

# CDM.DEPAUL.EDU

- ▶ 90 full time faculty
- ▶ 300 courses/quarter
- ▶ 1858 graduate students in 20 programs (CS, IS, ...)
- ▶ 1763 undergraduate students in 16 programs
- ▶ DHS and NSA center of excellence

## MASSIMO DI PIERRO

- ▶ PhD in Physics (Lattice Quantum Chromodynamics)
- ▶ “Director” MS in Computation Finance
- ▶ Interests: Numerical Algorithms, Web Development



# WEB DEVELOPMENT WITH WEB2PY

Massimo Di Pierro

School of Computing and Digital Media



Chicago, IL

my job is to make web development **easy**

my job is to make web development **easy**

**easy** != dumbed down

**easy** != visual programming

**easy** => more **intuitive** / less error prone

**easy** => more **expressive**

**easy** => more **powerful** syntax

**easy** is not just for kids

**easy** means **experienced developers** can concentrate on what is important: algorithms

**easy** means less development and maintenance costs

Disclaimer: I do not claim any success. I am just trying....

# WE2PY: BATTERIES INCLUDED

# web server

ssl enabled

# DAL + database

## auto-migrations

## SQLite

# web IDE

design, deploy, manage



html, xml, json, rss, ics, pdf, rtf,  
xmlrpc, jsonrpc, soap,  
ldap, pam, janrain, dropbox, google,  
CAS, OpenID, oauth 1&2, x509  
marmin, markdown,  
google wallet, authorize.net, stripe.com  
memcache, redis  
twitter bootstrap



.zip

# No installation. No configuration. Just Unzip and Click!

# WEB2PY CONTRIBUTORS

2011



2012



# Ideas we borrowed

- Model View Controller on WSGI (like everybody else)
- w2p files (like Java's Web application ARchives)
- Routing mechanism (like Django, but optional like Rails)
- Pure Python Template Language (like Mako)
- Helpers (like Rails) but easier: DIV, SPAN, A, ...
- web based database interface (like Django admin)

# Ideas we had ...

- 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 and multi db but 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 pam, janrain, google, dropbox)
- Every app is a Central Authentication Service consumer and provider.
- Built-in portable cron and master/workers task scheduler
- Full Auditing for all tables
- Ajax embeddable crud & grid controls

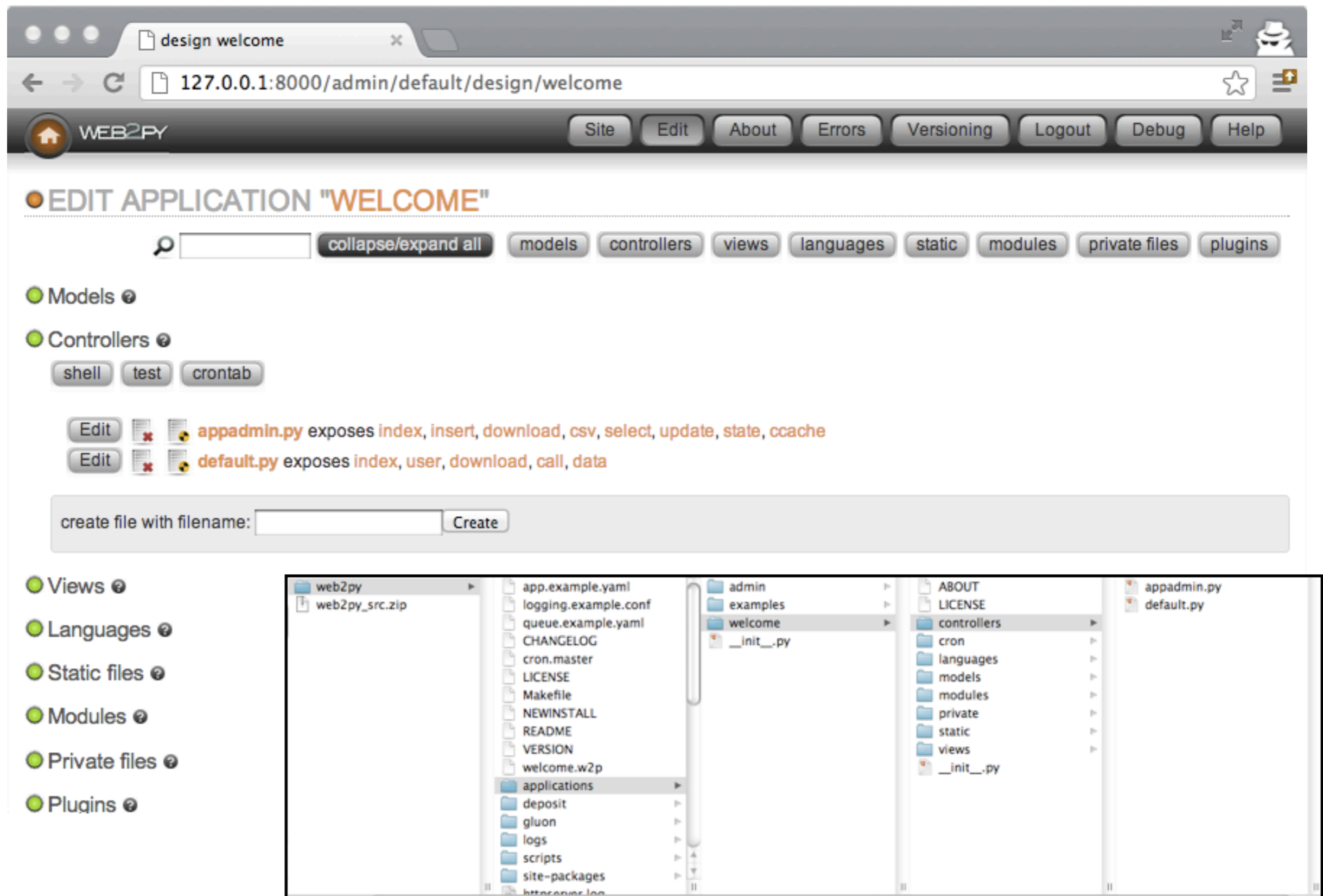


# Web based IDE “admin” with hot plug and play of multiple apps

The screenshot displays the Web2Py web-based IDE admin interface in a browser window. The address bar shows the URL `127.0.0.1:8000/admin/default/site`. The interface includes a navigation bar with buttons for **Site**, **Logout**, **Debug**, and **Help**. The main content area is titled **INSTALLED APPLICATIONS** and lists three applications: **admin (currently running)**, **examples**, and **welcome**. Each application has associated buttons for **Edit**, **About**, **Errors**, and **Clean**. A hand holding a smartphone is positioned in front of the interface, displaying a mobile version of the application list. The phone screen shows the same list of applications and a footer that reads "powered by web2py - ©2012". To the right of the application list, there are several sections for system status and actions:

- Change admin password** button.
- Version 2.2.1 (2012-11-13 10:24:37) stable** with a note **Unable to check for upgrades** and **Running on Rocket 1.2.5**. A link **Try the mobile interface** is also present.
- New application wizard** section with a **Start wizard** button (noting it requires internet access).
- New simple application** section with an **Application name:** input field and a **Create** button.
- Upload and install packed application** section with an **Application name:** input field, an **Upload a package:** section with a **Choose File** button (showing "No file chosen"), and an **Or Get from** input field.

# Thin-IDE: only shows file system, no metadata



# Web based editor (code-mirror)

The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000/admin/default/edit/welcome/controllers/default.py?id=controllers__default__py`. The browser's tab is labeled "edit welcome/controllers/de". The page has a dark header with the "WEB2PY" logo and navigation buttons: "Site", "Edit", "About", "Errors", "Versioning", "Logout", "Debug", and "Help".

The main content area is titled "EDITING FILE 'WELCOME/CONTROLLERS/DEFAULT.PY'". Below the title, a grey box contains the text: "exposes: index, user, download, call, data" and "edit views: index, user".

Below this, a status bar shows "Saved file hash: 355e75b88ee524c96b" and "Last saved on: Fri Oct 19 14:12:09 201". To the right of the status bar are buttons for "toggle breakpoint", "<<back", and "docs".

The code editor displays a Python file with line numbers 1 through 19. The code is as follows:

```
1 # -*- coding: utf-8 -*-
2 # this file is released under public domain and you can use it as you wish
3
4 #####
5 ## This is a samples controller
6 ## - index is the default action
7 ## - user is required for authentication
8 ## - download is for downloading
9 ## - call exposes all registered services
10 #####
11
12
13 def index():
14     """
15     example action using the internationalization operator T and flash
16     rendered by views/default/index.html or views/generic.html
17
18     if you need a simple wiki simple replace the two lines below with:
19     return auth.wiki()
```

A callout bubble highlights the `def index():` and `return "hello world"` lines, indicating the current state of the code being edited.

# Web based database administration (per app)

SQLite, MySQL, PostgreSQL, MSSQL, Firebird, Oracle, DB2, Ingres, Informix, Ingres, Sybase, GAE, ...

The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000/newapp/appadmin/index#`. The page features a dark header with the **web2py™** logo on the left and a **Login** button on the right. A dropdown menu is open next to the login button, showing options: **Register**, **Lost password?**, and **Login**. Below the header, the main content area has the title **Newapp** followed by the subtitle **Database Administration (appadmin)**. A section titled **Available Databases and Tables** lists several database tables with corresponding **New Record** buttons:

<b>db.auth_user</b>	New Record
<b>db.auth_group</b>	New Record
<b>db.auth_membership</b>	New Record
<b>db.auth_permission</b>	New Record
<b>db.auth_event</b>	New Record
<b>db.auth_cas</b>	New Record
<b>db.scheduler_task</b>	New Record
<b>db.scheduler_run</b>	New Record
<b>db.scheduler_worker</b>	New Record

Annotations with callout boxes identify specific features: **Twitter Bootstrap Layout** points to the header; **Auth Actions** points to the login/register dropdown; **Role Based Access Control Tables** points to the **db.auth\_\*** tables; and **Scheduler Tables** points to the **db.scheduler\_\*** tables. A **Share** button is visible in the bottom right corner.

# Web translation page for internationalization (per app)

The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000/admin/default/edit_language/welcome/languages/it.py`. The page title is "EDITING LANGUAGE FILE 'WELCOME/LANGUAGES/IT.PY'". A navigation bar at the top includes links for Site, Edit, About, Errors, Versioning, Logout, Debug, and Help. The main content area is titled "ORIGINAL/TRANSLATION" and contains several input fields for editing language strings, each with a "delete" button.

Hide/Show Translated strings

● ORIGINAL/TRANSLATION

!=

!= delete

!langcode!

it delete

!langname!

Italiano delete

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

delete

%(nrows)s records found

%(nrows)s records found delete

%d seconds ago

%d seconds ago delete

%s %%%{row} deleted

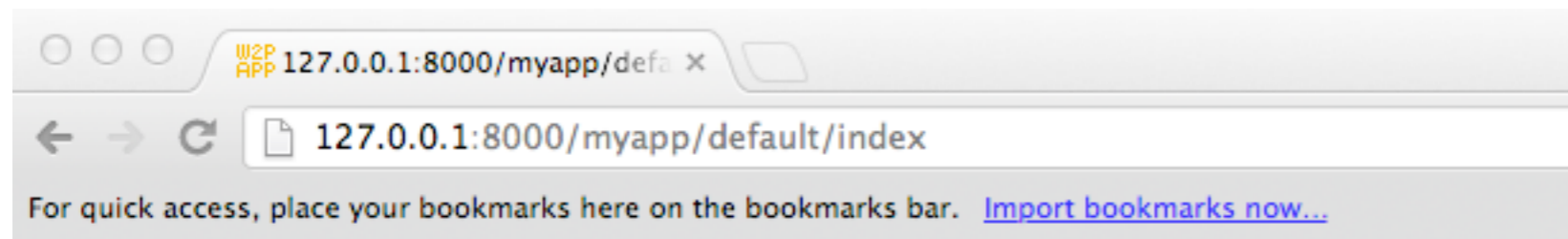
# Built-in pluralization system

The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000/admin/default/edit_plurals/welcome/languages/plural-en.py?nplurals=2`. The page title is "EDITING PLURAL FORMS FILE 'WELCOME/LANGUAGES/PLURAL-EN.PY'". The interface includes a navigation bar with buttons for Site, Edit, About, Errors, Versioning, Logout, Debug, and Help. The main content area displays a table for editing plural forms. The table has three columns: Singular Form, Plural Form #1, and a delete button. The rows contain the following data:

Singular Form	Plural Form #1	
account	accounts	delete
book	books	delete
is	are	delete
man	men	delete
miss	misses	delete
person	people	delete
quark	quarks	delete
shop	shops	delete
this	these	delete
was	were	delete
woman	women	delete

Below the table is an "update" button. At the bottom of the page, a footer bar states: "Powered by web2py™ created by Massimo DI Pierro ©2007-2012 - Admin language English (US)".

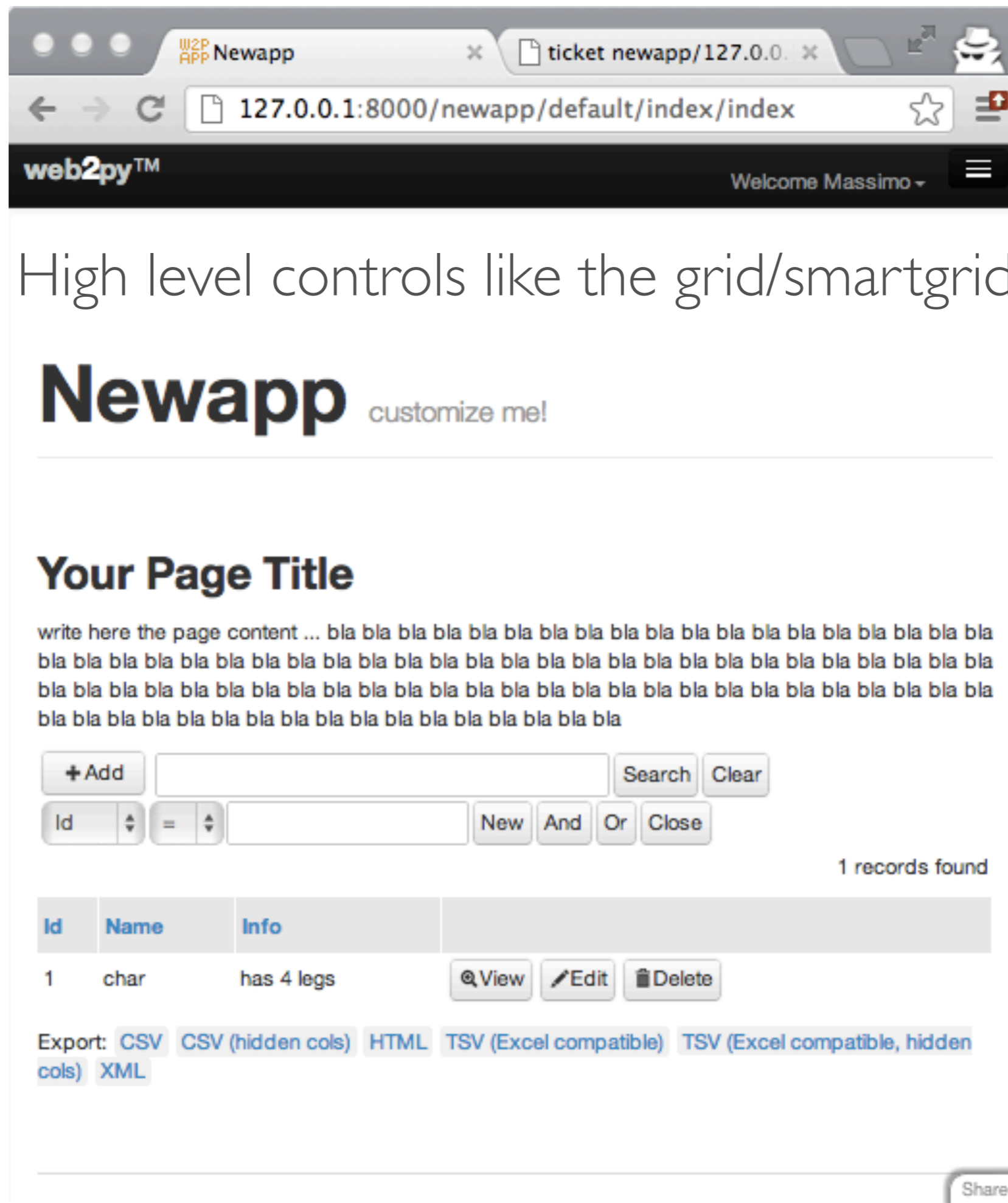
# Built-in ticketing system



## Internal error

Ticket issued: [myapp/127.0.0.1.2012-11-16.07-17-58.f9a9fd5d-5507-4cae-bb1d-2e697f3145f9](#)

A screenshot of the Web2py admin interface. The browser address bar shows '127.0.0.1:8000/admin/default/errors/myapp#'. The page title is 'ERROR LOGS FOR "MYAPP"'. There are buttons for 'check all', 'uncheck all', and 'delete all checked'. Below these, there is a green circle icon and the text 'Click row to expand traceback'. There are also buttons for 'source : filesystem', 'switch to : db', and 'lists by ticket'. A table with columns 'Delete', 'Count', 'File', and 'Error' is shown. The table has one row with a checkbox, the count '1', the file 'default.py', and the error 'ZeroDivisionError: integer division or modulo by zero'. To the right of the error message is a '+ details' link. Below the table, there is a traceback showing the stack of calls. The footer of the page says 'Powered by web2py™ created by Massimo Di Pierro ©2007-2012 - Admin language English (US)'.



# SYNTAX



```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, World");  
    }  
}
```

VS

```
print "hello world"
```

KEEP NEW PROGRAMMERS IN MIND

# BOTTLE EXAMPLE

```
from bottle import run, route, get, static_file
```

required inputs

```
@get('/index')
```

routing logic

```
def index()
```

action

```
    return 'hello world'
```

```
@route('/static/<filename>')
```

```
def server_static(filename):
```

handler for static files

```
    return static_file(filename, root='static')
```

```
run(host='localhost', port=8080)
```

start web server

# FLASK EXAMPLE

```
from flask import Flask, request
```

required inputs

```
app = Flask(__name__)  
app.config.from_object(__name__)
```

boilerplate config  
logic

```
@app.route('/index', methods=['GET'])  
def index()  
    return 'hello world'
```

routing logic

action

```
app.run(port=8080)
```

start web server

# TORNADO EXAMPLE

```
import tornado.ioloop
import tornado.web
```

required inputs

```
def index(request):
    return 'hello world'
```

action

```
class MainHandler(tornado.web.RequestHandler):
    def get(self): return index(self.request)
```

routing logic

```
application = tornado.web.Application([
    (r"/index", MainHandler),
    (r"/static/(.*)", tornado.web.StaticFileHandler, {"path": "static"})])
```

handler for static files

```
application.listen(8080)
tornado.ioloop.IOLoop.instance().start()
```

start web server

# PYRAMID EXAMPLE

```
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response
from pyramid.static import static_view
```

```
def index(context, request):
    return Response('hello world')
```

```
config = Configurator()
config.add_route('index', '/index')
config.add_view(index, route_name='index')
config.add_static_view(name='static', path='static')
app = config.make_wsgi_app()
server = make_server('0.0.0.0', 8080, app)
server.serve_forever()
```

required inputs

action

routing logic

handler for static  
files

start web server

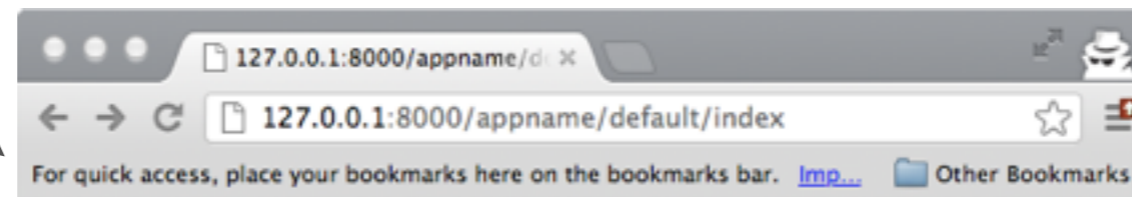
# WEB2PY EXAMPLE

<http://127.0.0.1:8000/appname/default/index>

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

call

action



Hello world

...HUH?



# IMPORT VS EXEC



# IMPORT VS EXEC

user app

imports



framework

```
from bottle import ...  
from flask import ...  
from tornado import ...  
from pyramid import ...
```

explicit better  
than implicit

framework

executes



user app

... app

... app

```
env = build_environment(request)  
app = find_application(request)  
exec app in env (oversimplification)
```

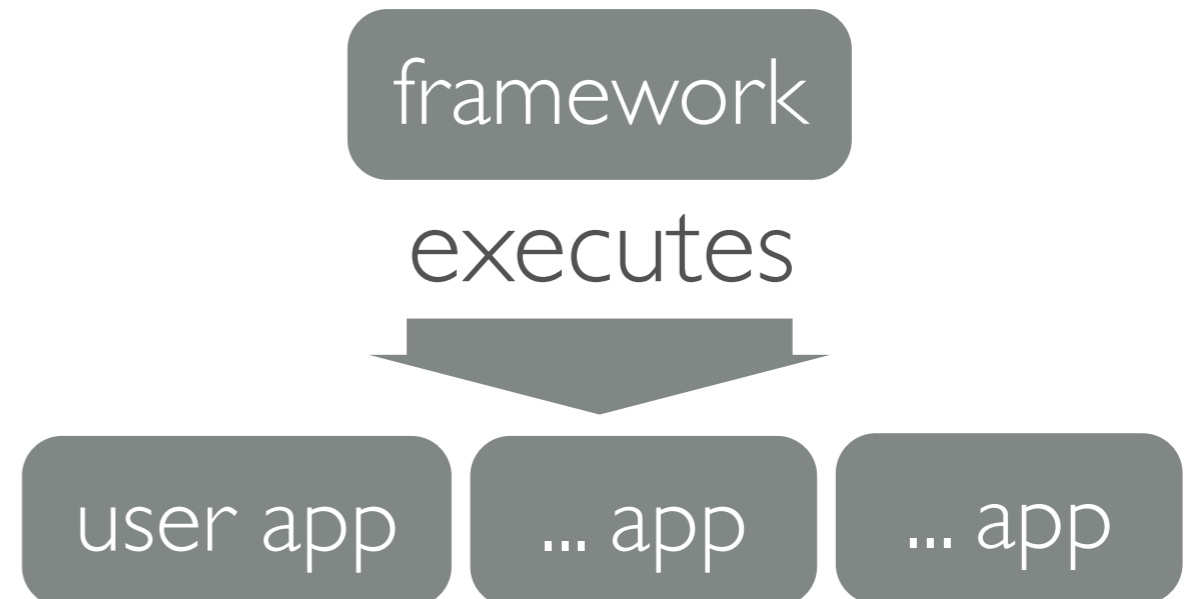
do not repeat  
yourself

convention over  
configuration

# IMPORT VS EXEC



faster (for simple apps)  
more flexibility  
no “magic”



less code (for simple apps)  
how swap of code  
multi app/multi project  
homogeneous environment  
“magic”

# LAYERS OF CODE

SQL inside Python

(DAL or ORM)

```
execute('select * from users where id=1')
```

```
db(db.users.id==1).select()
```

HTML inside CODE

(helpers)

```
return '<div><h1>%s</h1></div>' % x
```

```
return DIV(H1(x))
```

CODE in HTML

(MVC)

```
<div>{{if x}}check{{endif}}</div>
```

```
<div>{{if x:}}check{{pass}}</div>
```

JS in HTML

```
<div><script>alert('hi!')</script></div>
```

```
<div>{{=LOAD('action',ajax=True)}}</div>
```

# WEB2PY DAL

- ▶ SQLite, MySQL, PostgreSQL, MSSQL, Firebird, Oracle, DB2, Ingres, Informix, Ingres, Sybase, GAE, ...
- ▶ automatic migrations
- ▶ multiple dbs, connection pooling, Round Robin redundancy, distributed transactions
- ▶ joins, left joins, aggregates, nested selects, recursive selects

```
db = DAL('postgresql:...', pool_size=10)
db.define_table('person', Field('name'))
db.define_table('thing', Field('name'), Field('owner', db.person))
db.thing.insert(name='PC', owner=db.person.insert(name='John'))

ownership = (db.person.id == db.thing.owner)
thing_counter = db.thing.id.count()
rows = db(ownership).select(db.person.name, thing_counter,
                             groupby=db.person.id)

for row in rows: print row.person.name, row(thing_counter)
```

# PROGRAMMING AS WIKI

```
# models/db.py
db.define_table('thing',
    Field('name'),
    Field('info', 'test'))

# controllers/default.py
def index():
    return auth.wiki()

def things():
    return SQLFORM.grid(db.thing)
```

# PROGRAMMING AS WIKI

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000/newapp/default/index/\_edit/index/0'. The page title is 'Newapp'. The main content area contains a form with the following elements:

- Title:** A text input field containing the word 'Index'.
- Body:** A large text area containing placeholder text: '## Your Page Title' followed by several lines of 'bla bla bla'. At the bottom of this area, the text '@{component:default/things}' is highlighted with an orange box. An orange arrow points from this box to the right.
- Tags:** A text input field followed by a '+' button.
- Changelog:** A text input field.
- Submit:** A button at the bottom left of the form.

The image is a screenshot of a web browser displaying the 'Newapp' application. The browser's address bar shows the URL '127.0.0.1:8000/newapp/default/index/index'. The page has a dark header with the 'web2py™' logo on the left and 'Welcome Massimo' on the right. The main content area features a large heading 'Newapp' with the subtitle 'customize me!'. Below this is a section titled 'Your Page Title' followed by a paragraph of placeholder text. A table is shown with one record. The table has three columns: 'Id', 'Name', and 'Info'. The record has the value '1' in the 'Id' column, 'char' in the 'Name' column, and 'has 4 legs' in the 'Info' column. To the right of the table, it says '1 records found'. Below the table are three buttons: 'View', 'Edit', and 'Delete'. At the bottom of the page, there are export options: 'Export: CSV', 'CSV (hidden cols)', 'HTML', 'TSV (Excel compatible)', 'TSV (Excel compatible, hidden cols)', and 'XML'.

# CONCLUSIONS

- ▶ We need to build a society where people understand and control technology, not vice versa.
- ▶ We need to build tools that are easy to use to allow more people to use technology for the public good
- ▶ web2py is one of such tools
- ▶ web2py reduces entry barrier to web programming
- ▶ web2py reduces maintenance costs for large projects