

An Overview of Turbogears the Web “Mega Framework”

Mark Biggers¹

¹biggers@utsl.com – Python & Linux consulting

TriZPUG meeting, 26 Sep 2006

Outline

- 1 Introduction
 - who, what
 - Python module integration
- 2 Bootstrapping a TurboGears project
- 3 Get into the TG Code!
 - doing Model-View-Controller
 - support for “smart” Forms

Who are You?

- Zope2, Plone developer, or CMS manager?
- Java-J2EE/.NET/PHP seeker, of the Python way?
- PyServlet Engine, Skunkweb, Spyce ... “Python web” developer?

Who am I?

- Mark Biggers, software & systems engineer
- Python developer, PgSQL DBA, (former) WebSphere admin
- b4 W3: X Window/Motif, C, C++ devel & training
- 1996-2006: Web related experience
 - Python/web for UnitedMedia
 - Perl/web for Eli Lilly
 - Tcl/web (OpenACS) for LoneSource.com
 - Plone BootCamp (trizpug.org) 2005
 - Zope2 apps, Twisted, Py develop, maintain

What is TurboGears?

- “front-to-back” Web framework
- buzzword-compliant:
MVC, ORM, AJAX/JSON, RSS, SOAP, TDD, DRY, ...
- get C.R.U.D. with your Web
- TG is *not* Plone! *not* a *Content Management System*

TG integrates major Python modules

- **SQLObject**: SQL database repr, in Python classes [M]
- **Kid**: XHTML & XML templating engine [V]
- **CherryPy**: Web “I/O” [C]
- **Mochikit**: Javascript (“AJAX”) browser GUI [V]

Kevin Dangoor's bits

- TG Toolbox: CatWalk, ModelDesigner [M,V]
- form Widgets: see the WidgetBrowser
- Identity: authenticate, authorize [C,M]
- TG validators; error-handlers; controller “views”
- “TurboXYZ”: a Py module, integrated by Kevin

other Py community bits

- FormEncode: validate, convert Form values [C]
- WSGI support, using "Paste" [C?]
- RuleDispatch: P.E.A.K. rules engine, flexible auth/auth [C]
- nose: PyUnit unit-testing assistance

tg-admin, help!

- sql: create, update DB from SQLAlchemy class defs
- toolbox: Web-based DB design, DB editing, etc
- shell: “SQLAlchemy command-line” to your DB/model

```
$ tg-admin help
TurboGears 1.0b1 command line interface

Usage: /usr/local/bin/tg-admin [command] [options]

AVAILABLE COMMANDS:

quickstart  Create a new TurboGears project
  sql       Run the SQLAlchemy manager
  toolbox   Launch the TurboGears Toolbox
  shell     Start a IPython prompt with your DB
.....
```

tg-admin quickstart

- quickstart: generate a “basic” TG project
- fill in the blanks: controllers.py, model.py, *.kid

```
tgapp/ dev.cfg prod.cfg devdata.sqlite
      tgapp/ controllers.py model.py
          templates/ master.kid welcome.kid ...
          static/ css/ images/ javascript/
          subcontrollers/ __init__.py a_controller.py (*)
```

python start-tgapp.py

- runs CherryPy HTTP server, on your project-code
- configuration: `tgapp/dev.cfg` (or) `tgapp/prod.cfg`
- `dev.cfg`: specify DB URL; HTTP port; log-levels

how does TG do C.R.U.D.

- Object-Relational Manager: SQLAlchemy
- 2nd choice (no Toolbox support): SQLAlchemy
- `tg-admin shell` get an iPython "DB session"
- what does this look like?

how does TG do page-display

- Kid XHTML/XML templating - “like” Zope Page Templates
- required: *well-formed* XHTML, on input & output
- what does this look like?
- examples: `tmpl_ex*.kid` & `tmpl_ex.py`
- in 1.0: *Cheetah* “free-form” templates

do URLs? controllers.RootController

- simple URL traversal: RootController methods or variables
- `@turbogears.expose()` "view" a page w/dict-values
- `@turbogears.validate()` validate and convert form-values
`turbogears.validators.Int()`
`turbogears.validators.Bool()`
- what does this look like?
- `gears2go/gears2go/controllers.py`

do Forms? using tg.widgets

- want built-in Form-value validation?
- one way: just use `turbogears.widgets` Forms

```
comment_form =
    tg.widgets.TableForm(fields=CommentFields(),
                        action='save')
.....
@expose(.templates.user_comment.kid)
@validate(form=comment_form)
def save(self, name, email, comment, notify=False):
    comments.add(name, email, comment)
    ....
```

do Forms? using tg.validators

- a rich variety of tg.validators methods
- simplify N-fields validation: subclass validators.Schema

```
class Items(controllers.Controller):

    @expose(template=''.templates.items_form'')
    @validate( validators = {
        'item': validators.PlainText(), # regex: [w-]
        'size': validators.OneOf(['S','M','L'], if_empty="M"),
        'price': validators.Money(not_empty=True),
    })
    def store_item(self, tg_errors=None, kwargs):
        if tg_errors:
            msg = [str(err) for err in tg_errors.values()]
            msg = ', '.join(msg)
            return dict(msg=msg)
        else:
            return dict(msg='Item Stored.'')
```

Summary

- TG is a solid “Turbo integration” of great Python web technology. **The “Py community” is on your side.**
- if you **grasp the M-V-C concept & know Python 2.3+**, you can get up-to-speed very fast
- 1.0 TurboGears is coming. **TG works, today!** Woo-hoo!
- 2000 subscribers to turbogears@googlegroups.com and growing
- 1.1 TG should be richer, faster, and easier on the eyes than “Ruby on Rails”

References

- Docs: <http://docs.Turbogears.org/1.0>
- <http://trac.Turbogears.org/turbogears/wiki/DocumentationPlayground>
- <http://Del.icio.us/popular/turbogears>
- <http://groups.Google.com/group/turbogears>