

Unil

UNIL | Université de Lausanne

```
int main(void) {  
    printf ("Hello World");  
    printf ("");  
}
```



HEC > MscBIS > Base de Données d'Entreprise

A Quick TurboGears Overview

Cédric Gaspoz (based on Christopher Arndt tutorial)

What are we going to cover?



- What is TurboGears?
- What makes it cool?
- Starting a project
- Using the Model, Views, and Controllers
- The Toolbox
- Some useful hints for your project

TURBOGEARS

the rapid web development megaframework
you've been looking for.



- A **Python** web meta framework!



- Comparable to Django and Ruby on Rails
- **Open Source** (MIT License)
- Buzzword compliant: MVC, AJAX, REST etc.



What can it be used for?

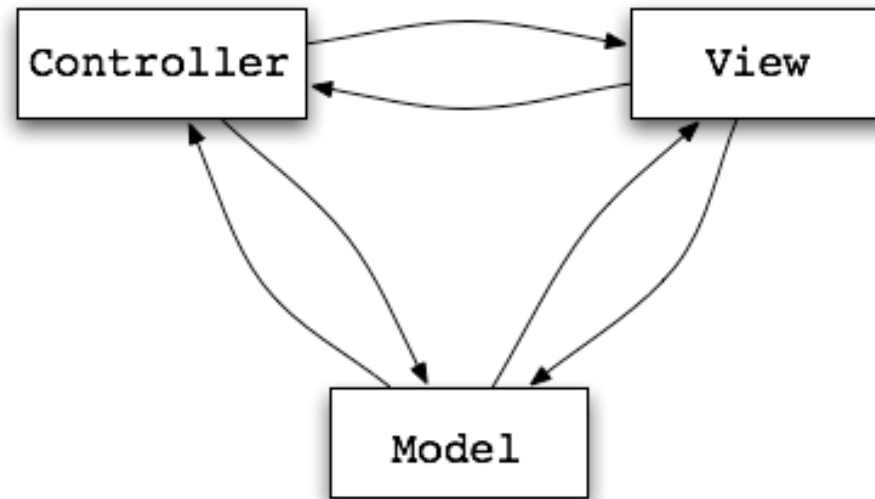


- „**Classic**“ web apps, e.g. Blogs, Wikis, CMS
- **Intranet apps**, e.g. WhatWhat Status or Web administration front ends, e.g. WebFaction.com Control Panel
- „**Microapps**“ (<http://microapps.org/>) à la Gravatar.com, Websnapr.com, etc.



See <http://docs.turbogears.org/1.0/SitesUsingTurboGears>

The Model-View-Controller pattern



- Web applications:
database / data retrieval methods / templates
- Goal: separation of components for easier replacement

Which components make up the TurboGears framework?



Client-side JavaScript:
MochiKit



Template engine:
Kid (Genshi in TG 1.1)



Application server:
CherryPy



Database abstraction:
SQLObject (SQLAlchemy in TG 1.1)

10 steps to your TurboGears application

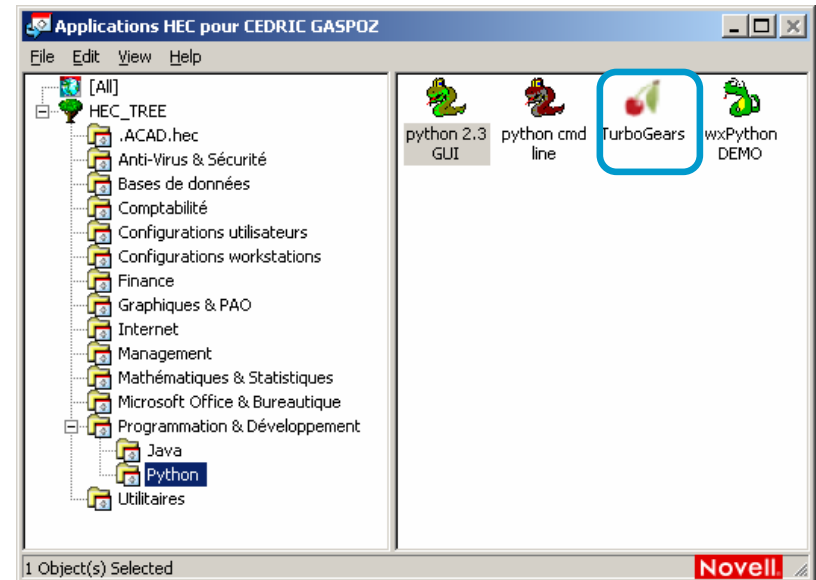


1. Quickstart your project
2. Code your data model
3. Create the database
4. Add some bootstrap data using CatWalk
5. Design your URLs
6. Write your controller methods
7. Write your templates
8. Add some CSS and/or JavaScript
9. Build an egg
10. Deploy!

Step 1: Quickstart your application



- Launch TurboGears on the Novell Application Launcher (NAL)



```
H:\turbogears> tg-admin quickstart
```

```
Enter project name: Bookmarker
```

```
Enter package name [bookmarker]:
```

```
Do you need Identity (usernames/passwords) in this project? [no] yes
```

```
[ long output follows...]
```

```
$ cd Bookmarker
```

So, what did that do?



- **dev.cfg** – Development config
- **README.txt** – How to start your program
- **sample-prod.cfg** – Production config
- **setup.py** – Build and release logic
- **start-bookmarker.py** – Run your program
- **test.cfg** – Database test connector
- **bookmarker/** – Directory where your code lives
- **Bookmarker.egg-info** – Egggy goodness

So, what did that do? (2)



H: \turbogears\Bookmarker\bookmarker

- **controllers.py** – Your logic
- **json.py** – How to represent your objects with JSON
- **model.py** – How to persist your objects
- **release.py** – Release info
- **config/** – Directory for application configuration
- **static/** – JavaScript, CSS, and image files
- **templates/** – Directory for Kid templates
- **tests/** – 3 free unit tests!

Starting your application



```
H:\turbogears\Bookmarker>python start-bookmarker.py
```

[long output follows...]

INFO HTTP: Serving HTTP on http://localhost:8080/

TURBOGEARS

✓ Your TurboGears application is now running.

1. MODEL
Design models in the `model.py`.
Start with a pre-configured SQLite database, or Edit `dev.cfg` to use a different backend.
Use `tg-admin sql create` script to create the database tables.

2. VIEW
Edit html-like templates in the `templates` folder.
Put static contents in the `static` folder;

3. CONTROLLER
Edit `controllers.py` and build your website structure with the simplicity of Python objects.
TurboGears will automatically reload itself when you modify your project.

Learn more
Learn more about TurboGears and take part in its development

- [Official website](#)
- [Documentation](#)
- [Trac \(bugs/suggestions\)](#)
- [Mailing list](#)

If you create something cool, please [let people know](#), and consider contributing something back to the [community](#).

TURBOGEARS G
under the hood

TurboGears is an open source front-to-back web development framework written in Python
Copyright © 2006 Kev in Dangpor

Step 2: Code you data model



- Two application-specific data objects:
 - Bookmarks
 - Tags
- TurboGears creates standard data objects for us:
 - Users, Groups, Permissions
- ModelDesigner
 - \$ tg-admin toolbox
 - [...]
 - HTTP INFO Serving HTTP on http://localhost:7654/
 - [...]

Documentation: <http://www.sqlobject.org/SQLObject.html>

ModelDesigner



Toolbox



Welcome to the TurboGears ToolBox

A collection of web based tools for TurboGears applications.



[System Info](#)

TurboGears System Information. Lists your TurboGears packages and version information



[CatWalk](#)

Model Browser. Administration tool for listing, creating, updating or deleting your SQLAlchemy instances



[WebConsole](#)

Web based Python interpreter



[ModelDesigner](#)

Designer for SQLAlchemy models. Create your Classes, define your fields and manage your relations. Visualize and generate code for SQLAlchemy models.



[Widget Browser](#)

Browse usage samples, description and source code for the available TurboGears Widgets



[admi18n](#)

i18N administration tool. Collect your strings, add and manage locales, edit and compile you catalogs

Bookmarker model



Toolbox » ModelDesigner



Sample models ▾ Load

Save Current Session

Clear Current Session

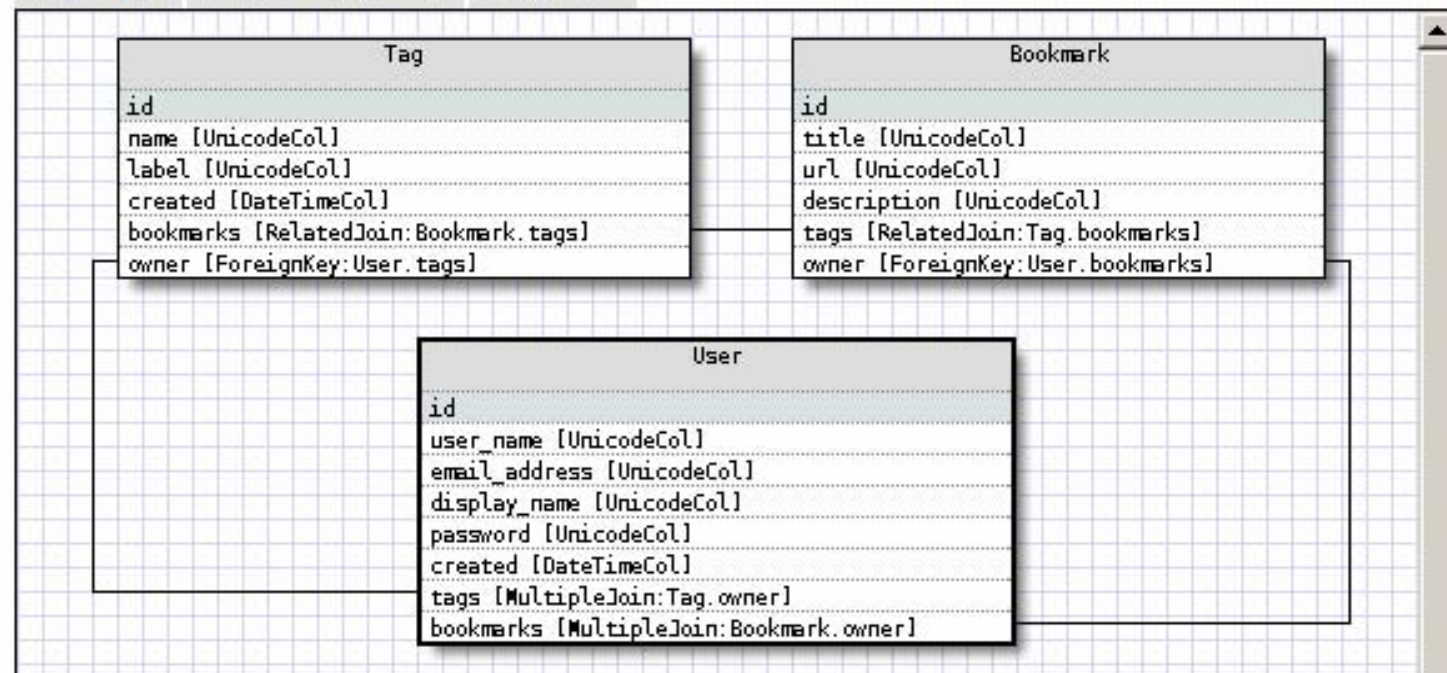
Add new class M

Bookmark

Tag

User

Settings Generate Code Diagram



Data model - Bookmark objects



in model.py:

```
class Bookmark(SQLObject):
```

```
    title = UnicodeCol(length=255, notNull=True)
```

```
    url = UnicodeCol(length=255, notNull=True)
```

```
    description = UnicodeCol()
```

```
    tags = RelatedJoin('Tag', orderBy='name')
```

```
    # meta data
```

```
    created = DateTimeCol(default=datetime.now)
```

```
    owner = ForeignKey('User', notNull=True)
```

Data model - Tag objects



still in model.py:

```
class Tag(SQLObject):
```

```
    name = UnicodeCol(length=100, notNull=True)
```

```
    label = UnicodeCol(length=100, notNull=True)
```

```
    bookmarks = RelatedJoin('Bookmark', orderBy='-created')
```

```
    # meta data
```

```
    owner = ForeignKey('User', notNull=True)
```

```
    created = DateTimeCol(default=datetime.now)
```

Step 3: Test the SQL script



Everything is already set up for the default SQLite backend:

```
$ tg-admin sql sql
```

Using database URI ...

```
CREATE TABLE bookmark (  
  id INTEGER PRIMARY KEY,  
  title VARCHAR(255) NOT NULL,  
  url VARCHAR(255) NOT NULL,  
  description TEXT,  
  created TIMESTAMP,  
  owner_id INT NOT NULL CONSTRAINT owner_id_exists REFERENCES  
  tg_user(id)  
);
```

Step 3: Create the database



Everything is already set up for the default SQLite backend:

```
H:\turbogears\Bookmarker> tg-admin sql create
```

Using database URI

```
sqlite:///H:\turbogears\Bookmarker/devdata.sqlite
```

Step 4: Add bootstrap data



TurboGears comes with a nice web administration interface called **CatWalk**.

We'll add groups, users and permissions and a few bookmarks and tags.

Toolbox » CatWalk



Bookmark
Group
Permission
Tag
User
VisitIdentity
Visit

Add Tag +

Records 0 - 10 (total:16) Page size:

#	owner	Created	Name	Label	Bookmarks
1	Test user	2006-12-05 00:51:00	python	Python	4
4	Test user	2006-12-14 22:16:59	homepage	Homepage	2
5	Test user	2006-12-14 22:43:12	pug	PUG	1
9	Test user	2006-12-14 22:45:18	framework	Framework	1
10	Test user	2006-12-15 00:22:26	application server	Application Server	1
11	Joe Doe	2006-12-23 04:28:49	python	Python	2
12	Joe Doe	2006-12-23 04:28:49	orm	ORM	1
15	Joe Doe	2006-12-23 04:38:28	german	German	1
17	Test user	2007-04-09 20:49:23	cologne	Cologne	1
18	Test user	2007-04-09 23:26:11	köln	Köln	

TURBOGEARS
UNDER THE HOOD

Step 5: Designing your URLs



http://mysite/bookmarks/

- /bookmarks/
- /bookmarks/<id>
- /bookmarks/<id>/**view**
- /bookmarks/<id>/**edit**
- /bookmarks/<id>/**add**
- /bookmarks/<id>/**delete**
- /bookmarks/<id>/**update**

List of bookmarks

Show bookmark details /
Show edit form

Delete/update bookmark

URL mapping



- **URL mapping** is the process of turning a request for a certain URL into a **function** or **method call** in your web application.
- **Example:**
<http://mysite.com/bookmarks/2/edit>
- Question: which part of the URL is the method name and which are the parameters?

URL mapping à la CherryPy



in controllers.py:

```
class BookmarkController(controller.Controller):
```

```
    @expose()
```

```
    def edit(self, id):
```

```
        return "The given ID is %s" % id
```

```
class Root(controller.RootController):
```

```
    bookmarks = BookmarkController()
```

URL: `http://mysite/bookmarks/edit/2`

Resulting call: `Root().bookmarks.edit(2)`

CherryPy REST URL mapper



```
@expose()
def default(self, *params, **kw):
    if len(params) == 1:
        id = params[0]
        redirect(url('%s/view' % id))
    elif len(params) >= 2:
        id, verb = params[:2]
        action = getattr(self, verb, None)
        if not action or not getattr(action, 'exposed'):
            raise cherrypy.NotFound
        action(item, *params[2:], **kw)
```

Step 6: Write controller methods



- We need the following methods:
 - Show a welcome page*
 - Show list of bookmarks
 - Show bookmark details / edit form
 - Show form for new bookmark*
 - Create/Update bookmark from form submission
 - Delete bookmark

* left as exercise for the reader

Controller methods

List of bookmarks



in controllers.py:

```
class BookmarksController(controllers.Controller):  
    @expose(template='bookmarker.templates.list')  
    def index(self):  
        bookmarks = Bookmark.select()  
        return dict(entries=bookmarks)  
list = index
```

Controller methods

Bookmark details/edit form



still in controllers.py:

```
class BookmarksController(...):
```

```
    [...]
```

```
    @expose(template='bookmarker.templates.edit')
```

```
    def view(self, id, *params, **kw):
```

```
        try:
```

```
            bookmark = Bookmark.get(id)
```

```
        except SQLAlchemyObjectNotFound:
```

```
            flash('Bookmark not found.')
```

```
            redirect('/')
```

```
        return dict(entry=bookmark)
```

Controller methods

Update/Create bookmark



```
@expose()
def update(self, id, *params, **kw):
    try:
        bookmark = Bookmark.get(id)
    except SQLAlchemyObjectNotFound:
        bookmark = Bookmark(
            title = kw.get('title'),
            url = kw.get('url'),
            description = kw.get('description'))
    else:
        bookmark.set(title = kw.get('title'), url=...)
    # TODO: handle tags specially
    redirect('/bookmarks/')
```

Controller methods

Delete bookmark



```
@expose()
def delete(self, id, *params, **kw):
    try:
        Bookmark.delete(id)
    except SQLAlchemyObjectNotFound:
        flash('Bookmark not found.')
    else:
        flash('Bookmark deleted.')
    redirect('/bookmarks')
```

Step 7: Edit templates

List of bookmarks



```
<div py:if="entries" class="bookmarks">
  <dl py:for="bookmark in entries">
    <dt><a href="{bookmark.url}"
      py:content="bookmark.title" /></dt>
    <dd><p py:content="bookmark.description" />
      <p><a href="{tg.url('/bookmarks/view/%i' %
        bookmark.id)}">Edit</a></p></dd>
  </dl>
</div>
<div py:if="not entries" class="bookmarks">
  No bookmarks found
</div>
```

Edit templates

Show bookmark / edit form



```
<form action="/bookmarks/update/${entry.id}"
  method="POST">
  <input type="text" name="title" value="${entry.title}" />
  <input type="text" name="url" value="${entry.url}" />
  <textarea name="description">
    ${entry.description}
  </textarea>
  <input type="text" name="tags"
    value="${','.join([tag.name for tag in entry.tags])}" />
  <input type="submit" value="Save">
</form>
```

Step 8: Add CSS and/or JavaScript



Edit `static/css/style.css` and give your application a facelift:

The screenshot shows a web application interface for TurboGears. At the top, there is a blue header with the text "TURBOGEARS". Below the header, there are navigation links: "Home", "Bookmarks", and "Users". On the right side, there is a user greeting: "Welcome Test user. [Logout](#)".

The main content area contains a form for managing bookmarks. The form has the following fields:

- Title:** TurboGears: Front-to-Back Web Development
The link title. Required
- URL:** http://turbogears.org/
The URL of the bookmarked page. Required
- Description:** Create a database-driven, ready-to-extend application in minutes. All with designer friendly templates, easy AJAX on the browser side and on the server side, not a single SQL query in sight with code that is as natural as writing a function.
Short description of the bookmarked page. Optional
- Tags:** Framework, Python, Web development
A comma-separated list of tags. Optional

At the bottom of the form, there is a button labeled "Update bookmark".

At the bottom of the page, there is a footer that says "Powered by TurboGears".

Step 9: Build an egg



- Edit **release.py** to add **package meta data**.
 - H:\turbogears\Bookmarker> **python setup.py bdist_egg**
- Copy egg to target host and do
 - H:\turbogears\Bookmarker> cd dist
 - > **easy_install** Bookmarker-1.0-py2.4.egg

See <http://docs.turbogears.org/1.0/DeployWithAnEgg> for more information

Step 10: Deployment options



- Pure CherryPy-Server (for development/testing)
- Apache with **mod_proxy** (recommended)
- Apache with **mod_python**
- Alternative light-weight webservers:
 - nginx
 - LightTTP

Conclusion



- We edited **3 Python source code** files:
 - model.py
 - controllers.py
 - release.py
- We edited **3 Kid template** files:
 - welcome.kid
 - list.kid
 - edit.kid
- Plus **some CSS**
- and **no SQL statement** in sight!

What's next?



- Read the book:
 - <http://www.turbogearsbook.com/>
- Visit the Wiki:
 - <http://docs.turbogears.org/>
- Easy forms with TurboGears **widgets**:
 - <http://docs.turbogears.org/1.0/Widgets>
- The future: **SQLAlchemy and Genshi**:
 - <http://docs.turbogears.org/1.0/SQLAlchemy>
 - <http://docs.turbogears.org/1.0/GenshiTemplating>
- **Develop your SCUSI application!**

tg-admin



\$ **tg-admin**

TurboGears 1.0.1 command line interface

Commands:

i18n Manage i18n data

info Show version info

quickstart Create a new TurboGears project

shell Start a Python prompt with your database available

sql Run the database provider manager

toolbox Launch the TurboGears Toolbox

update Update an existing turbogears project

SQLObject Inheritance



```
from sqlobject.inheritance import InheritableSQLObject
```

```
class Ressource(InheritableSQLObject):
```

```
    class sqlmeta:
```

```
        idName = 'ressource_id,
```

```
        typeressource = ForeignKey('TypeRessource')
```

```
        description = UnicodeCol(length=255)
```

```
        depphase = ForeignKey('Phase')
```

SQLObject Inheritance (II)



```
class TempsCalcul(Ressource):
    class sqlmeta:
        idName = 'temps_calcul_id',
        _inheritable = False
        libelle = UnicodeCol(length=25)
        heure_devis = DecimalCol(size=10, precision=2)
        tauxhoraire = CurrencyCol()
    def _get_cout_devis(self):
        cout = self.heure_devis * self.tauxhoraire
        return cout
```

<http://www.sqlobject.org/Inheritance.html>