Advanced web2py

massimo.dipierro@gmail.com

web2py start in 2007

- one of the most popular web frameworks
- 2011 Bossie Award
- 2012 Technology of the Year Award
- 2 books
- about 6000 registered users

How did we get here?

Priorities

- Ease of use (+ expressive, - maintenance)
- Security (no choices to developers)

- Batteries included
- Convention over configuration

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

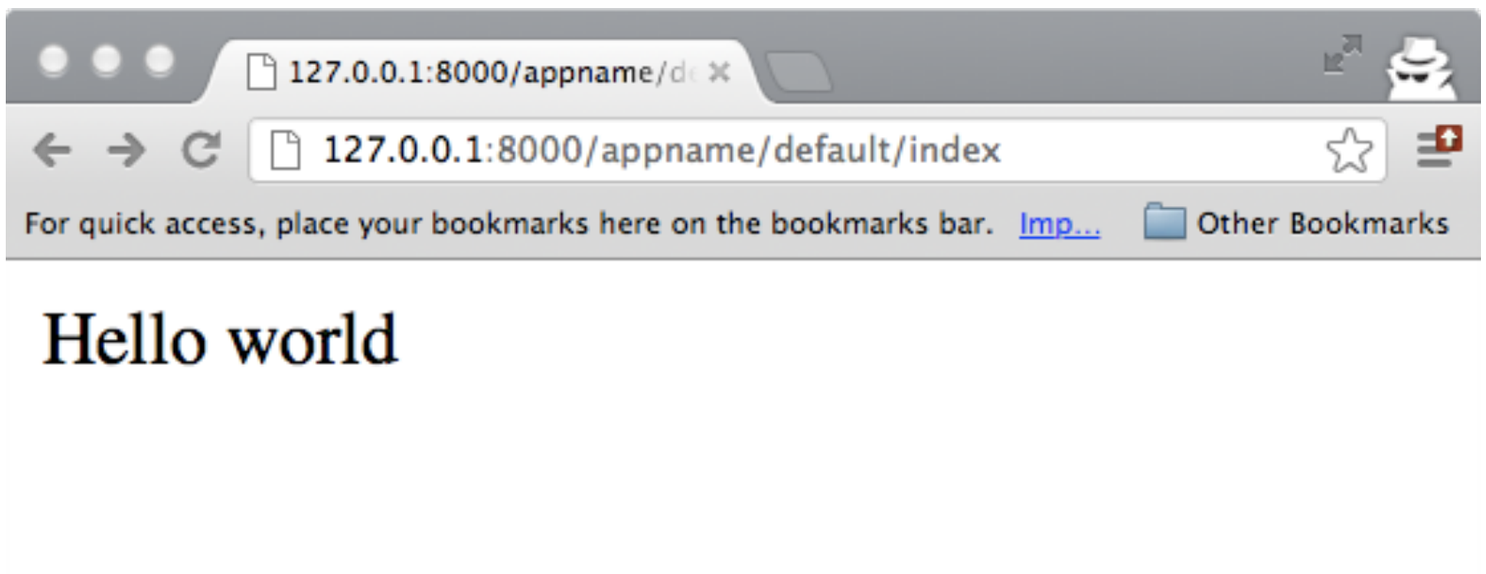
ticketing system

No installation. No configuration. Just Unzip and Click!

Convention over configuration (*a la* RoR)

applications/appname/controllers/default.py

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



Models

```
book
    - title
    - authors
    - description
    - cover_image
```

Models

```
db.define_table(
    'book',
    Field('title'),
    Field('authors'),
    Field('description'),
    Field('cover_image'))
```

Models

```
db.define_table(
    'book',
    Field('title',requires=IS_NOT_EMPTY()),
    Field('authors','list:string'),
    Field('description','text'),
    Field('cover_image','upload'))
```

Insert example

```
db.define_table(
   'book',
   Field('title',requires=IS_NOT_EMPTY()),
   Field('authors','list:string'),
   Field('description','text'),
   Field('cover_image','upload'),
   auth.signature)

db.book.insert(title='web2py')
```

Select example

```
db.define_table(
   'book',
   Field('title',requires=IS_NOT_EMPTY()),
   Field('authors','list:string'),
   Field('description','text'),
   Field('cover_image','upload'),
   auth.signature)

books = db(db.book.title=='web2py').select()
```

Form example

```
db.define_table(
   'book',
   Field('title',requires=IS_NOT_EMPTY()),
   Field('authors','list:string'),
   Field('description','text'),
   Field('cover_image','upload'),
   auth.signature)

create_form = SQLFORM(db.book).process()
```

Crud example

```
db.define_table(
   'book',
   Field('title',requires=IS_NOT_EMPTY()),
   Field('authors','list:string'),
   Field('description','text'),
   Field('cover_image','upload'),
   auth.signature)

edit_form = SQLFORM(db.book,1).process()
```

Grid Example

```
db.define_table(
   'book',
   Field('title',requires=IS_NOT_EMPTY()),
   Field('authors','list:string'),
   Field('description','text'),
   Field('cover_image','upload'),
   auth.signature)

grid = SQLFORM.grid(db.book)
```

Auditing

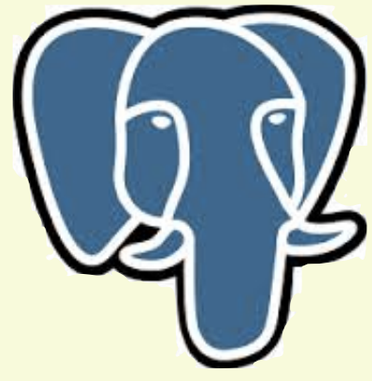
```
db.define_table(
   'book',
   Field('title',requires=IS_NOT_EMPTY()),
   Field('authors','list:string'),
   Field('description','text'),
   Field('cover_image','upload'),
   auth.signature)

auth.enable_record_versioning(db) # auditing
```

Portability

```
db = DAL('sqlite://storage.db')

db.define_table(
   'book',
   Field('title',requires=IS_NOT_EMPTY()),
   Field('authors','list:string'),
   Field('description','text'),
   Field('cover_image','upload'),
   auth.signature)

...
```

Portability

```
db = DAL('postgres://user:password@localhost/test')

db.define_table(
   'book',
   Field('title',requires=IS_NOT_EMPTY()),
   Field('authors','list:string'),
   Field('description','text'),
   Field('cover_image','upload'),
   auth.signature)

...
```



Google App Engine

```
db = DAL('google:datastore')

db.define_table(
   'book',
   Field('title',requires=IS_NOT_EMPTY()),
   Field('authors','list:string'),
   Field('description','text'),
   Field('cover_image','upload'),
   auth.signature)

...
```

Mongo DB

```
db = DAL('mongodb://user:password@server:port/db')

db.define_table(
   'book',
   Field('title',requires=IS_NOT_EMPTY()),
   Field('authors','list:string'),
   Field('description','text'),
   Field('cover_image','upload'),
   auth.signature)

...
```

MVC

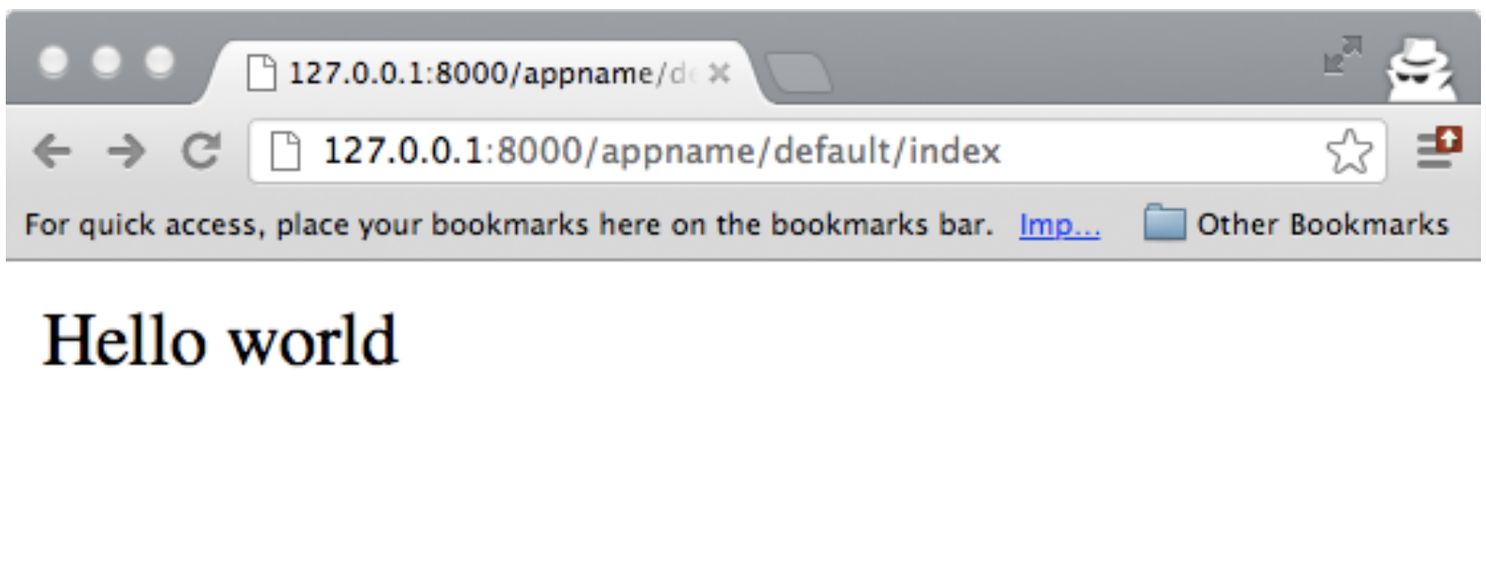
models/db.py

```
from gluon.tools import Auth
db = DAL('sqlite://storage.db')
auth = Auth(db)
auth.define_tables()
db.define_table(
    'book',
    Field('title',requires=IS_NOT_EMPTY()),
    Field('authors','list:string'),
    Field('description','text'),
    Field('cover_image','upload'),
    auth.signature)
```

MVC

controllers/default.py

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



MVC

controllers/default.py

```
def index():
    return {'g': SQLFORM.grid(db.book)}
```

MVC

controllers/default.py

```
@auth.requires_login()
def index():
    return {'g': SQLFORM.grid(db.book)}

def books():
    return {'rows': db(db.book).select().as_list()}
```

MVC

views/default/index.py

```
{{extend 'layout.html'}}
<h1>Interface to manage my books</h1>
<div>
   {{=g}}
</div>
```

Complete program

models/db.py

```
from gluon.to
db = DAL('sql
auth = Auth(d
auth.define_t
db.define_table(
    'book',
    Field('tit
    Field('aut
    Field('des
    Field('cov
    auth.signa
```

controllers/default.py

```
@auth.requires_login()
def index():
    return {'g': SQLFORM.grid(db.book)}
```

views/default/index.py

```
{{extend 'layout.html'}}
<h1>Interface to manage my books</h1>
<div>
   {{=g}}
</div>
```

Access Control

```
@auth.requires_login()
def index():
    return {}
```

Access Control

```
@auth.requires(auth.has_membership('mygroup') or …)
def index():
    return {}
```

use lambda

```
@auth.requires(lambda:auth.has_membership('mygroup') or …)
def index():
    return {}
```

Better because evaluated only if function is called

global permissions

```
if not(auth.has_membership('mygroup') or …):
     redirect(URL('error_controller','my_error_page'))

def index1():
    return {}

def index2():
    return {}

def index3():
    return {}
```

Avoid code duplication

auth.user_groups

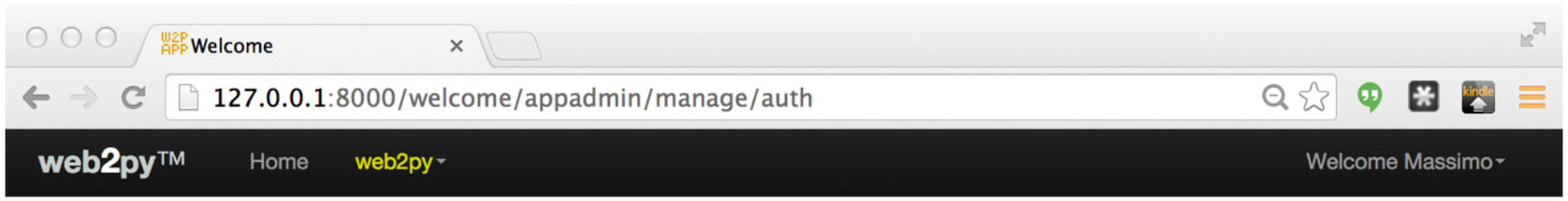
```
if auth.has_membership('mygroup'): # DB ACCESSS
```

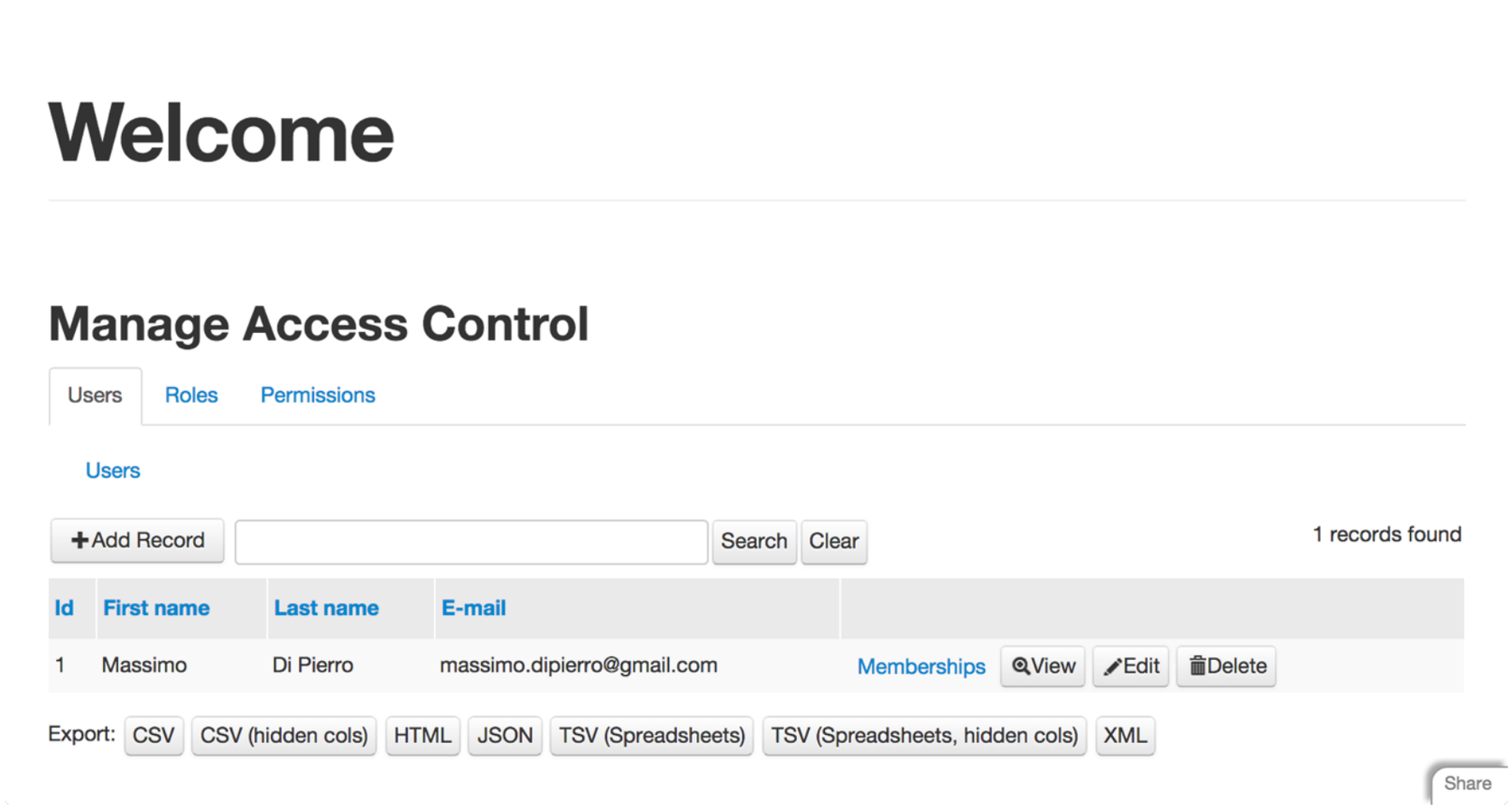
```
if 'mygroup' in auth.user_groups.values(): # NO DB ACCESS
```

auth.user_groups = {<id>:<role>, …}

appadmin/manage/auth

```
auth.settings.auth_manager_role = 'manager'
if auth.user and not db.auth_group(role='manager'):
    auth.add_membership(auth.add_group('manager'))
```





Ajax Widgets

Ajax components

```
def index():
    return {}

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

views/default/index.py

```
....
<div>
{{=LOAD('default','grid',ajax=True)}}
</div>
...
```

The built-in wiki

```
def index(): return auth.wiki()
```

is it a wiki?

is it a blog?

is it a CMS?

is it a way to set permissions on content?

is it a way to store pages in db?

is it a way to store code in db?

The built-in wiki

```
auth.add_memberhsip(auth.add_group(role='wiki_author'))
auth.add_memberhsip(auth.add_group(role='wiki_editor'))

return auth.wiki(render='markmin')
return auth.wiki(render='html')
return auth.wiki(manage_permissions = True)
return auth.wiki(force_prefix='%s-' % auth.user.username)
```

The built-in wiki (extra)

```
return auth.wiki(extra=dict(x = lambda text: ‘x’+text+’x’))

MARKMIN(“…”, extra=dict(par = lambda text: ‘(’+text+’)’))

        ``hello``:par -> (hello)
```

The built-in wiki (env)

```
return auth.wiki(env = dict(double=lambda a: a+a))

MARKMIN("…", env = dict(double=lambda a: a+a))

        @{double:hello}  -> hellohello
```

Task scheduler

Task scheduler

```
from gluon.scheduler import Scheduler
scheduler = Scheduler(db)

def f(n): return sum(i for i in range(n))

scheduler.queue_task(f, pvars=dict(n=8),
    start_time=request.now, repeats=5, period=10,
    timeout=60, retry_failed=3)
```

web2py.py -K app
web2py.py -K app
web2py.py -K app

Restful API

REST API

```
def api():
    if request.env.request_method == ‘GET’:
        return db(db.book).select().as_json()
    if request.env.request_method == ‘POST’
        return {'id': db.book.insert(title=request.vars.title)}
    raise HTTP(400)
```

Restful API

REST API

```
@response.restful()
def api():
    def GET():
        return db(db.book).select().as_json()
    def POST(title):
        return {'id': db.book.insert(title=title)}
    return locals()
```

Collection + JSON

Self-Documenting Hypermedia API

```
def api():
    from gluon.contrib.hypermedia import Collections

    policy = {‘book’:{’GET’:{},‘POST’:{},‘PUT’:{},‘DELETE’:{}}

    return Collection(db).process(request,response,policy)
```

Stripe.com

```
def pay():
    from gluon.contrib.stripe import StripeForm
    form = StripeForm(
        pk='pk_test_6pRNASCoBOKtIshFeQd4XMUh',
        sk='sk_test_BQokikJOvBiI2HlWgH4olfQ2',
        amount=int(150), # $1.5 (amount is in cents)
        description="you bought a candy!")
    if form.process().accepted:
        # db.purchase.insert(receipt_id = form.response['id'])
        redirect(URL('thankyou'))
    return dict(form=form)
```

My favorite JS libs

```
jQuery.js             # you know
quo.js                # touch events
sugar.js              # dates, arrays, objects, debounce, throttle
ractive.js            # client-side dynamic templates
handsontable.js       # tabular data with excel export
highcharts.js         # plotting (non free)
fullcalendar.js       # for displaying events (day/week/month)
together.js           # for collaboration (share page view)
```

ractive.js

```
def index():
    return dict(a=“hello”, b=“world”)
```

```
<html>
 <body>
    <div id="target"></div>
    <script id="template" type=“text/ractive">
       <input value="[[a]]"/> + <input value="[[b]]"/>
       = [[a + b]]
    </script>
  </body>
  <script src="js/jquery.js"></script>
  <script src=“js/sugar.min.js"></script>
  <script src=“js/ractive.min.js"></script>
  <script>{{=ASSIGNJS(a=a, b=b)}}</script>
  <script src="js/page.js"></script>
</html>
```

ractive.js

```
/* in js/page.js */

page = new Ractive({
        el: "target",
        template: '#template',
        delimiters: ['[[',']]'],
        tripleDelimiters: ['[[[',']]]'],
        data: { a:a, b:b }
    });
```

OWASP TOP 10

A1: Injections

- web2py's DAL prevents SQL Injections
- URLs are validated by default
- All variables embedded in HTML are escaped

OWASP TOP 10

A2: Authentication and Session Managent

- web2py manages sessions for you
  - session on filesystem (session token uuid)
  - or in database or cookies (encrypted and signed)
- token uuid cleared on logout
- short expiration
- integration with third party Auth mechanisms

OWASP TOP 10

A3: XSS

- All code in HTML {{=value}} is always escaped
- Auth redirects only allowed from localhost
- Urls can optionally be signed
- (grid urls always signed by default)

OWASP TOP 10

A4: Insecure direct object references

- About Admin
- Admin only accessible from localhost or over HTTPS
- Admin can be disabled
- Admin can be removed

OWASP TOP 10

A5: Security Misconfiguration

- There is very little security configuration in web2py
- Strongest choices are default:
  - password strength check
  - password salting + hashing (pbkdf2)

OWASP TOP 10

A6: Sensitive data exposure

- session.secure()
- automatic session.clear() on logout
- Field(…,'password',…) to "*****"
- Field(…, 'upload', authorization=…)
- Field(…, readable = …, writable = …)
- Collection(… rules=…) for REST Hypermedia API
- Stripe payment system (CISP compliant)

OWASP TOP 10

A7: Missing Function Level Access Control

- @auth.requires_login(…)
- @auth.requires_membership(…)
- @auth.requires_permission(…)
- @auth.requires(lambda:…)
- @auth.requires_signature()

OWASP TOP 10

A8: CRSF

- All web2py forms use CSRF protection by default
- CSRF tokens are one time UUIDs
- Pages with forms can optionally be signed

OWASP TOP 10

A9: Using components with known vulnerabilities

- This is why we ship authentication libraries
- … credit card payment libraries and …
- … database adapters …
- with web2py.
- we monkeypatch pymysql (shipped with web2py)

OWASP TOP 10

A10: Unvalidated redirects and forwards

- Auth redirects only allowed from localhost
- URLs can optionally be signed
- All forms perform postbacks with CSRF protection
- Explicit prevention of open redirects in Auth
- including password reset attacks.

Conclusions and Challages

- web2py is 8 years old and very mature
- The world has changed
- From Python 2 to Python 3
- Client side programming more important
- JS and CSS frameworks are a zoo