# web2py overview

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?

Daniel.Lin
Jose.L..Redrejo.RodriguezHans.Murx
Douglas.Soares.de.AndradeNiccolo.Polo
Jose.Vicente.de.Sousa
Denes.Lengyel
Robert.Valentak
Nicolas.Bruxer
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
CJ.Lazell
Graham.Dumpleton
Angelo.Compagnucci
Alexey.Nezhdanov
Thadeus.Burgess
Phyo.Arkar.Lwin
Simone.Bizzotto
Stuart.Rackham
Niall.Sweeny
Patrick.Breitenbach
Gilson.Filho
Dave.Stoll
Fran.Boon
Jose.Jachuf
Mark.Larsen
Anders.Roos
Hans.C..v..Stockhausen
Carsten.Haese
Gyuris.Szabolcs
Chris.Steel
Zahariash
Ruijun.Luo
Kenji.Hosoda
Timothy.Farrell
Christian.Foster.Howes
Scott.Roberts
Carlos.Galindo
Limodou
Yair.Eshel
Chris.Clark
Jan.Beilicke
Ben.Goosman
Branko.Vukelic
Sergey.Podlesnyi
Ian.Reinhart.Geiser
Christopher.Smiga
Jonathan.Lundell
Sterling.Hankins
Paolo.Caruccio
Anthony.Bastardi
Telman.Yusupov
David.Wagner
Jonathan.Benn
Martin.Mulone
Josh.Goldfoot
Alan.Etkin
Tim.Michelsen
Lucas.D'Avila
Mark.Moore
Keith.Yang
Sriram.Durbha
Ondrej.Such
Markus.Gritsch
Francisco.Gama
Brian.Meredyk
Mariano.Reingart
Ovidio.Marinho.Falcao.Neto
Craig.Younkins
Boris.Manojlovic
Michael.Willis
Vinicius.Assef
Olaf.Ferger
Farsheed.Ashouri
Sharriff.Aina
Hans.Donner
Nathan.Freeze
Fred.Yanowski
Ross.Peoples
Younghyun.Jo
Marcel.Hellkamp
Hamdy.Abdel-Badeea
Marcello.Della.Longa
Yarko.Tymciurak
Andrew.Willimott
Martin.Hufsky
Robin.Bhattacharyya
Josh.Jaques
Marin.Pranjic
Alvaro.Justen
Bruno.Rocha
Kyle.Smith
Ryan.Seto
Eric.Vicenti
Chris.May

## Priorities

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

- Batteries included
- Convention over configuration

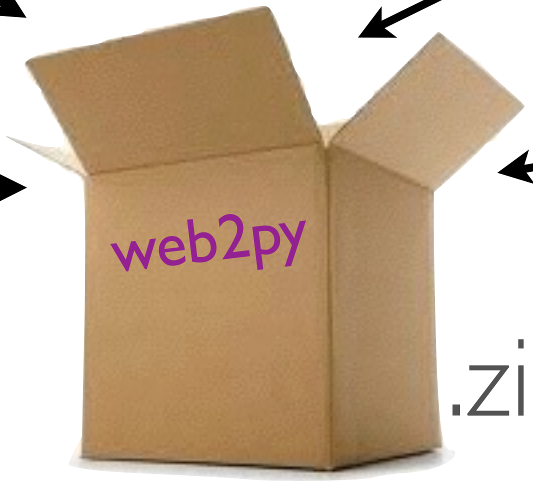
# Batteries included

## 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

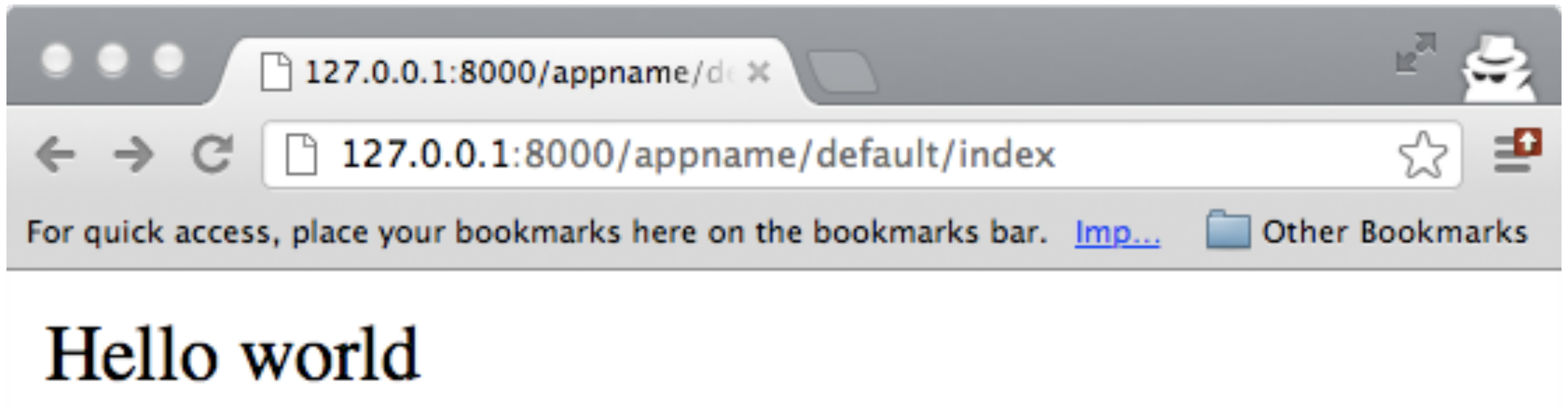
## ticketing system

No installation. No configuration. Just Unzip and Click!

## Convention over configuration (*a la* RoR)

`applications/appname/controllers/default.py`

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



127.0.0.1:8000/appname/d

127.0.0.1:8000/appname/default/index

For quick access, place your bookmarks here on the bookmarks bar. Imp... Other Bookmarks

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

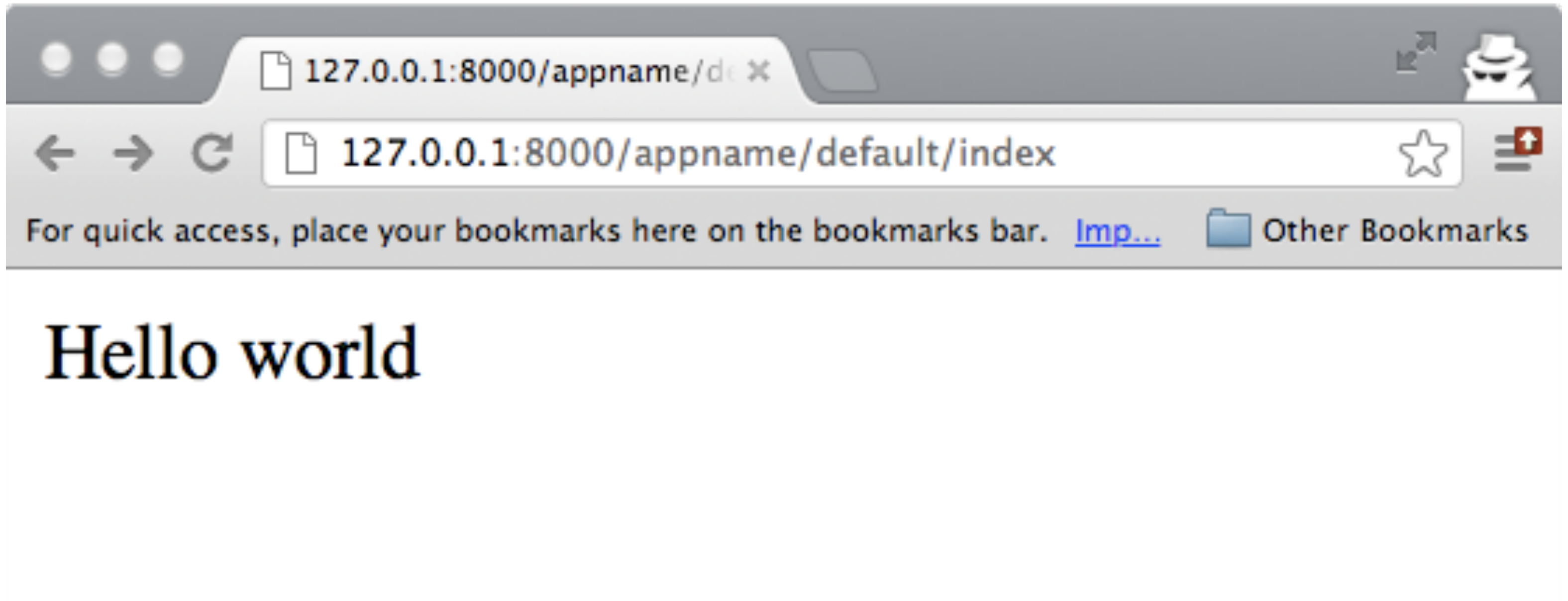
models/db.py

```
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"
```


127.0.0.1:8000/appname/d
127.0.0.1:8000/appname/default/index
For quick access, place your bookmarks here on the bookmarks bar. Imp...
Other Bookmarks

Hello world

## MVC

controllers/default.py

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

# MVC

controllers/default.py

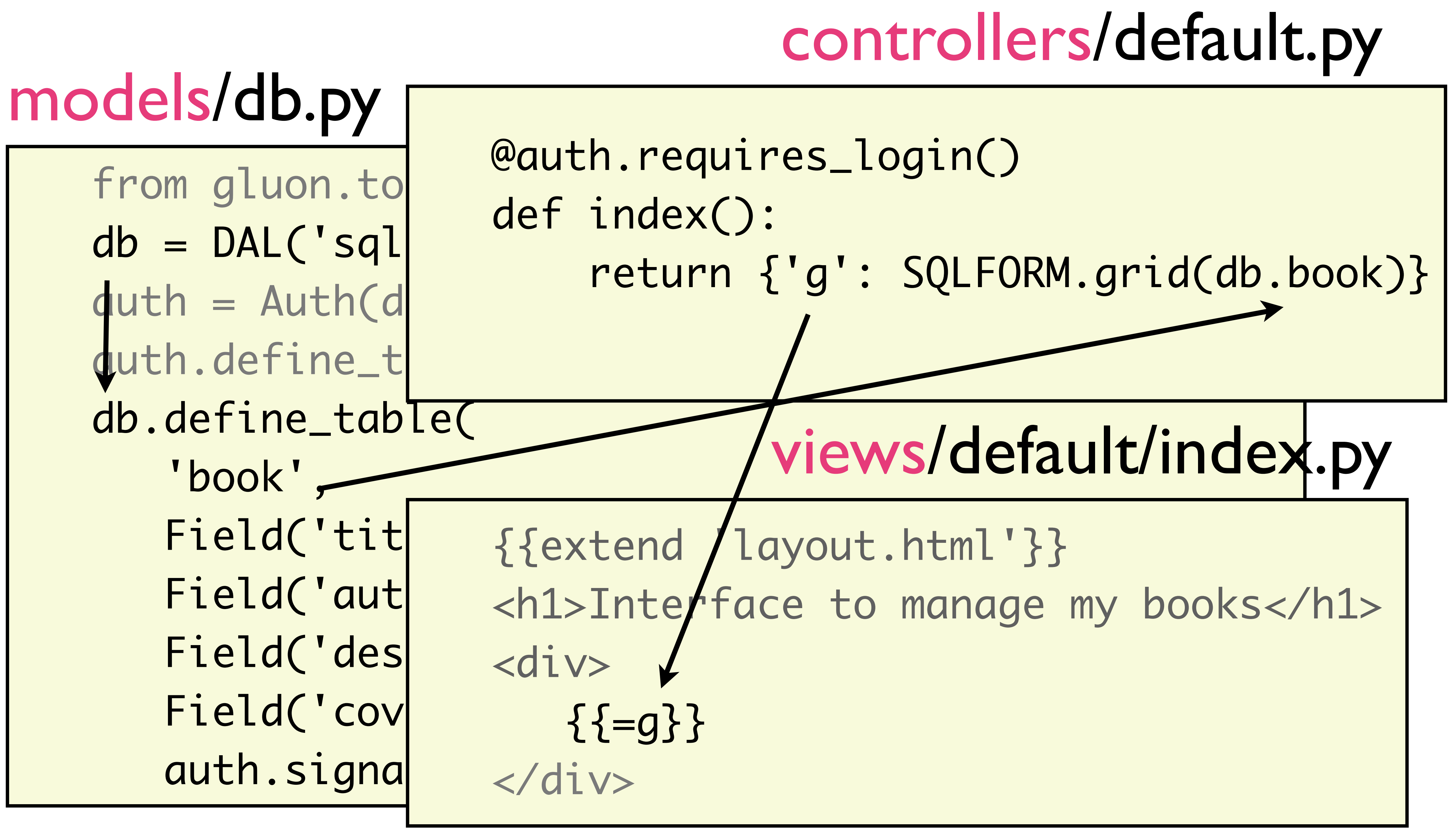
```
@auth.requires_login()
def index():
    return {'g': SQLFORM.grid(db.book)}

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>
```

# Less known stuff

Ajax components

```
def index():
    return {}

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

views/default/index.py

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

# Less known stuff

## Task scheduler

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

## Built-in WIKI / OEMBED

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

## Less known stuff

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 Hypermedia API

```
def API():
    from gluon.contrib.hypermedia import Collections
    rules = {…}
    return Collection(db).process(request,response,rules)
```

# 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