</a>
%s rows updated	<a href="#">delete</a>
%s righe ("record") modificate	<a href="#">delete</a>
%Y-%m-%d	<a href="#">delete</a>
%d/%m/%Y	<a href="#">delete</a>
%Y-%m-%d %H:%M:%S	<a href="#">delete</a>
%d/%m/%Y %H:%M:%S	<a href="#">delete</a>
Administrative interface	<a href="#">delete</a>
Interfaccia amministrativa	<a href="#">delete</a>

# Tickets

The screenshot shows the Web2py error ticket interface. At the top, there's a navigation bar with a home icon and the text "WEB2PY". Below it, a yellow box highlights the text "no distinction debug vs production". The main area has a title "ERROR TICKET FOR 'A1'" with a gear icon. To the left, another yellow box highlights "ERROR LOGS FOR 'A1'". Below this, there are buttons for "check all", "unchecked all", and "delete all checked". A green circular icon with a dot indicates "Click row to expand traceback". To the right, a yellow box highlights "web debugger". In the center, a yellow box highlights "groups common errors". Below the logs, there's a table with columns "Delete", "Count", "File", and "Error". Two entries are listed: one for "default.py" with a "ZeroDivisionError" and another for "default.py" with a "NameError". Each entry has a "+ details" link. To the right, a yellow box highlights "traceback". The traceback text is as follows:

```
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

The image shows a screenshot of the Web2py application structure and a terminal window.

**File Explorer:** Shows the directory structure of the Web2py source code. It includes files like app.example.yaml, logging.example.conf, queue.example.yaml, CHANGELOG, cron.master, LICENSE, Makefile, NEWINSTALL, README, VERSION, welcome.w2p, and various application folders (applications, deposit, gluon, logs, scripts, site-packages). A blue arrow points from the 'welcome' folder in the applications structure towards the terminal window.

**Web2py Application Editor:** Shows the 'EDIT APPLICATION "WELCOME"' interface. It has tabs for Models, Controllers, Views, Languages, Static files, Modules, and Plugins. The 'Controllers' tab is selected, showing subfolders like ABOUT, LICENSE, controllers, cron, languages, models, modules, private, static, views, and \_\_init\_\_.py.

**Terminal Window:** A green-highlighted terminal window displays the following commands:

```
cd applications
mkdir friends
cp -r welcome/* friends
emacs friends/models/db.py
emacs friends/controllers/default.py
emacs friends/views/default/index.html
```

A yellow arrow points from the terminal window towards the 'Controllers' tab in the application editor.

# 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 title bar says "Bookmark". The top navigation bar includes "Home", "My Bookmarks", and "Welcome Massimo Logout | Profile". The main content area has a heading "myapp" and a sub-section "Bookmarks". A "Share" button is visible. Below is a table titled "My Bookmarks" with the following data:

Link	Category	Tags	Id	Url	Visits	Screenshot	View	Edit	Delete
<a href="http://127.0.0.1:8000/K">http://127.0.0.1:8000/K</a>	None		1	http://www.google...	0	<a href="#">File</a>	<a href="#">View</a>	<a href="#">Edit</a>	<a href="#">Delete</a>
<a href="http://127.0.0.1:8000/y">http://127.0.0.1:8000/y</a>	None		2	https://www.googl...	0	<a href="#">File</a>	<a href="#">View</a>	<a href="#">Edit</a>	<a href="#">Delete</a>

# 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 header includes "Bookmark!", "Home", "My Bookmarks", "Welcome Massimo Logout | Profile", and a search icon. The main content area has a "Bookmark" button and a "Share" button. Below this, it displays a bookmark for `https://www.google.com/` with the following details:

Bookmark for `https://www.google.com/`

Trustworthiness: 95% Reliability: 93% Privacy: 0% Child safety: 93%

Link: <http://127.0.0.1:8000/y>

Category: work

Tags:

Submit

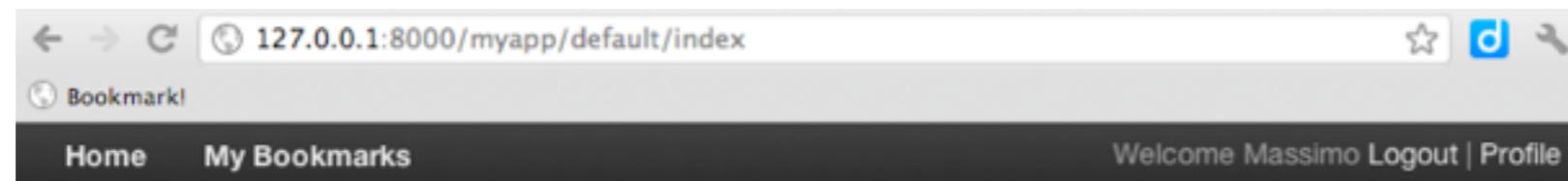
To the right of the form, there is a small preview image of the Google homepage.

# 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}}
```



# myapp

Index

+ Share

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

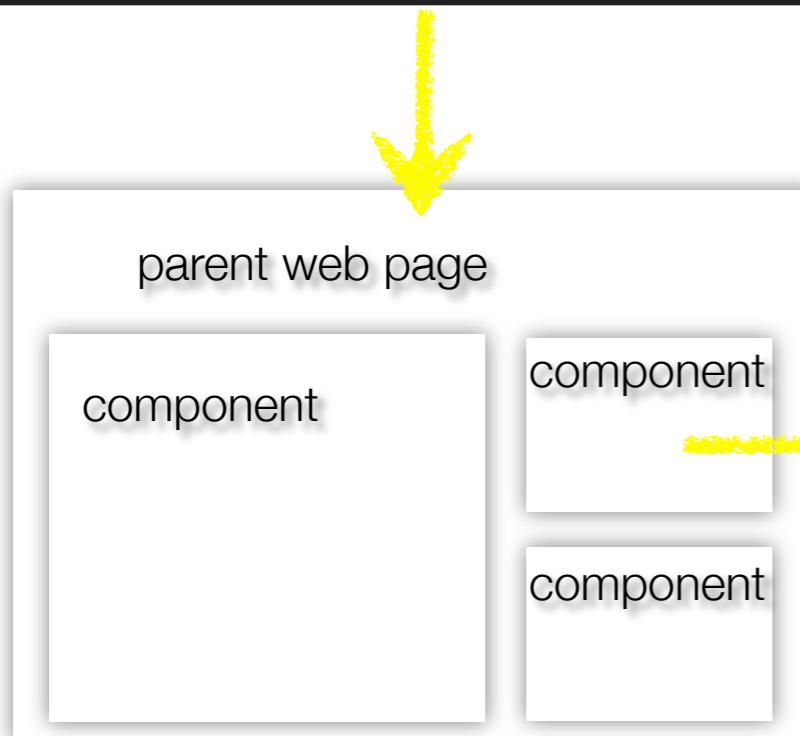
def emailme():
    me = 'mdipierro@cs.depaul.edu'
    message = '%s users' % db(db.auth_user).count()
    mailer.send(to=me, subject='stats', message=message)
    return 'done!'

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 &
```

# Components

```
<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()
```

# Thank you!

